Garceta
grupo editorial

Técnico Superior en **Administración de Sistemas Informáticos en Red**

Implantación de Aplicaciones Web

Jorge Sánchez Asenjo

Implantación de Aplicaciones Web

Jorge Sánchez Asenjo

Técnico Superior en **Administración de Sistemas Informáticos en Red**

Implantación de Aplicaciones Web

Garceta
grupo editorial

Implantación de Aplicaciones Web

Jorge Sánchez Asenjo

ISBN: 978-84-1622-830-0
IBERGARCETA PUBLICACIONES, S.L., Madrid 2015

Edición: 1.ª
Impresión: 1.ª
N.º de páginas: 476
Formato: 20 x 26 cm

Implantación de Aplicaciones Web

© Jorge Sánchez Asenjo

1.ª edición, 1.ª impresión
OI:471/2016
ISBN: 978-84-1622-830-0
Deposito Legal: M-24648-2015
Imagen de cubierta: © Andrés Sanz Mulas

Impresión:
PRINT HOUSE, marca registrada de Coplar, S. A.

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

PRÓLOGO

Este libro está escrito con la vocación de ser utilizado como libro de texto del módulo de **Implantación de Aplicaciones Web** del ciclo de Grado Superior de Formación Profesional de **Administración de Sistemas Informáticos en Red** (ASIR, como es conocido por alumnos y profesores).

Por ello se ha puesto especial atención en ayudar a cumplir los resultados de aprendizaje y contenidos expresados en el R.D. 1629/2009 de 30 de octubre en el que se establece el título de **Técnico Superior en Administración de Sistemas Informáticos en Red**.

Soy profesor del módulo de Implantación de Aplicaciones Web desde el año 2011 (el primer año en el que se pudo impartir esta materia). He sido profesor de muchos otros módulos. También he sido profesor de Formación Ocupacional, Formación Continua y Educación Secundaria. He tenido muy en mente el problema que tenemos los profesores al impartir una nueva materia.

Por ello, mi pretensión principal ha sido escribir el libro que yo mismo, como docente, me gustaría utilizar.

Creo también que este libro puede ser de gran utilidad para aquellas personas y profesionales que, sin ser alumnos o docentes de este módulo, quieran empezar a formarse o profundizar en esta interesante materia.

Cada unidad está escrita la siguiente forma:

- Se exponen los contenidos teóricos de la misma tratando de fomentar la aplicación inmediata de los mismos, para lo cual se proponen algunas actividades.
- Tras los contenidos se plantean prácticas que se explican y resuelven en el propio texto y que se proponen para realizar a la vez que se estudia la parte teórica. Hay, además, otras prácticas propuestas, cuya solución está a disposición de los docentes que deseen utilizar esta obra como libro de texto en sus clases.
- Hay una página final de resumen de la unidad con las ideas más interesantes vistas en la misma.
- Al final de cada unidad se han elaborado preguntas de tipo test para evaluar la comprensión del texto.

La disposición de los capítulos y contenidos se han escrito en un orden en el que los conocimientos que se van adquiriendo ayudan a asimilar los siguientes. Pero es perfectamente posible alterar este orden o centrarse en temas que coincidan más con los intereses del lector.

A este respecto quiero señalar que los contenidos del libro se pueden dividir en cinco bloques:

[1] **Teoría sobre las aplicaciones web**, unidad 1. Se trata de una unidad eminentemente teórica, en la que se explica qué son las aplicaciones web y cuál es su relación y evolución con la web

actual. Se profundiza más allá de los contenidos mínimos, para dar un sentido histórico y de contexto a la materia, abordando los principales conceptos y elementos que intervienen en la creación de aplicaciones web y lo que aportan a los usuarios que las utilizan.

[2] **Preparación del entorno de trabajo**, unidad 2. Este bloque estudia cómo implementar la tecnología central del libro, **Apache**, **PHP** y **MySQL**, que sigue siendo la dominante hoy en día. Se trabaja, de manera práctica, su instalación en sistemas tanto Windows como Linux, y, además, aprendiendo las diferentes formas de hacerlo: totalmente manual, a través de instaladores, mediante paquetes completos integrados, etc.

Esto permite disponer de un entorno totalmente adaptado a nuestros intereses e incluso armar varios sistemas que simulen las diferentes fases profesionales de trabajo.

[3] **Programación en PHP**, unidades de la 3 a la 6. Se estudian las bases de la programación en este lenguaje. Siendo un lenguaje tan extenso y con tantas posibilidades, el conocimiento completo del mismo daría, no para un libro sino para bastantes más.

No obstante, no se ha pretendido pasar por encima del lenguaje. La materia cubierta en el libro permite crear aplicaciones web propias interesantes y facilitar la posterior profundización en el lenguaje, si el lector así lo quisiera.

[4] **Sistemas de gestión de contenidos**, unidades 7 a 10. Tras una primera unidad que explica la utilidad y ventajas del uso de **CMS** (Sistemas de Gestión de Contenidos), las tres siguientes unidades recorren las acciones fundamentales de trabajo con estas herramientas. Se utiliza para ello a **WordPress** y **Drupal** como modelo explicativo, ya que ambas son los ejemplos más interesantes de CMS por sus capacidades y grado de implantación.

[5] **Implantación de aplicaciones de ofimática web**, unidad 11. Se estudian las ventajas y desventajas de este servicio. Tras comparar los servicios más utilizados hoy en día, se estudia el funcionamiento completo de **Google Docs**.

Teniendo en cuenta los bloques, se puede leer en otro orden los contenidos o focalizar la atención en aquellos que se acomoden a las necesidades del lector.

Apache, PHP, MySQL, WordPress y Drupal, bajo los sistemas Windows y Linux, son las tecnologías sobre las que camina este texto. La idea no es tanto dominar estas tecnologías concretas, sino explicarlas con una profundidad suficiente como para trabajar profesionalmente con ellas o migrar a otras tecnologías sin grandes dificultades; ya que espero que las bases del funcionamiento de este tipo de herramientas se hayan cubierto tras la lectura de este libro.

Desde la URL http://www.jorgesanchez.net/libroiaw, se puede encontrar material extra relacionado con el libro. En esa misma dirección se pueden encontrar formas de contacto conmigo, así como otros textos, ejercicios y presentaciones de otras materias.

Siempre recibo con gratitud vuestras aportaciones ya que gracias a ellas creo seguir mejorando los contenidos que escribo.

Jorge Sánchez Asenjo
@jorgesancheznet

ÍNDICE GENERAL

UNIDAD 1

CONCEPTOS SOBRE APLICACIONES WEB

OBJETIVOS

- Reconocer la diferencia entre una aplicación web y una aplicación de escritorio
- Valorar los cambios que han sufrido las aplicaciones web a lo largo de la historia
- Identificar el estado actual de implantación y funcionamiento de las aplicaciones web
- Distinguir los elementos que forman parte de la arquitectura de una aplicación web moderna
- Asimilar la idea, implicaciones y funcionamiento básico de una aplicación en la nube
- Distinguir la diferencia entre una aplicación web y una aplicación móvil
- Identificar las principales tecnologías del lado del cliente y del lado del servidor

CONTENIDOS

1.1 EVOLUCIÓN DE LA WEB E INTERNET

1.1.1 PERSPECTIVA HISTÓRICA. EL ÉXITO DE LAS APLICACIONES WEB

El éxito de Internet está absolutamente ligado a la web. Tanto que hoy en día para la inmensa mayoría de las personas es indistinguible qué es la web y qué es Internet. Todo se hace en la web. Las tareas que antes requerían el uso de protocolos distintos a **http**, como el correo electrónico, los grupos de debate, chats... Ahora se realizan mediante servicios accesibles desde el protocolo http, usando un cliente de navegación web (un navegador, en definitiva).

Esto no ha sido así siempre. Inicialmente en Internet no había páginas web. Tenía servicios como el correo electrónico, la transmisión de ficheros o los grupos de noticias, que permitían realizar muchas de las tareas que se siguen realizando actualmente. Pero no había ***World Wide Web***. Los diferentes servicios se realizaban a través de software cliente especializado en la tarea concreta, lo que hacía que un usuario de Internet tuviera toda una pila de aplicaciones para acceder a los diferentes servicios, y esas aplicaciones no eran precisamente fáciles de manejar; solo los profesionales de la informática se sentían cómodos con ellos.

Así para leer el correo electrónico hacía falta un cliente de correo, la transmisión de ficheros se realizaba mediante un software cliente FTP, los foros de debate se leían a través de gestores de news, para el chat hacía falta también una aplicación especial y así ocurría con cada servicio que ofrecía Internet.

Esos servicios se siguen pudiendo utilizar de esa forma hoy en día; por ejemplo, se puede utilizar software como **ThunderBird** u **Outlook** para leer el correo en lugar de hacerlo a través de un navegador web. Pero ya no es lo habitual; los usuarios actuales realizan todas las tareas relacionadas con Internet a través de un único servicio: la web.

La cuestión es ¿por qué? Hay varias razones para el éxito absoluto de la web:

- **Su manejo es sencillo**. Unos simples clics de ratón nos permiten navegar por los diferentes contenidos y servicios. Los usuarios entienden rápidamente esa forma de trabajar.
- **Es muy visual**. La web fue el primer servicio que permitió mostrar contenidos con imágenes, sonidos y vídeos. Antes de ella, los contenidos que ofrecía Internet eran solo textos. Actualmente una simple página web es capaz de mostrar cualquier elemento visible desde una pantalla, y proporcionar tanta interactividad a los contenidos como la pericia y la imaginación del creador de la página permitan.
- **Solo necesitamos un único software**. El navegador es el software que permite hacer peticiones **http** e interpretar el **HTML** (y el **CSS**, **JavaScript**....) necesario para ver adecuadamente una página web. Además, es un software incluido en todos los sistemas electrónicos, incluidos los dispositivos móviles; es decir, es una aplicación que seguro que el usuario tiene instalada, sea cual sea su sistema.

Inicialmente las páginas web estaban compuestas de texto y enlaces (a otras páginas), pero en poco tiempo, los usuarios de las páginas web demandaron que estas ofrecieran más servicios: no solo mostrar estáticamente un texto fijo, sino que la información también incluyera imágenes, vídeos, animaciones,... y poco a poco: acceso a servidores de bases de datos, manejo del correo electrónico, transmisión de ficheros, compra y venta de productos, etc.

Hoy en día desde la Web se puede hacer cualquier tarea, editar documentos, leer el correo electrónico, enviar mensajes, retocar fotos, ver películas, escuchar radio o incluso administrar un servidor remoto.

La web parece el servicio definitivo, pero también tiene sus pegas. A medida que las páginas se hacen más complejas, se requiere en el navegador más componentes (***plugins***) para poder verlas bien.

Componentes como la máquina virtual de ***Java***, el plugin de ***Flash*** o el de ***Silverlight*** o un intérprete veloz de ***JavaScript,*** pasan a ser imprescindibles y hacen que el navegador sea un software cada vez más complejo que exige al usuario la instalación esos componentes para disfrutar adecuadamente de los contenidos.

En todo caso el problema de los plugins ha pasado a ser un problema menor, ya que los navegadores ya suelen traer hoy en día los plugins mínimos necesarios y los creadores de aplicaciones web se adaptan a esos mínimos. Incluso los contenidos hechos en ***Flash***, que era la tecnología más habitual para crear aplicaciones web enriquecidas, se van retirando de las aplicaciones web para sustituirse por tecnologías equivalentes compatibles con HTML5.

En la actualidad solo el mundo de las ***apps*** en los dispositivos móviles parece arrojar alguna competencia al éxito de las aplicaciones web.

1.1.2 HISTORIA DE LA WEB

La creación de aplicaciones web ha tenido numerosos cambios a lo largo de la breve (aunque ya no tanto) historia de Internet. Se señalan a continuación algunos hechos importantes en la historia de las aplicaciones web:

- 1989
  - **Tim Bernes Lee**[1], científico británico que trabajaba en el **CERN**[2], el centro de desarrollo nuclear ubicado en Suiza, intenta trasladar el hipertexto a los documentos científicos, mediante el cual es posible avanzar de un documento a otro mediante enlaces existentes en el propio texto. Teorizó la forma de transportar este tipo de documentos (el actual protocolo **http**) y sobre el lenguaje de marcas a utilizar.
- 1990
  - Aparece el primer navegador web, llamado **WorldWideWeb** (más tarde **Nexus**) para realizar pruebas, se creó para un ordenador **Next.**

1 Premio Príncipe de Asturias 2002 junto a **Lawrence Roberts**, **Robert Kahn** y **Vinton Cerf**: considerados los ***padres*** de Internet

2 http://public.web.cern.ch/public/

- 1991
  - **Tim Bernes Lee** acude a un grupo de discusión en Internet para discutir sobre cómo implementar el hipertexto de forma más conveniente. Con ello no pretende privatizar su invento, sino hacerlo público desde el primer momento.
- 1992
  - **Pei-Yuan Wei** crea **ViolaWWW**, considerado el verdadero primer navegador, era capaz de interpretar un lenguaje de script además del propio HTML; era un producto muy innovador.
- 1993
  - El **CERN** anuncia que la web será libre para todo el mundo. Renuncia a sus posibles patentes.
  - La **NCSA**[3] se interesa por la, ya llamada, **web** y crea el primer navegador realmente exitoso: **Mosaic.** Entre sus creadores está **Eric Bina y Marc Adreessen** y participó la universidad de **Urbana-Champaign**.
  - **Lou Montulli** desarrolla **Lynx** para los sistemas **Unix**, el primer navegador de texto en la web. Será ampliamente utilizado en los años siguientes, aunque luego quedará rápidamente superado por las capacidades de los navegadores gráficos.
  - Se empieza a hacer popular la etiqueta **img** (gracias a Mosaic), las imágenes empiezan a poblar la web.
  - Aparece el primer borrador de HTML gracias a **Tim Bernes-Lee** y **Dan Conely**.
- 1994
  - Conferencia global sobre la web.
  - La **IETF**[4] asigna un grupo de trabajo para estandarizar HTML.
  - El lenguaje HTML empieza a ser caótico porque aparecen numerosas etiquetas inventadas por cada entidad privada. **Dan Connolly** recopila las etiquetas HTML más utilizadas de la época y se crea el borrador de **HTML 2.**
  - **Marc Adreessen** y **Jim Clark** abandona la NCSA y fundan **Mosaic Communications** (futura **Netscape**). Dejan también los estándares y crean elementos nuevos en el lenguaje HTML para permitir la creación de páginas más vistosas para su navegador.
  - **Guido Van Rossum** crea **Python**, considerado uno de los lenguajes más cómodos y fáciles de entender. Se ha utilizado y se utiliza extensamente para programar aplicaciones web en los servidores.
  - A finales del año se crea la **World Wide Consortium**[5] **(W3C)** fichando a algunos de los principales impulsores de la web (incluido **Tim Bernes Lee**). Se convertirá en el

3 **National Center for Supercomputing Applications**, Centro Nacional de Estados Unidos de Supercomputación, creador de las primeras grandes redes de cálculo y supercomputadoras.

4 **Internet Engineering Task Force**, grupo que estandariza diferentes aspectos de Internet

5 http://www.w3.org y en español http://www.w3c.es/

principal organismo de estandarización de las tecnologías relacionadas con la web en general y de HTML en particular.

- La empresa **Mosaic Communications** se convierte en **Netscape Communications** y lanza el navegador **Netscape Navigator**. En los siguientes años será el navegador más utilizado.
- A finales de año hay 10.000 servidores web.

1995

- Aparece **CGI**, *Common Gateway Interface*, Interfaz de Pasarela Común que permite incorporar código programado en lenguajes avanzados (C, Perl...) y conseguir comunicar esos programas con las páginas web a fin de dotarlas de servicios más avanzados.
- Aparece **Windows 95.** Con él, la informática de consumo llega a casi todos los hogares del mundo desarrollado.
- Siguen apareciendo nuevos elementos en HTML que impulsan las posibilidades de las páginas web. Se crea el borrador **HTML 3**, que incluye tipos de letra y otras mejoras.
- **Microsoft** crea **Internet Explorer** y lo incorpora rápidamente como parte del sistema operativo Windows 95. Comienza la primera guerra de los navegadores entre el propio Explorer y Netscape Navigator.
- El grupo de trabajo de la IETF para HTML se desmantela por su escasa influencia. El **World Wide Consortium** queda como principal organismo de estandarización de HTML.
- **Hakom Wum Lie** de la empresa noruega **Telenor**, crea el navegador **Opera**. Nunca ha alcanzado una gran cuota de público, pero sigue presente después de tantos años.
- A finales de año aparecen los primeros elementos de creación de hojas de estilo, raíz del lenguaje **CSS** que permite dar formato avanzado a las páginas web y que sigue siendo una de las tecnologías imprescindibles en la actualidad para crear páginas web.
- **Sun Microsystems** crea el lenguaje **Java**, que tendrá una enorme influencia en el desarrollo de las aplicaciones web.
- Se lanza el servidor http de código abierto **Apache**. Todavía hoy en día sigue siendo el software más utilizado para implementar servidores web.
- Los hermanos **Allaire**, crean **ColdFusion**, un lenguaje de script sobre HTML que se ejecuta en el servidor que aloja las páginas web (servidores compatibles con esta tecnología) de modo que el cliente no necesita tener un software especial que reconozca esta tecnología. Al cliente le llegan páginas web normales que ha preparado el servidor tras traducir este lenguaje.

  Será el primero de los lenguajes de script en el lado del servidor.
- Netscape desarrolla **JavaScript**, un lenguaje basado en C y Java que se incrusta dentro del código HTML de las páginas para darles una mayor potencia. Hoy en día sigue

siendo uno de los lenguajes más influyentes en el desarrollo de páginas y aplicaciones para la web. **Netscape Navigator 2.0** es el primer navegador en utilizar este lenguaje. Debido a su éxito, todos los navegadores tuvieron que ir incorporando un intérprete de este lenguaje entre sus capacidades básicas.

- Aparece la primera versión de **MySQL** base de datos relacional de código abierto (actualmente pertenece a la empresa **Oracle**). Se convertirá en el sistema gestor de base de datos más utilizado (especialmente en sitios pequeños) de Internet.
- **Explosión de la web**, Internet se empieza a hacer muy popular y la web se empieza a convertir en el servicio más habitual para acceder a Internet. Gradualmente el número de sitios va creciendo hasta alcanzar los millones de sitios web en Internet.
- Se crea el sitio web **Geocities**, antecesor de las redes sociales. Los usuarios publicaban fácilmente sus propias páginas web dentro del propio sitio de Geocities y se agrupaban por gustos en ***ciudades virtuales*** que el resto de usuarios podía recorrer.

- 1996
  - Se crea el **HTML ERB** (***Editorial Review Board***), en el que participan empresas como **IBM, Microsoft, Netscape, Novell**... y el propio **W3C**. Es una reunión trimestral para ayudar en el estándar.
  - Se crea **Yahoo!** la primera página exitosa que permite organizar la web para facilitar la búsqueda de otras páginas. Rápidamente se convertirá en el sitio web más popular de Internet y en el primer índice de la Web. La empresa será una de las de mayor valor en bolsa durante varios años.
  - **Rasmus Lerdorf** crea el lenguaje script del lado del servidor **PHP**. Todavía sigue siendo la tecnología más popular para crear aplicaciones web en el lado del servidor.
  - La empresa **Macromedia** lanza el software **Flash** (creado inicialmente por la empresa **FutureWave**). Se trata de una tecnología del lado del cliente (requiere un plugin en el navegador) que permite crear contenidos multimedia muy avanzados. Así las páginas con Flash incluían todo tipo de elementos visuales e interactivos que enriquecían enormemente la experiencia de usuario. En los siguientes años se convertiría en la tecnología más influyente para crear aplicaciones web enriquecidas.
  - La **W3C** estandariza **CSS1** su popularidad aumenta rápidamente.
  - Se lanza **Hotmail** el sitio web que permite utilizar correo electrónico desde la web. Se convierte en el correo electrónico más popular y acaba siendo comprado por **Microsoft**.

- 1997
  - **Escándalo Lewinsky**. Los informes oficiales y detalles sobre el escándalo aparecen antes en la web que en los medios tradicionales de información. El público se acostumbra a leer noticias en la web, lo que poco a poco iniciará una crisis en los medios tradicionales que ven muy reducido su mercado.

- Aparece la especificación estándar **HTML 3.2**, la primera en ser ampliamente aceptada. Incluye tablas, applets (pensadas para añadir elementos Java a las páginas) y otros formatos avanzados de diseño.
- **Sun Microsystems** crea **Java Servlets** y **Microsoft** crea el lenguaje **ASP**. Son dos tecnologías del lado del servidor que tendrán una gran influencia en los años siguientes.
- Versión 4 de **Internet Explorer** y **Netscape Navigator**. Se recrudece la guerra entre estos navegadores.

■ 1998

- La W3C publica los estándares **HTML 4.0** y **CSS 2**. Los estándares de la W3C cada vez se tienen más en cuenta y ambos alcanzan un gran éxito rápidamente.
- La combinación **HTML+JavaScript+CSS** se conoce desde este año como **DHTML** (HTML dinámico). Alcanzará una enorme notoriedad y será la combinación habitual para hacer páginas web atractivas sin utilizar plugins de terceros.
- Aparece **XML 1.0** por parte de la W3C. En en palabras del propio **Tim Bernes Lee**, *"XML es el lenguaje que debió ser HTML"*. No ha llegado a suplantar a HTML, pero sigue teniendo una enorme influencia en todo tipo de tecnologías.
- **Netscape** crea la fundación **Mozilla** para mejorar el código del navegador **Netscape Navigator** que pasará a ser un software de código abierto. Se crea el motor **Gecko** núcleo futuro de varios navegadores (especialmente de **Firefox**).
- Se funda la **Web Standards Project** fundación encargada de promover el uso de los estándares HTML. Es famosa por la creación de los test **Acid** que validan el respeto de los estándares por parte de los navegadores.

■ 1999

- **Sun** crea **JSP** (páginas de servidor en lenguaje Java) y la plataforma de trabajo **J2EE** (**Java Enterprise**) con lo que pretende crear un entorno poderoso de trabajo para crear aplicaciones y servicios de Internet.
- El navegador **Internet Explorer** de **Microsoft** empieza a dominar el mercado.
- Se crea el test **Acid 1** (**CSS Acid Test**) para testear el funcionamiento de HTML 4 y CSS 1.
- Aparece **RSS** un formato de contenido basado en XML que permite la sindicación de contenidos y así obtener información sobre temas deseados de forma veloz.
- La **W3C** presenta **HTML 4.01**, indicando que será la última versión estándar del lenguaje HTML.
- **Microsoft** crea para **Internet Explorer 5** un nuevo elemento XML llamado **XMLHttpRequest** que será copiado en los demás navegadores en los años sucesivos. El acceso a este elemento desde JavaScript permite crear páginas muy dinámicas y potentes, ya que pueden mostrar en un elemento HTML respuestas a peticiones http. A este forma de programar se le llamará (hacia el año 2005) **AJAX**, acrónimo de ***Asynchronous JavaScript and XML***

- 2000
  - Aparece **Wikipedia**, enciclopedia creada con la colaboración desinteresada de miles de personas cuyos artículos crecen de manera exponencial cada año.
  - Aparece el estándar W3C **XHTML 1.0**, versión de HTML basado en XML que pretende derrocar a HTML. Fue muy popular hasta aproximadamente el año 2011 con el inicio de la influencia de HTML 5.
  - **ISO** (organismo internacional de estándares) publica la norma **ISO 15445** con la que normaliza HTML. Esta norma es prácticamente la misma que la correspondiente al HTML 4.01 de la W3C.
  - **Roy T. Fielding** publica su tesis doctoral de la que aparece la idea de las web de tipo **REST**, ***Representational State Transfer*** (Estado Representativo de Transferencia), o **RESTful**, que consiste en que las direcciones (URL) de las aplicaciones web en Internet representen el servicio al que dan acceso. Su influencia ha sido enorme, especialmente en los últimos años.
  - Se lanza **Konqueror 2.0** para el sistema **Linux** con ventanas **KDE** navegador basado en el motor **KHTML** raíz del motor **Webkit** en el que se basarán los navegadores Safari y Chrome entre otros.
- 2001
  - Aparece **Explorer 6** integrado en el popularísimo sistema operativo **Windows XP**. Con ello Microsoft gana la primera guerra de navegadores.
  - La W3C lanza el estándar **XHTML 1.1** último XHTML considerado por la industria.
  - **PHP**, como tecnología en el lado del servidor, y **Flash**, en el lado del cliente, son las tecnologías dominantes para crear aplicaciones web enriquecidas (ya conocidas como **RIA**, ***Rich Internet Applications***).
  - Los sucesos del **11S** provocan un colapso en Internet, los principales periódicos digitales del planeta muestran solo información muy esquemática sobre los atentados para ahorrar ancho de banda y poder seguir atendiendo a los usuarios.
  - Estallido de la **burbuja punto com** (***Dot-com Buble***), cientos de empresas que habían tenido ganancias enormes, pero especulativas, comienzan a desplomarse y desaparecer. Se habla de redefinir la web. Al año siguiente ya se habla de la **web 2.0.**
  - Se populariza el uso de blogs (o weblogs) para publicar en Internet, al ser más fáciles de crear por todo tipo de usuarios.
  - Se crea **Drupal**, herramienta de gestión de contenidos (CMS) que permite crear sitios colaborativos. Su éxito verdadero comienza a partir del 2007, compitiendo con Joomla como principal sistema de administración de contenidos.
  - Se crea **Bit torrent** un protocolo para compartir archivos entre equipos ***peer to peer*** (de igual a igual), sin servidor central. Tiene un gran éxito e influencia en los años siguientes.

- El sitio web **Napster** utilizado para descargar música, pierde juicios contra las leyes de autor y es obligada a cerrar.

2002

- La fundación **Mozilla** recoge el código liberado por Netscape y crea su propio navegador **Firebird** (futuro **Firefox**).
- Microsoft crea la plataforma de aplicaciones **.NET** con vocación de competir con la plataforma Java Empresarial, **J2EE**.
- Apple crea una bifurcación del proyecto KHTML y lo llama **Webkit**. Este será el motor del futuro **Safari** y de **Google Chrome**.
- Aparece **MySpace** popular web social de éxito espectacular en esos años.

2003

- Apple lanza al mercado el navegador **Safari**.
- Se crea **Wordpress**, gestor de contenidos web **CMS**. Permite crear páginas web (especialmente blogs) sin prácticamente conocimientos técnicos y facilita la gestión de usuarios que editen diferentes contenidos en un sitio web. A día de hoy, es el CMS más popular.
- **Chris Pederik** crea la **Web Developer Toolbar** para Firefox, barra de herramientas para desarrolladores, que permite inspeccionar cada elemento de una página web; muy importante para que los creadores de aplicaciones web puedan depurar su código.
- Aparece la red social gráfica y virtual **Second Life**. Tuvo un gran éxito en esos años.
- Se crean las redes sociales profesionales **Linkedin** y **Xing** (llamada inicialmente **Open Social Club**).

2004

- Se comercializa el navegador **Firefox**, comienza la segunda guerra de navegadores.
- Se forma el **WHATWG** para conseguir un HTML versión 5 que se convierta en nuevo estándar. Lo impulsan **Opera** y **Mozilla,** principalmente, y es rápidamente apoyado por **Apple** y **Google** para contrarrestar el dominio de **Microsoft**.
- Aparece **Facebook,** verdadera red social digital cuyo éxito desborda en los siguientes años hasta llegar a cientos de millones de usuarios.
- Aparece **Gmail** servicio de correo web de Google, que poco a poco, se populariza por el tamaño de su buzón y por sus grandes prestaciones y experiencia de usuario. El resto de servicios de correo web van mejorando también sus prestaciones.

2005

- **AJAX**, tecnología que combina JavaScript, HTML, CSS y XML se populariza pasando a ser una de las tecnologías fundamentales para crear páginas web dinámicas. Gracias a esta tecnología comienza la llamada **tercera era de la web**.

- Se incorporan patrones **MVC (Modelo-Vista-Controlador)** a la mayoría de tecnologías de creación de aplicaciones web, facilitando a los desarrolladores el mantenimiento y diseño de sus aplicaciones.
- Se lanza **Joomla!** En los siguientes años se convertirá en el CMS más popular (después será rebasado por WordPress).
- Se lanza el framework **Ruby on Rails** diseñado por **David Heinemeier** que permite crear aplicaciones web basadas en **MVC (Modelo-Vista-Controlador)** de forma sencilla utilizando como lenguaje base, el lenguaje Ruby.
- Aparece el test **Acid2** con el que se intenta verificar el cumplimiento de CSS2 por parte de los navegadores. **Safari 2** es el primer navegador que lo pasa al 100%.
- Aparece **Google Maps** con mapas de Estados Unidos y Canadá que, poco a poco, incluirán información de todo el planeta y le convertirán en el buscador de rutas más popular.
- Aparece **Youtube** el servicio más popular para mostrar vídeos de todo tipo, en poco tiempo recibe cientos de millones de visitas. Actualmente es el sitio más popular para ver vídeos a través de Internet.

■ 2006

- **Joe Hewitt** lanza **Firebug** (bajo la idea de la **Web Developer Bar**), la extensión para Firefox que permite depurar el código de cualquier sitio web. Su forma de trabajar es copiada, en los años siguientes, por todas las herramientas para desarrolladores web en el resto de navegadores.
- **John Resig** crea **jQuery**, framework que facilita el uso de **JavaScript** y le dota de mayores capacidades. Es una de las tecnologías más influyentes de estos últimos años.
- Tras muchos años sin sacar una nueva versión, y tras el éxito de **Mozilla Firefox**, Microsoft lanza **Internet Explorer 7**.
- Aparece el sitio de microblogging, **twitter**. En poco tiempo se convierte en uno de los más populares por su velocidad para transmitir información.
- **Google** compra el popular sitio de alojamiento de vídeos **Youtube**.

■ 2007

- Google lanza **Street View** como complemento a Google Maps desde el que se ven fotografías de todas las calles de varias ciudades estadounidenses. En los años siguientes, conseguirá fotografiar la mayoría de las carreteras y calles de diversos países, entre ellos España.
- Se crea la red social **Tumblr**.
- Se crea **SoundCloud**, plataforma de distribución de música online para nuevos artistas.

- Apple comercializa el primer **iPhone**. Comienza el éxito de los smartphones que pasarán a competir con los ordenadores personales como los dispositivos habituales para utilizar aplicaciones web.
- Google presenta el sistema **Android**. En poco tiempo estará presente en numerosos smartphones.

- 2008
  - La guerra de los navegadores se recrudece con la llegada de **Google Chrome**. En solo 5 años se convierte en el navegador más popular. Chrome aparece con el motor **V8** que interpreta JavaScript a gran velocidad.
  - Aparece el primer borrador de HTML 5.
  - Aparece el test **Acid3** con el que se verifica el cumplimiento de los estándares **CSS 2.1**, **DOM** y **JavaScript** estándar (**ECMAScript**). Los primeros en pasarle al 100% son los navegadores **webkit** (**Safari** y **Chrome**) y **Opera**.
  - Aparece **Spotify**, un servicio para la reproducción de música en streaming que consigue acuerdos con la mayoría de discográficas para tener un enorme catálogo.
  - Se crea **Dropbox**, servicio de sincronización de archivos en la nube
  - El modelo de computación en la nube (***cloud computing***) comienza a imperar.
  - La empresa **Nitobi** (comprada por Adobe) desarrolla **PhoneGap** basado en el código abierto **Apache Cordova**. Permite realizar aplicaciones para móviles utilizando los lenguajes habituales para hacer aplicaciones web: HTML, CSS y JavaScript.

- 2009
  - **Satoshi Nakamoto** crea la criptodivisa digital **Bitcoin** que pasa a ser considerada una moneda virtual. Es capaz de cotizar en los mercados y ser utilizada como moneda independiente para realizar transacciones financieras en Internet.
  - Se funda el sitio de ***crowdfunding*** **Kickstarter**, que permite que proyectos presentados por usuarios cuenten con la financiación obtenida de otros usuarios.
  - **Ryan Dahl** crea **Node.js**, un software servidor de JavaScript basado en el motor V8 del navegador Google Chrome. Permite crear aplicaciones en JavaScript fuera del navegador, dotando a este lenguaje de muchas más capacidades. Permite incorporar numerosas extensiones que, entre otras muchas cosas, permiten utilizar Node.js como servidor web. Su éxito ha permitido a JavaScript disfrutar de una edad de oro en la actualidad.
  - Aparece **MongoDB**, sistema gestor de bases de datos **NoSQL** de código abierto. En poco tiempo se convierte en el sistema NoSQL de base de datos más popular. Las bases de datos NoSQL comienzan su influencia, superando a las clásicas bases de datos relacionales cuando los sitios web requieren una enorme cantidad de transacciones por minuto.
  - Google lanza **Angular.js**, framework para crear aplicaciones web utilizando un patrón MVC. La combinación **MongoDB** (gestor de bases de datos) junto a Node.js y

Express.js como servidor de aplicaciones junto a Angular.js se conoce como **MEAN** y en los siguientes años se convertirá en una de las tecnologías dominantes para crear aplicaciones web.

- 2010
  - **Explosión móvil**. El crecimiento desorbitado del uso de **smartphones** hace que cada vez más gente acceda a la web desde dispositivos móviles.
  - Apple lanza el **iPad**, con él comienzan a proliferar las tabletas, compitiendo con los ordenadores portátiles tradicionales.
  - China tiene 460 millones de usuarios en Internet. Es el país con mayor número de usuarios desbancando por primera vez a Estados Unidos.
  - Myanmar se queda sin acceso a Internet debido a un ataque de denegación de servicio (**DoS**).
  - **Netflix** empieza a dominar el mercado del vídeo en streaming a través de Internet. Actualmente tiene cerca de 40 millones de usuarios suscritos en todo el mundo.
- 2011
  - La W3C presenta la norma **CSS 2.1**, que arregla fallos de CSS2. Es insuficiente por lo aceptado que está ya el no oficial **CSS3**.
  - **HTML 5** comienza a adoptarse por la mayoría de desarrolladores web y Flash empieza a dejar de utilizarse (aunque sigue siendo muy utilizado). La **W3C** también acepta **HTML5** y acuerda con la **WHATWG** el futuro estándar.
  - En China ya se navega más desde dispositivos móviles que desde ordenadores de sobremesa o portátiles.
  - Las revueltas de la primavera árabe hacen un uso intenso de **twitter** y **facebook** como medio de comunicación, expresión y convocación ciudadana.
- 2012
  - Los usuarios de Internet consiguen paralizar las leyes antipiratería en EEUU (conocidas como **SOPA** y **PIPA**).
  - Microsoft lanza **Windows 8** con la pretensión de que sea el sistema imperante en ordenadores de **escritorio, tabletas** y **móviles** al permitir usar todos estos aparatos con el mismo sistema.
- 2013
  - **Yahoo!** compra la red social **Tumblr** que es la que más estaba creciendo en ese momento, siendo la red social con mayor número de blogs alojados.
  - Mozilla lanza el sistema operativo móvil **FirefoxOS** orientado a aplicaciones en la nube y al código abierto.
  - Google lanza las **Google Glasses** gafas permanentemente conectadas. Con este y otros pequeños aparatos conectados, se dice que comienza la **Internet of Things** (el

Internet de las cosas) la revolución digital siguiente a la de los dispositivos móviles y que permite que todo tipo de aparatos electrónicos se conecten a Internet.

- Las tecnologías relacionadas con el **Big Data**, consistentes en el análisis de enormes volúmenes de información comienzan a proliferar.

- 2014
  - **Facebook** compra **Whassap**, el servicio de mensajería entre móviles más popular.
  - **Apple** presenta el reloj **Apple Watch** con lo que entra también en la era de los ***wearables*** (dispositivos inteligentes que se ponen como un accesorio de moda).
  - **Ali baba** la tienda on line más fuerte de China bate el record de venta de acciones en Wall Street; las aplicaciones chinas compiten con cada vez más fuerza.
  - La red empresarial de Sony es atacada y penetrada. La empresa se ve obligada a retrasar el estreno de la película ***The Interview*** (***La Entrevista***) debido al supuesto chantaje de *hackers* norcoreanos. Las cuestiones sobre la seguridad y la libertad en las redes son sometidas a debate global. Queda patente que las tecnologías encaminadas a la protección son críticas en el Internet actual.

1.1.3 MOMENTO ACTUAL

En la actualidad la proliferación de dispositivos móviles está propiciando un gran cambio en la forma de crear aplicaciones web. Cualquier aplicación creada para la web tiene que pensar que la audiencia mayoritaria será la de los smartphones y las tabletas. Eso implica que la mayoría de aplicaciones web tienen que cambiar su apariencia externa para adaptarse a la nueva realidad.

La adaptabilidad (conocida con el término ***Responsive Web***) de las páginas web ha pasado a ser una cuestión crucial, ya que el dispositivo con el que los usuarios acceden a Internet cada vez es más variado.

Tecnológicamente en la creación de aplicaciones triunfan los patrones **MVC** y los lenguajes sencillos (**Ruby**, **Python**, **JavaScript**...) En el lado del cliente, **HTML 5** está eliminando a **Flash** como tecnología mayoritaria para crear aplicaciones visualmente atractivas y dinámicas.

Se espera un cambio todavía más espectacular debido a la aparición de pequeños dispositivos conectados (gafas, relojes, pulseras, sensores...), lo que se conoce como el **Internet de las Cosas**, que pese a su tamaño, envían ingentes cantidades de información a Internet. Por ello, las florecientes tecnologías encaminadas al análisis de enormes cantidades de datos (**Big Data**) aún tendrán más importancia. La adaptabilidad de las aplicaciones llega al extremo en los nuevos dispositivos de acceso a la web cuya pantalla puede ser incluso redonda.

La **virtualización y** la computación en la nube (***cloud computing***) hace que el hardware deje de ser tan importante y que todo se delegue en servidores en Internet que virtualizan servicios que, tradicionalmente, se instalaban en el ordenador del usuario (almacenaje de archivos, agendas, contactos, entornos de trabajo, el propio código de las aplicaciones,...) Las aplicaciones de todo tipo se crean exclusivamente para que el usuario acceda a servicios disponibles a través de la web.

El usuario ha pasado a ser el centro absoluto de las aplicaciones, dejando de ser un elemento pasivo para ser enormemente activo; hasta el punto de ser el principal creador de contenido.

Finalmente, el cúmulo de posibilidades para los desarrolladores hace que la forma de crear aplicaciones nunca haya sido tan variada, con tanta cantidad de plataformas y herramientas para producirlas. El ritmo de aparición de nuevas herramientas, lenguajes, marcos de trabajo y componentes para la depuración y el testado de aplicaciones, es, simplemente, abrumador.

Lejos de disminuir su influencia, las aplicación y servicios en Internet siguen aumentando cada día. Pero sobre todo, siguen aumentando en calidad y prestaciones.

ACTIVIDAD 1.1: TENDENCIAS EN INTERNET

- La dirección http://www.kpcb.com/internet-trends da acceso al informe de tendencias en Internet de la empresa **KPCB**, una de las fuentes más reputadas en este aspecto.
- El informe está en inglés, su lectura es casi obligada ya que es un análisis exhaustivo y muy bien documentado. Se renueva cada año

1.2 APLICACIONES WEB

1.2.1 ¿QUÉ ES UNA APLICACIÓN WEB?

En informática, una aplicación es un software que permite al usuario realizar una determinada tarea o servicio. Las aplicaciones clásicas se crean en lenguajes como C o C++ que ponen a disposición del programador todas las capacidades de la computadora.

Una aplicación web, simplemente, es una aplicación creada para ser ejecutada por un navegador. El hardware queda oculto al programador, que solo ve la capa de trabajo que le cede el navegador.

Hoy en día las aplicaciones web son las aplicaciones más populares porque todos los usuarios actuales poseen y manejan navegadores (***Google Chrome***, ***Internet Explorer***, ***Mozilla Firefox***...) para recorrer Internet y están muy acostumbrados a trabajar con ellos.

Una aplicación web, en principio, es menos potente que una aplicación clásica (a las aplicaciones clásicas se las llama ahora **aplicación de escritorio**) al no poder optimizar su código para la máquina en la que se está ejecutando. Las aplicaciones de escritorio pueden acceder a todo el hardware de la computadora y, por ello, optimizar su rapidez y prestaciones. Las aplicaciones web dependen de la habilidad del navegador para manejar el hardware de la máquina en la que reside.

Las aplicaciones web se crean en HTML y sus tecnologías asociadas (CSS, JavaScript...). Aunque los navegadores no tienen capacidad para compilar código de lenguajes clásicos (C, C++, Pascal...) sí contienen intérpretes muy potentes para lenguajes incrustados en las aplicaciones web (especialmente JavaScript), como es el caso del motor V8 del navegador Chrome.

1.2.2 VENTAJAS DE LAS APLICACIONES WEB

Si las aplicaciones web han triunfado sobre las tradicionales aplicaciones de escritorio es por sus ventajas, las cuales se enumeran a continuación:

- **Gran compatibilidad**. Funcionan en todo tipo de sistemas, tanto de escritorio como sistemas móviles. Lo único que requieren es la presencia de un navegador, requisito fácil de cumplir ya que los navegadores se incluyen en cualquier dispositivo, sea del tipo que sea.

  En las aplicaciones de escritorio existe el problema de que, por ejemplo, una aplicación creada para Windows, normalmente no funcionará en un entorno Linux. Aunque hay soluciones de virtualización de aplicaciones para solucionar este problema, lo cierto es que siempre es más fácil utilizar aplicaciones web.

- **Requerimientos mínimos en el cliente**. El único requisito de las aplicaciones web es el navegador, las aplicaciones de escritorio exigen un Sistema Operativo concreto, una determinada cantidad de memoria RAM, instalación de ciertas plataformas, paquetes o servicios... Es verdad que a veces las aplicaciones web exigen determinados navegadores, o bien ciertos plugins en ellos, pero disponiendo de un navegador de última generación (teniendo en cuenta que todos son gratuitos y que los smartphones ya los incorporan) una aplicación web bien hecha no debería requerir nada más.

- **Fáciles de manejar por los usuarios**. El entorno de trabajo (es decir, el navegador) es conocido por los usuarios, por lo que les es más fácil manejar una aplicación web, ya que cualquier usuario actual está acostumbrado a recorrer páginas web con su navegador.

- **Facilidad de mantenimiento**. Quizá es el aspecto más importante. Cuando una aplicación de escritorio se renueva, el usuario necesita actualizarla en su equipo. Todos los usuarios de Windows estamos acostumbrados a las periódicas actualizaciones del sistema que resultan enormemente pesadas, por el tiempo de espera que suponen.

  Las aplicaciones web también se actualizan, pero solo en el servidor en el que se alojan. En cuanto se actualiza allí, todos los usuarios verán la última versión la próxima vez que accedan a ella, sin tener que instalar nada en su máquina.

  Por otro lado, tendremos la seguridad de saber qué versión de la aplicación poseen los usuarios. En las aplicaciones de escritorio dependemos de si el usuario instala o no la última versión.

  En las aplicaciones clásicas, si hay fallos, por ejemplo, de seguridad que se detectan en una versión, el correspondiente parche a aplicar depende de que los usuarios lo instalen; fácilmente habría versiones inseguras de la aplicación, ya que no todos los usuarios habrían instalado ese parche. En una aplicación web, el parche se instalaría e instantáneamente todos los usuarios accederían a la versión segura de la aplicación.

- **Datos centralizados**. Los datos que maneja una aplicación web se encuentran en un único almacén o base de datos. Lo que facilita su mantenimiento y administración. El usuario además se despreocupa de estas tareas.

- **No hay instalación.** El proceso de instalar algunas aplicaciones de escritorio es, a veces, largo, tedioso y difícil. Además de postergar la experiencia de usuario, en el sentido de que sin instalación, no podremos disfrutar de la aplicación. Una aplicación web no requiere instalación, es simplemente un servicio accesible desde Internet a través del navegador.

- **Costes reducidos en su implantación.** No hay material de instalación, ni impreso, ni cajas de producto. Todo lo relacionado con la aplicación es digital. Hoy en día las propias aplicaciones de escritorio utilizan en gran medida esta idea al poderse comprar y descargar desde Internet.

- **Accesibles desde diferentes máquinas y ubicaciones.** De cara al usuario es la gran ventaja. Las aplicaciones de escritorio cuentan con una gran tara, el hecho de que si cambio de máquina tengo que instalar en ella la aplicación de escritorio, lo que implica además tener permisos de usuario para realizar esta labor, además de que las licencias de uso suelen estar restringidas a un número concreto de máquinas. Cada vez es más habitual que los usuarios utilicen máquinas diversas para trabajar: la del puesto de trabajo, el portátil personal, su tableta, smartphone...

  Las aplicaciones web son accesibles desde cualquier computadora y desde cualquier parte del mundo. La idea es muy diferente: si es una aplicación de pago, se paga por el uso (pagará cada usuario) y no por el número de máquinas desde las que se accede a ellas.

- **Se permite el uso de *Thin Clients*.** Los Thin Clients son los llamados clientes ligeros, software y hardware de baja potencia. Las aplicaciones web son capaces de funcionar incluso en ordenadores ya obsoletos, porque solo requieren un navegador (ejemplo claro de Thin Client) y eso permite seguir aprovechando hardware que, de otro modo, no valdría para el trabajo.

1.2.3 DESVENTAJAS DE LAS APLICACIONES WEB

- **Menos potentes.** Visto el apartado anterior, cualquier persona se podría plantear la pregunta ¿para qué sirven las aplicaciones de escritorio? La cuestión es que no todas las tareas que se pueden realizar desde una aplicación web; o al menos, no con la misma eficiencia y velocidad.

  Por ejemplo, este manual está creado con la aplicación ***Adobe InDesign,*** que es un software de maquetación de documentos. De este software se requiere que manipule, coloque y edite con rapidez, textos, imágenes, formas... Para ello exprime la potencia del ordenador en el que se instala: es más, una aplicación de este tipo no se puede instalar en cualquier ordenador, tiene requisitos muy altos.

  Lo mismo ocurre con un juego de última generación, que requerirá renderizar escenarios y música de gran calidad a tiempo real; esto no está al alcance (al menos por ahora) de los navegadores.

  Sin ir tan lejos, es fácil percibir que es más grata la experiencia de escribir un documento en un procesador de textos instalado en el ordenador (como ***Microsoft Word***, por ejem-

plo) que hacer la misma tarea en un procesador de textos online (como *Office 365* por ejemplo).

- **Aprovechan peor el hardware**. Tiene clara relación con la desventaja anterior. El hecho de tener un mejor o peor ordenador no influye apenas en el rendimiento de una aplicación web; evidentemente, esto es una ventaja para los usuarios de ordenadores menos potentes, pero significa que no se está aprovechando debidamente el rendimiento que pueden proporcionar los ordenadores más veloces y se infrautiliza la inversión que el usuario hizo en la compra de su máquina.

- **Se requiere conectividad**. Es decir, debemos estar conectados a Internet de forma ininterrumpida para utilizar la aplicación web. Hoy en día no parece un requisito excesivo, pero lo cierto es que no siempre funciona una conexión a Internet y este hecho podría paralizar la realización de una tarea importante. Esta es la razón por la que la conectividad de las empresas a Internet se ha convertido en una cuestión tan crítica, requiriendo de enlaces redundantes dentro de su red interna, así como de la posibilidad de utilizar diversas (y también redundantes) conexiones a Internet.

- **Las aplicaciones web son más difíciles de crear**. No es difícil crear una simple página web con HTML y CSS; pero una aplicación de escritorio en general es más fácil de crear que una aplicación web, por el hecho de que en una aplicación web hay muchos más elementos dinámicos. La depuración es compleja en el caso de las aplicaciones web ya que debemos esperar respuestas a peticiones del servidor, las cuales se consiguen tras realizar numerosos procesos en diversos lenguajes.

  Se agrava esta circunstancia por el hecho de que crece las exigencias de los usuarios hacia las aplicaciones web, lo que fuerza al desarrollador de la aplicación a incorporar cada vez más elementos dificultando el mantenimiento.

  No obstante también aparecen continuamente nuevas herramientas que facilitan este trabajo, disponiendo actualmente de una miríada de herramientas para producir, mantener, codificar, testar, simular, etc.

- **Delegación del control de nuestra información**. Probablemente sea la cuestión más ignorada por los usuarios de las aplicaciones web, y sin embargo, la que tiene implicaciones más importantes. Cuando una persona maneja una aplicación de escritorio para escribir un documento, este (por lo general) se almacena en nuestra máquina y la cuestión sobre dónde lo almacenamos, hacer copias de seguridad del mismo, quién tiene acceso... recaen sobre nosotros mismos, es nuestra responsabilidad.

  En una aplicación web, normalmente, los datos se almacenan en Internet y los gestiona la empresa creadora de la aplicación. Eso supone que no necesitamos preocuparnos por las cuestiones anteriores sobre nuestros datos; pero, a la vez, significa que dependemos de las buenas prácticas que la empresa propietaria de la aplicación realice sobre nuestros datos.

  El hecho de que, además, estos estén centralizados, hace que una persona que obtenga acceso a los mismos de forma indebida, tenga la posibilidad de manejar información

confidencial de miles o millones de usuarios. A lo largo de la historia de Internet es una situación que, desgraciadamente, ha ocurrido varias veces y que hay que tener muy en cuenta.

Realmente no solo las aplicaciones web provocan delegar el uso de nuestra información a terceros, es muy habitual usar aplicaciones de escritorio y que los datos se almacenen en lo que llamamos ***la nube***, que no es más que un servicio en Internet que centraliza el almacenamiento de los datos y por lo tanto, adolecerá de estas mismas ventajas e inconvenientes.

1.3 LA WEB 1.0, LA WEB 2.0 Y LA WEB 3.0

Las primeras aplicaciones web eran poco más que un conjunto de páginas que servían información estática a los usuarios sin, prácticamente, interaccionar con ellos. Esta primera web, constaba de páginas que eran simplemente un conjunto de textos e imágenes, junto con los hipervínculos que permitían saltar hacia otros contenidos.

Poco a poco, se pretendió integrar contenidos más variados: animaciones, vídeo, sonido, juegos, aplicaciones ofimáticas... Este tipo de páginas empezaron a proliferar tras el estallido de la burbuja punto com (ocurrida en el año 2001 debido a que muchos de los servicios en Internet eran especulativos) y es en el año 2004 cuando **Jeff Bezos,** de la empresa **O'Reilly Media**, utiliza el termino **Web 2.0** en una conferencia (incluso llega a registrarlo) y lo hace popular. Hoy en día la mayoría de las aplicaciones web pertenecen a esta web 2.0, o bien a la nueva web 3.0.

1.3.1 LA WEB 2.0

Podemos decir que hay tres pilares que conforman las páginas web 2.0: **Aplicaciones Ricas de Internet**, **SOA** y **Web social**. Se desglosan a continuación estos conceptos.

1.3.1.1 RIA: APLICACIONES RICAS DE INTERNET.

Se llaman Aplicaciones Ricas de Internet (también conocidas con las siglas **RIA** de ***Rich Internet Applications***) a aquellas aplicaciones web que ofrecen resultados que las hacen casi indistinguibles respecto a las aplicaciones de escritorio.

Las aplicaciones web iniciales no permitían apenas interacción usuario/máquina. No era posible siquiera en pensar en crear servicios como los actuales **Google Maps**, **Facebook** u **Office 365**, por nombrar algunos.

Sin embargo, los navegadores han mejorado sus prestaciones hasta el punto (como se ha comentado anteriormente) de ser un verdadero software compilador de aplicaciones, capaces de traducir lenguajes (como **JavaScript**) que permiten la interactividad, así como incorporar medios mucho más atractivos. Esto ha permitido que las aplicaciones web compitan con las aplicaciones de escritorio en experiencia de usuario.

Así hay aplicaciones web con unas capacidades sorprendentes, como es el caso de **Google Docs** que desde hace años es un competidor del paquete **Office** de **Microsoft** (la propia empresa Microsoft consciente del éxito de estas aplicaciones ***on line***, posee su propio software de ofi-

mática en línea, **Office 365**) y otras herramientas empiezan a sustituir a muchas aplicaciones de escritorio, así como a aplicaciones que permiten el uso de servicios espectaculares que jamás han existido en la informática de escritorio como **Google Maps**.

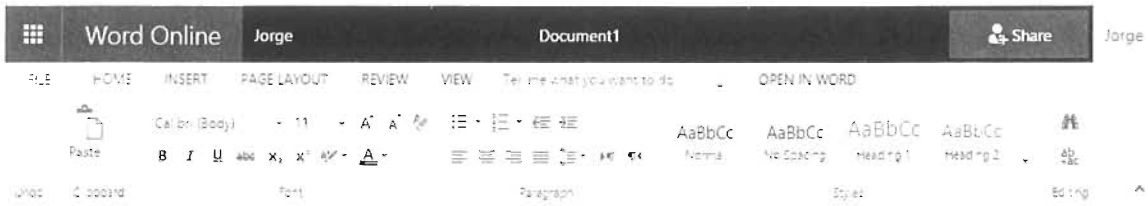


Figura 1.1: Ejemplo de creación de un nuevo documento de trabajo con la Aplicación Web de Microsoft, *Office 365*. Casi es indistinguible respecto de la aplicación *Office* de escritorio.

Por ello realmente la creación de aplicaciones web está dejando atrás la creación de aplicaciones de escritorio, ya que los usuarios cada vez perciben menos ventajas en instalar una aplicación en su ordenador (aunque no hay que olvidar sus inconvenientes).

1.3.1.2 SOA: ARQUITECTURA ORIENTADA AL SERVICIO

Otra de las claves de las aplicaciones web 2.0 es el paradigma conocido como SOA, que aglutina el conjunto de tecnologías y técnicas que permiten diseñar aplicaciones como un conjunto de servicios que resuelven peticiones de los usuarios. De esta forma, se puede crear pequeños elementos software reutilizables de forma independiente respecto al lenguaje con el que fueron creados.

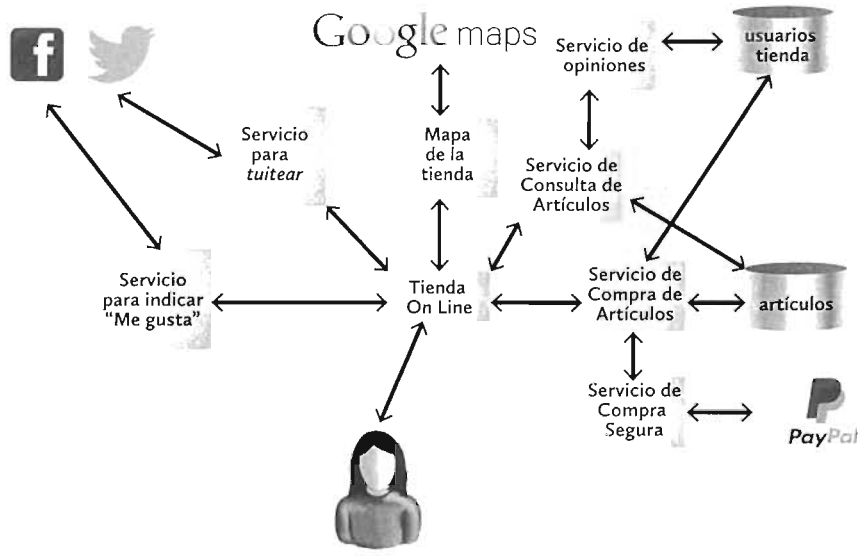


Figura 1.2: Ejemplo de arquitectura orientada al servicio

SOA es una arquitectura donde cualquier componente software se diseña para ser utilizado como un servicio en la red. Un servicio en la red es un simple componente que recibe peticiones y entrega los resultados correspondientes a la misma. Es la forma habitual de trabajar en un entorno en red, pero ahora ligado a resolver una funcionalidad de alto nivel. El lenguaje o tecnología concreta en el que se haya programado el servicio, es irrelevante. Un servicio se puede usar de la misma manera independientemente de como fue creado.

Así, una aplicación web se puede modelar como si fuera un conjunto de piezas que se comunican entre sí. Esa orientación al servicio ha supuesto una auténtica revolución en Internet, ya que si creamos un servicio, por ejemplo, que nos permita conocer las opiniones de los usuarios sobre los artículos de una tienda, este servicio puede ser reutilizado, tanto por el de venta de artículos de la propia tienda, como por otro, incluso, fabricado por otra empresa que muestre información al consumidor sobre artículos en general, de nuestra tienda u otras.

Un caso muy evidente son las numerosas páginas en Internet de empresas y entidades que muestran en un recuadro la dirección de la empresa utilizando los servicios de Google Maps.

Esto ha dado lugar a un nuevo modelo de programación que produce el llamado **Software as a Service** (***software como servicio*** o **SaaS**) y que varía absolutamente la forma clásica de crear aplicaciones. La idea final es que las aplicaciones no están pensadas para ser instaladas en el ordenador del cliente (como ocurría y ocurre en la programación tradicional de aplicaciones), sino que se coloca en un servidor en Internet al que acceden los clientes. De este modo, la aplicación se convierte en un servicio que está disponible desde cualquier punto del planeta y que puede ser invocado por otras aplicaciones.

1.3.1.3 WEB SOCIAL

Es la parte más evidente y entendible de la web 2.0. El término se refiere a que el usuario pasa a ser el centro de las aplicaciones web 2.0. El usuario conecta con otros usuarios a través de la aplicación, participa de los contenidos creándoles, como ocurre en **Twitter**, **Facebook** o **Instagram**, u opinando sobre ellos, como ocurre hoy en día en las noticias de cualquier periódico en Internet.

El usuario ha dejado de ser pasivo para ser enormemente activo, los contenidos ya no pertenecen solo a la entidad propietario del sitio web, sino a los propios usuarios.

Eso hace que el contenido de una aplicación escape al control de la entidad creadora de la misma, las aplicaciones tratan de traer a usuarios entusiastas y activos que publiciten el producto de la empresa (lo cual sale gratis a la misma) y para ello la aplicación debe proporcionar mecanismos. Además pasa a ser imprescindible la conexión de las aplicaciones con los principales servicios de red social en Internet.

1.3.2 LA WEB 3.0

Desde aproximadamente el año 2010, muchos analistas consideran que estamos en la versión 3.0 de la web. Otros afirman que simplemente en estos años los pilares de la web 2.0 se han hecho más extremos.

Lo que es innegable es que la nueva guerra de navegadores entre **Internet Explorer**, **Mozilla Firefox** y **Google Chrome** principalmente (que parece ganar Google Chrome) ha dotado de una mayor potencia al lado del cliente, lo que ha revertido en un espectacular crecimiento de las prestaciones disponibles para los programadores front-end.

Esto, unido al hecho de que los usuarios son los autores de la mayoría de los contenidos disponibles en la web (en la web 1.0, unas pocas personas creaban contenidos y la mayoría eran simples *visualizadoras* de ese contenido) y a la brutal competencia por el número ingente de páginas web en Internet, ha provocado el crecimiento los llamados pilares de la web 3.0:

- **Web inteligente**. Concepto que se refiere a la inteligencia artificial aplicada a la web; es decir, se refiere a que la propia web reaccione de forma inteligente ante los usuarios. Un ejemplo de web inteligente es una aplicación web que nos muestra artículos de ropa de montaña porque ha detectado que nos gusta ir a la montaña, en lugar de mostrarnos publicidad sobre artículos que no nos interesan. La web inteligente utiliza estas áreas de la ciencia de la computación:
  - **Procesadores de lenguaje natural** (NLP). Que hacen que las aplicaciones web se comuniquen en un lenguaje semejante al que emplearía cualquier persona.
  - **Aprendizaje automático** (***Machine Learning***). Ciencia que trabaja en que las computadoras sean capaces de aprender por sí mismas nuevos conceptos y aplicaciones de los mismos.
  - **Sistemas de recomendación**. Se trata de la tecnología que nos permite recomendar productos a un usuario en base a sus elecciones de productos anteriores y a otros datos que poseemos sobre él. Es un sistema predictivo muy interesante para las empresas de venta de productos on line.
- **Web semántica**. Se trata de que los contenidos en la web dispongan de etiquetado o meta datos que permitan darles significado. Cuanto mejor es ese significado, mejor podremos analizar esos datos. El lenguaje **XML** y especialmente los sublenguajes **RDF** y **OWL** son los más empleados para dar este sentido semántico a los datos de las aplicaciones web. Para muchos analistas ***web 3.0*** y ***web semántica*** son términos sinónimos.

1.4 FUNCIONAMIENTO DE UNA APLICACIÓN WEB

A la hora de crear aplicaciones web se pueden utilizar diversos lenguajes y tecnologías. Hay dos estrategias:

- **Lenguajes y tecnologías en el lado del cliente**. Se trata de los elementos que se incorporan junto al código HTML de una aplicación web y que necesitan ser interpretados en el navegador del usuario.
- **Lenguajes y tecnologías en el lado del servidor**. En este caso se trata de aplicaciones creadas con lenguajes y elementos que se interpretan en el servidor que aloja la aplicación.

1.4.1 FUNCIONAMIENTO EN EL LADO DEL CLIENTE

En modo cliente, la página entregada por el servidor web que la alberga, contiene (además del código HTML) elementos pertenecientes a otros lenguajes y tecnologías. Las principales tecnologías son:

- **CSS.** El lenguaje de las hojas de estilos. Permite modificar la apariencia de una página web. Está en continua mejora, ayudando a conseguir resultados cada vez más espectaculares.
- **JavaScript.** Lenguaje de programación sencillo pero potente, cuyo código se puede incluir directamente dentro de una página web. Su éxito está haciendo que sea un lenguaje imprescindible para los desarrolladores. Aporta interactividad a las páginas y un número en constante crecimiento de utilidades. Además, para este lenguaje se han creado en los últimos años una cantidad innumerable de herramientas y marcos de trabajo, como **jQuery** (sin duda el más exitoso), **MooTools**, **Modernizr** o **Knockout**.

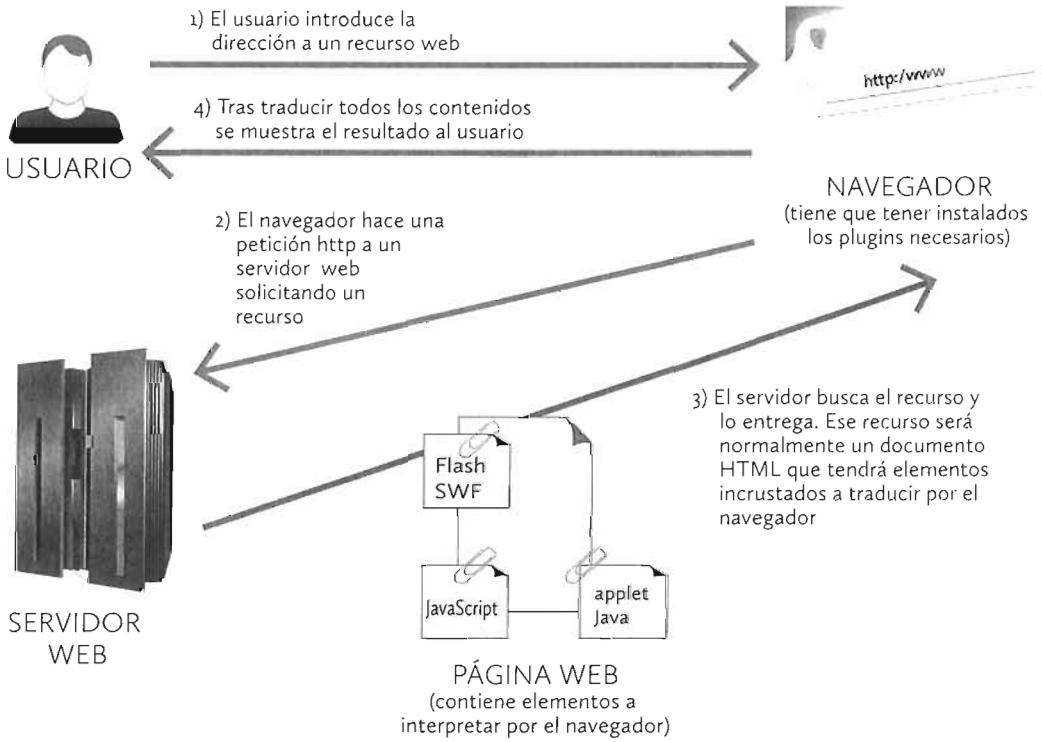


Figura 1.3: Funcionamiento de las aplicaciones web en el lado del cliente

- **Flash.** Componente que se suele incrustar mediante una etiqueta HTML de tipo **object** que hace referencia a un archivo de tipo SWF que es el que contiene el código Flash. El navegador requiere un plugin para poder mostrar el contenido del archivo SWF. Los componentes de tipo Flash se crean con el programa homónimo de la empresa **Adobe**.

- **Silverlight.** Componente, semejante en comportamiento al anterior, que requiere también un plugin. Silverlight es propiedad de **Microsoft** y el contenido de este tipo se crea a través de software especial.
- **Applets de Java.** Se trata de un componente que se incrusta mediante una etiqueta HTML de nombre, precisamente, **applet** y que hace referencia a un archivo que contiene código Java. Requiere de un plugins llamado **máquina virtual de Java (JVM)** en el navegador.

En general, las aplicaciones que utilizan tecnologías del lado del cliente requieren que el navegador del usuario tenga instalados los ***plugins*** que le permitan interpretar dichas tecnologías y que aporte la potencia necesaria para ejecutar esos componentes de forma eficiente. Así, un navegador se convierte en una especie de máquina virtual capaz de procesar aplicaciones creadas en todo tipo de lenguajes y tecnologías.

Todo el esfuerzo, usando esta técnica, lo realiza el navegador. De modo que se habla de **tecnología web de cliente pesado** para referirse a una aplicación web que hace un uso intensivo de componentes del lado del cliente.

Esta forma de trabajo parecía que se iba quedando obsoleta, por la dificultad de tener un navegador con todos los componentes requeridos por las distintas aplicaciones. Además se obliga al usuario, en muchas ocasiones, a descargar los componentes y plugins que faltan para poder utilizar adecuadamente una aplicación web.

Esas descargas ,en muchos casos, son muy pesadas. Por ejemplo, el plugin para poder ver ***applets*** en lenguaje Java ocupa en torno a 12 MB. Además, los usuarios no entienden el por qué de estas descargas que resultan ser muy incómodas para ellos. Tanto que, en muchas ocasiones, las acababan cancelando y abandonando la aplicación que las requiere.

En la actualidad (gracias al triunfo de HTML5) están en desuso las tecnologías avanzadas del lado del cliente (como **Flash**), pero sí triunfan las que pertenecen al nuevo estándar HTML5. En especial JavaScript que se usa intensivamente y que, cada vez, es más potente. Lo que ha obligado a los navegadores a ser un auténtico software de proceso de aplicaciones avanzadas en ese lenguaje. Para el usuario es un proceso transparente, ya que funcionan simplemente con tener el navegador al día. Tener la última versión al día del navegador es, ciertamente, más entendible para el usuario que descargar un plugin.

Hoy en día todos los desarrolladores dan por hecho que los usuarios tienen navegadores con capacidad de traducir CSS y JavaScript, pero sin embargo no dan por hecho que tengan instalado ningún otro plugin para las otras tecnologías del lado del cliente.

1.4.2 FUNCIONAMIENTO EN EL LADO DEL SERVIDOR

En este otro paradigma, se crean páginas que contienen componentes realizados en lenguajes y tecnologías que deben de ser interpretadas por un servidor en Internet en lugar de en el navegador del usuario.

Cuando un usuario o usuaria hace una petición a un recurso web, el servidor que contiene dicho recurso se da cuenta de que contiene elementos a interpretar en el lado del servidor y pide al **servidor de aplicaciones** adecuado que traduzca esos elementos antes de enviar el resultado al navegador.

El servidor de aplicaciones, que suele ser realmente un módulo software, enviará el resultado al servidor web en un formato traducible en el lado del cliente; es decir, un documento HTML. El servidor web finalmente enviará al usuario este resultado.

La ventaja de este modo de trabajo es que el navegador puede ser más ligero (se la llama tecnología de **cliente ligero**) y la parte dura o pesada se la lleva el servidor web.

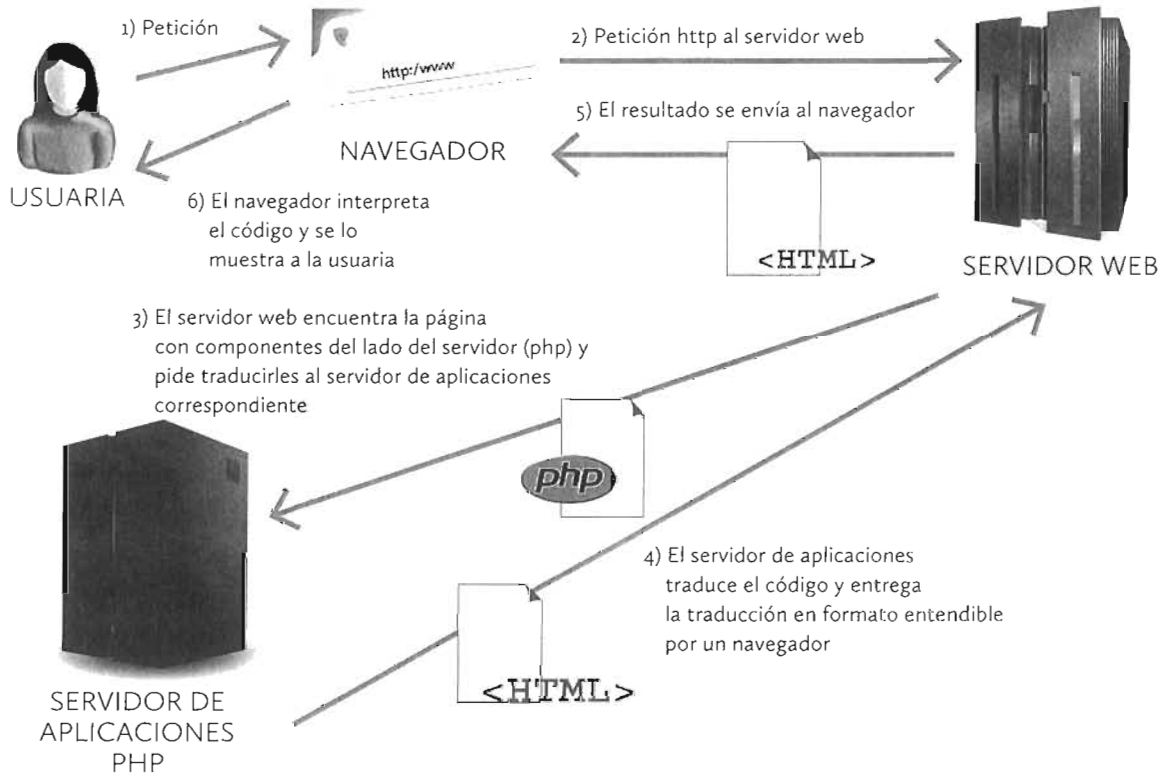


Figura 1.4: Funcionamiento de las aplicaciones creadas para ser interpretadas en el lado del servidor

Hoy en día ambos métodos se mezclan, de modo que los desarrolladores web crean aplicaciones que contienen elementos en el lado del servidor (***back-end***) y en el del cliente (***front-end***). Por ejemplo, se pueden crear páginas PHP que incluyen código JavaScript; lo que implica proceso en el lado del servidor (traducir el PHP) y en el del cliente (traducir JavaScript). La idea es crear aplicaciones web más ricas, funcionales e interactivas.

1.5 CREACIÓN DE APLICACIONES WEB

1.5.1 SERVIDORES WEB

Un servidor web es una máquina o software capaz de interpretar peticiones web realizadas con el protocolo **http o https** y de devolver el resultado de la petición, que suele propiciar la entrega de un recurso alojado en el servidor. A los servidores web se les llama también (con mejor tino probablemente) **servidores http**.

Normalmente es el navegador el que pide al servidor web el recurso que desea el usuario, así como de recibir el mismo, o bien una notificación de error si dicho recurso no existe. La mayoría de peticiones http resultan en la entrega de un documento HTML.

1.5.2 SERVIDORES DE APLICACIONES WEB

Los servidores web solo tienen la capacidad comentada en el punto anterior: resolver peticiones http. Pero no se molestan en descifrar el código de los documentos que entregan. Esa tarea la dejan en manos del cliente que hizo la petición (normalmente un navegador web).

La cuestión es que, como se ha visto en el Apartado 1.4.2 *"Funcionamiento en el lado del servidor"*, en la página 23 un servidor de aplicaciones es el encargado de traducir instrucciones hechas en lenguajes del lado del servidor y entregar el resultado de esa traducción al servidor web que le pidió dicha traducción. Además, el código del lado del servidor puede contener instrucciones que impliquen acceder a otros servidores (como por ejemplo acceso a bases de datos) con lo que el proceso puede ser muy complejo.

Los servidores de aplicaciones trabajan en conjunto con los servidores web para que el proceso se haga de forma transparente al usuario; es decir, el usuario pide el servicio a través, normalmente, de su navegador y el servidor web atiende la petición, y pide al servidor de aplicaciones la traducción de la aplicación contenida, a fin de mostrar al usuario el resultado de forma entendible por su navegador (es decir, en formato HTML). El proceso está detallado en la *Figura 1.4*.

En la práctica, los servidores de aplicaciones son simplemente módulos software que se añaden al servidor web, de modo que los términos servidor web y servidor de aplicaciones web se confunden, pero es importante distinguirles ya que no todos los servidores web se comportan como servidores de aplicaciones web y, sobre todo, porque la labor es muy diferente: una cosa es encargarse de las peticiones http y otra traducir lenguajes de servidor.

Dependiendo de la tecnología de servidor utilizada tendremos diferentes servidores de aplicaciones. Así hay servidores PHP, servidores .NET, servidores Java, etc.

1.5.3 ARQUITECTURA DE TRES NIVELES

Las aplicaciones web actuales utilizan lo que se conoce como arquitectura de tres niveles (en inglés ***three-tier architecture***), a veces incluso se habla de más capas. Estas capas son:

- **Capa de presentación.** Maneja la parte de la aplicación web que ve el usuario. Es decir, se encarga de la forma de presentar la información al usuario. Consta del código del lado del cliente (HTML, JavaScript, CSS, Flash...) que le llega al navegador, aunque ese código haya sido generado originalmente por una tecnología del lado del servidor.

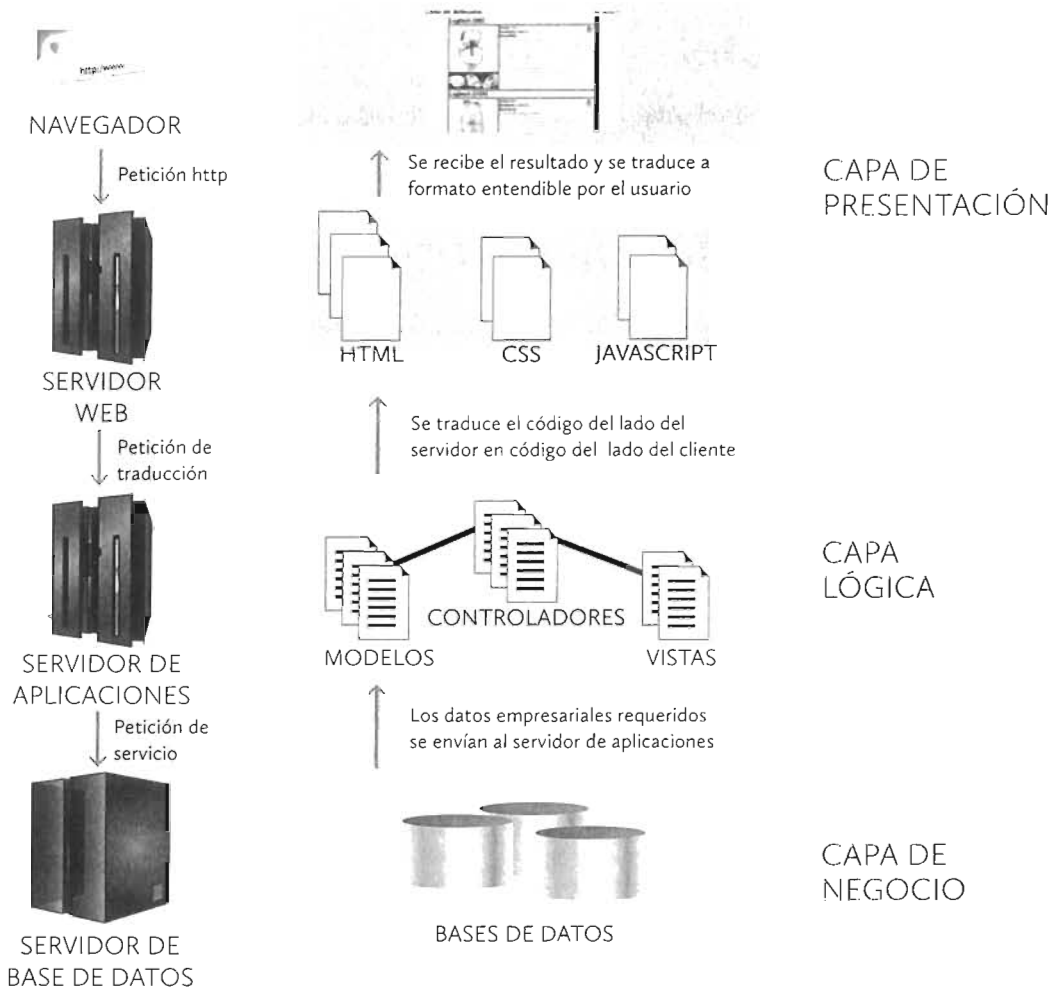


Figura 1.5: Esquema general de funcionamiento de una aplicación web moderna siguiendo un patrón de arquitectura en tres niveles

- **Capa lógica.** Es la encargada de gestionar el funcionamiento de la aplicación. En ella se encuentran los documentos escritos en un lenguaje que se debe de interpretar en el lado del servidor (por ello, esta capa está relacionada con el servidor de aplicaciones) y cuyo resultado se enviará al servidor web para que este, a su vez, lo envíe al cliente que hizo la petición.

  Habitualmente la programación en esta capa divide el código en tres partes: el **modelo**, el **controlador** y la **vista**, utilizando el exitoso **modelo MVC**, del que hablaremos más

adelante. Cada parte se encarga de una acción concreta de la aplicación y así se facilita el mantenimiento de la misma.

- **Capa de negocio**. Es la que contiene la información empresarial. Esta información siempre tiene como requerimiento que quede oculta a cualquier persona sin autorización. En esta capa, fundamentalmente, se encuentra el **sistema gestor de bases de datos** (SGBD) de la empresa, además de otros servidores que proporcionan otros recursos empresariales (como servidores de vídeo, audio, certificados...)

  El servidor de aplicaciones hace peticiones a estos servidores para obtener los recursos de la empresa que se requieren para cumplir la petición http original. De modo que el proceso de acceso a estos recursos queda oculto totalmente al navegador, lo que añade una mayor seguridad al proceso. Los servidores de esta capa entregarán los recursos solicitados, y el servidor de aplicaciones será el encargado de situarle de forma adecuada en el resultado que viajará hasta el navegador del usuario.

Todo este mecanismo de trabajo es el que involucra la creación de aplicaciones web. En general, los servidores web actuales actúan de servidores de aplicaciones, una vez que se les instala el software pertinente. Por ello cuando se habla de servidores web, en realidad también hablamos de servidores de aplicaciones web.

1.5.4 PROGRAMACIÓN *BACK-END* Y PROGRAMACIÓN *FRONT-END*

Existe también una división entre las tareas de las personas encargadas del desarrollo de aplicaciones web en base a lo cerca o lejos que su tarea está respecto al usuario. Así tenemos

- **front-end**. Se refiere a la parte del desarrollo encargado de producir la apariencia final de la aplicación que verá el usuario. En cierto modo es la interfaz de usuario. Las personas que se encargan del front-end de la aplicación son las que diseñan las maquetas o mockups de la aplicación, usando software como, por ejemplo, **Adobe Photoshop** u otras herramientas más especializadas para la producción de ***wireframes*** (diseños de líneas que simulan el aspecto final de la aplicación) y ***mockups*** (maquetas con una mínima funcionalidad).

  También forman parte del front-end las personas encargadas del HTML y CSS de la página, así como del JavaScript. Es decir del funcionamiento de la capa de presentación. Lo que el front-end intenta conseguir es una buena experiencia de usuario.

- **back-end**. Un desarrollador back-end se encarga de realizar la parte de la aplicación que queda oculta al usuario. El funcionamiento de la capa lógica y de negocio sería el back-end, ya que se usan los lenguajes del lado del servidor y los de base de datos. Es la parte de la aplicación encargada de que funcione debidamente la aplicación.

  Los objetivos del back-end son: un acceso rápido a los datos, una comunicación eficiente con el navegador, control de la seguridad, estructurar el código para un fácil mantenimiento, gestión de errores, etc. Es una zona más técnica, pero no más importante, es fundamental que el front-end y el back-end se complementen.

1.5.4.1 MODELO *FULL STACK*

La arquitectura por niveles y la división en back-end y front-end provocan que la programación de aplicaciones web utilice una **pila** (***stack*** en inglés) de lenguajes. De modo que un programador de aplicaciones típico debe programar en JavaScript (además de utilizar HTML y CSS) para programar el front-end; si pasa al back-end debe programar, por ejemplo, en PHP; si se dedica a gestionar la base de datos entonces debe utilizar SQL. Eso hace que al menos sean necesarios tres lenguajes (cuando no más) y por ello, este modelo acaba produciendo especialistas en cada área: programadores back-end, programadores front-end y programadores de base de datos.

La aparición de **Node.js** (servidor JavaScript muy utilizado como servidor de aplicaciones web) junto con la aparición de ciertas bases de datos **NoSQL**[6] (como **MongoDB**) está propiciando un nuevo paradigma en el que el mismo lenguaje, concretamente **JavaScript**, se utiliza en todas las capas. A eso se le conoce como programación **Full Stack,** ya que un solo lenguaje sirve para toda la pila de trabajo.

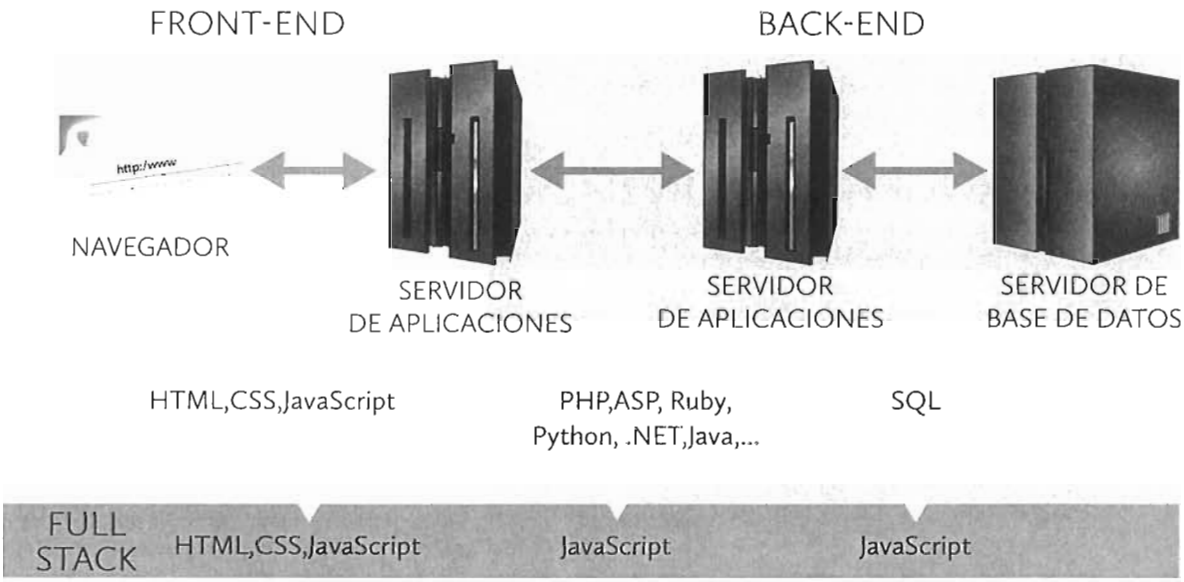


Figura 1.6: Ubicación conceptual del front-end y el back-end. En la parte inferior funcionamiento de un modelo Full Stack típico, por encima ejemplos de pilas de lenguajes en un modelo que no es Full Stack

El modelo ***Full Stack*** está resultando exitoso, porque la especialización de tareas puede aumentar la desmotivación del programador al verse relegado siempre a la misma tarea. El modelo de pila completa permite a los desarrolladores una facilidad de cambio en la capa de desarrollo, ya que el lenguaje es el mismo y por lo tanto, un experto en ese lenguaje podrá cambiar de área o nivel sin demasiados problemas.

Los detractores de este modelo argumentan que los lenguajes especializados, como SQL, están más adaptados a la tarea que realizan y, por lo tanto, la realizan mejor.

6 Las bases de datos NoSQL, se diferencian de las clásicas bases de datos relacionales en que no utilizan SQL como lenguaje de trabajo. De hecho, tampoco contemplan el uso de tablas y relaciones como modelo lógico de datos.

1.5.5 PARADIGMA MVC

Las aplicaciones web a medida que se van mejorando y ampliando, son más difíciles de mantener. Aparte, las cuestiones sobre los distintos niveles hacen que sea necesario separar el código para que podamos organizarlo de forma sostenible en el futuro.

A este respecto, el modelo MVC de creación de aplicaciones (y sus diversas variantes) ha tenido una espectacular aceptación en este campo. Puesto que la capa de trabajo de la aplicación web es la capa lógica, que es donde reside la generación del código final, es en esta capa donde se aplica el paradigma MVC.

Lo que hace esta técnica es separar la programación de la capa lógica en otras tres capas: el modelo, la vista y el controlador.

- **Modelo**. Contiene el código que se encarga de asociar la información que procede la capa de negocio a su formato entendible por el lenguaje y tecnología utilizado para programar la aplicación web.
- **Vista**. Genera la presentación de cara al usuario. Es la encargada de definir la interfaz de usuario.
- **Controlador**. Capa encargada de manejar las peticiones del usuario y de comunicar las capas anteriores para que obtengan lo necesario de dicha petición. Básicamente lo que hace es determinar la petición, requerir al modelo los datos necesarios y enviarles a la vista para que genere el resultado final que llega al usuario. La petición de usuario se lanza por una acción que realiza el usuario (un clic de ratón, desplazarse por la página...)

  En definitiva, el controlador es un mediador entre modelo y vista.

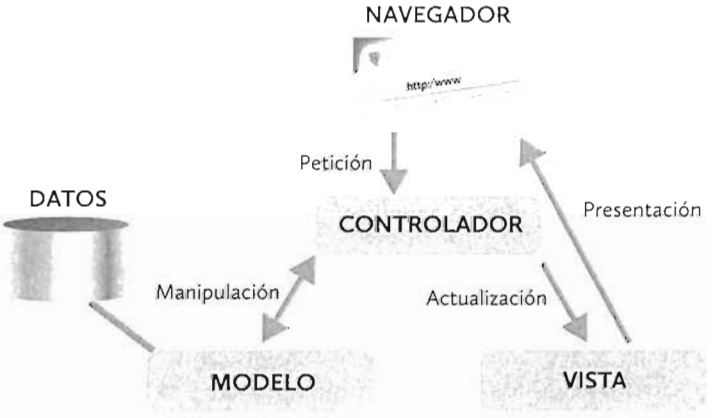


Figura 1.7: Funcionamiento habitual del paradigma MVC

El esquema de funcionamiento más habitual, ya que hay varias concreciones sobre este modelo. El más habitual es el que se muestra en la Figura 1.7. En ella se observa cómo el cliente, a través del navegador en el caso de una aplicación web, genera una petición, y ésta llega a la capa Controlador que se encarga de contactar con el Modelo, a fin de acceder y modificar, si la petición

lo requiere, los datos. El Controlador recibe los datos y con ellos pide a la capa Vista que modifique la presentación que ve el usuario.

La ventaja de este modelo de trabajo está en que tenemos una división que facilita la abstracción para poderse concentrar en tareas más concretas, además de facilitar el trabajo en equipo y el mantenimiento.

1.6 TECNOLOGÍAS PARA CREAR APLICACIONES WEB

En el Apartado 1.4.1 *"Funcionamiento en el lado del cliente"*, en la página 22 , se exponen las tecnologías del lado del cliente relacionadas con la creación de aplicaciones web. En este apartado se comentarán los principales lenguajes y plataformas para programar el back-end, el lado del servidor.

1.6.1 CGI

El llamado **Interfaz de Pasarela Común** (***Common Interface Gateway***) fue la primera tecnología que permitía crear aplicaciones en el lado del servidor.

La idea es crear una aplicación en un servidor (en principio utilizando cualquier lenguaje de programación) en base a un formato especial que permita comunicar dicha aplicación con el servidor web. La comunicación entre aplicación y servidor se realiza mediante la interfaz CGI.

Cuando a un servidor web le llega una petición http que contiene implícita una llamada a una interfaz CGI, el servidor web pide a la aplicación correspondiente que resuelva dicha petición, que desde su origen contendrá los parámetros necesarios para su correcta finalización, y le envíe el resultado, que será una respuesta compatible para ser enviada directamente por el servidor web.

En un caso típico, el funcionamiento es el siguiente:

[1] El cliente realiza una petición que provocará una solicitud CGI. Lo más habitual es el envío de datos de un formulario para su proceso; eso suele provocar una petición http de tipo **GET** o **POST** a la que se le envían como parámetros del mensaje (en el caso de POST) o como cadena de consulta (caso de GET) los datos del formulario.

[2] El servidor web recibe la petición y detecta la necesidad de lanzar una aplicación CGI a la que se pasa como parámetros los datos recibidos por GET o POST.

[3] A través de CGI el servidor de aplicaciones accede a los parámetros necesarios y se ejecuta la aplicación con esos datos.

[4] El resultado se comunica al servidor web indicando el formato de la respuesta (concretamente el llamado **tipo MIME** de la misma) y este envía el resultado al cliente a través de una respuesta http. Por supuesto, el resultado de la petición CGI suele estar en formato entendible por un navegador (es decir, normalmente un documento HTML).

1.6.2 LENGUAJES DE PROGRAMACIÓN HABITUALES PARA CREAR APLICACIONES EN EL LADO DEL SERVIDOR

La forma de funcionar de CGI ha sido la habitual durante muchos años y sigue teniendo influencia en la actualidad. Lógicamente hay lenguajes más apropiados que otros para crear aplicaciones de servidor.

Hay lenguajes que tienen una especial facilidad para crear aplicaciones web, casi siempre ayudados de frameworks (marcos/plantillas) de trabajo. Comparados con los lenguajes de script (que se comentan en el apartado siguiente) son más potentes, pero también más difíciles de aprender.

- **Perl**. Es el lenguaje clásico para crear aplicaciones de tipo CGI. Es muy poderoso para manejar ficheros y expresiones regulares, de ahí su éxito.
- **Python**. Es un lenguaje fácil y potente que además dispone de numerosas facilidades para crear aplicaciones de red y aplicaciones gráficas. Es el lenguaje que más crece actualmente para crear aplicaciones en el lado del servidor.
- **Ruby**. Es también un lenguaje fácil de aprender y que tiene bastante potencia. Es similar a Python. Se hizo muy popular gracias al marco de trabajo **Ruby on Rails** que facilita la creación completa de una aplicación web usando un patrón MVC.
- **Java**. Desde su nacimiento se convirtió en el lenguaje más popular para crear aplicaciones relacionadas con la red. Es muy potente y versátil, pero más difícil que los anteriores de aprender. Sus posibilidades para programar en el lado del servidor son muy extensas. Durante mucho tiempo ha sido el rey indiscutible de los lenguajes para programar aplicaciones web; ahora tiene mucha más competencia.
- **C#**. Ideado por Microsoft para desbancar el éxito de Java, es muy popular para programar aplicaciones utilizando servidores **.NET** (se comentan más adelante).
- **JavaScript**. El lenguaje clásico del lado del cliente está tomando mucha fuerza para programar en el lado del servidor, gracias a la aparición del servidor de JavaScript **node.js**, y a los numerosísimos marcos de trabajo especializados como **Angular.js** o **backbone.js**.

1.6.3 LENGUAJES DE SCRIPT DE SERVIDOR

Este tipo de lenguajes se basan en la incrustación del código HTML de una página web. Todos ellos indican mediante una apertura de etiqueta especial. (por ejemplo **<?php** en el caso de **PHP** o <% en el caso de **JSP**) un bloque de código que no es HTML, sino código perteneciente al lenguaje de servidor.

Además la página en sí tendrá una extensión especial, gracias a la cual el servidor web detecta que el archivo es de un tipo que requiere la intervención de un servidor de aplicaciones. Por ejemplo .php parta el lenguaje PHP o .jsp para el lenguaje JSP.

Los principales lenguajes de script son:

- **PHP** (*Personal Home Pages*). Es el más utilizado y de hecho es la tecnología del lado del servidor (en general) más utilizada. El código PHP se basa en el lenguaje C y en Perl. Es un lenguaje fácil de aprender, pero poco valorado por los programadores más formales por su falta de rigidez en las normas de escritura y poca potencia para hacer aplicaciones muy complejas.
- **ASP** (*Active Server Pages*). Tecnología de Microsoft similar a la anterior. Está pensada para utilizar en servidores web de Microsoft (concretamente en servidores **IIS, Internet Information Server**) . Actualmente se conoce como **ASP.net,** ya que se utiliza en la plataforma **.NET** de Microsoft.
- **JSP** (*Java Server Pages*). Lenguaje de script del lado del servidor basado en Java. Es la forma de script utilizada cuando una aplicación web se ha programado en Java.
- **ColdFusion**. Otro lenguaje de scripts, esta vez propiedad de Adobe. Es el más sencillo de todos, pero requiere servidores especiales (servidores **ColdFusion**, que son caros). Está en claro desuso.

1.6.4 PLATAFORMAS DE DESARROLLO DE SERVICIOS WEB EMPRESARIALES

Se trata de soluciones de desarrollo de aplicaciones web completas. Incluyen por lo general:

- Una estructura de directorios rígida en la que guardar cada tipo de archivo (programas, páginas de servidor, páginas HTML, archivos de configuración...)
- Una serie de archivos de configuración que permitirán configurar la aplicación como un servicio http que sepa cómo atender (al menos a qué programa desviar la petición y con qué datos) cada petición que le llegue.
- Un conjunto de programas realizado en uno o más lenguaje de programación completos y potentes (Java, C#...)
- Un conjunto de documentos ya preparados para enviar al cliente; se les conoce como archivos estáticos porque no requieren la traducción del servidor de aplicaciones. Son los archivos HTML, CSS, JavaScript, imágenes...
- Una serie de archivos de script de servidor (JSP, ASP.net...)
- Componentes reutilizables por los elementos anteriores (como los **Enterprise Java Bean,** por ejemplo, de Java).
- Librerías diversas (de acceso a bases de datos, de uso de módulos especiales para los programas...)
- Facilidad para incorporar patrones y marcos de trabajo compatibles con el paradigma MVC.

Las plataformas más populares desde hace años son:

- J2EE (*Java 2 Enterprise Edition*). Nombre que se da a la plataforma de creación de aplicaciones web empresariales de Java. Está formada fundamentalmente por el propio lenguaje Java, **EJB** (*Enterprise Java Beans*, componentes reutilizables empresariales), **servlets** y **JSP** además de otros componentes.
- **.NET**. Plataforma de Microsoft que permite (entre otras muchas posibilidades) crear aplicaciones y servicios web, haciendo especial énfasis en el transporte de datos mediante XML.

1.6.5 FRAMEWORKS MVC

En inglés *framework* se puede traducir como estructura. En el sentido que nos ocupa, un framework sería un marco de trabajo. MVC son las siglas del **Modelo-Vista-Controlador**, comentado antes, un paradigma de programación de aplicaciones que separa la misma en tres visiones diferentes.

Todos los frameworks (al igual que las plataformas de desarrollo) imponen, o al menos recomiendan, una estructura concreta de trabajo que separa los diferentes elementos de la aplicación; no solo el modelo, la vista y el controlador, sino los archivos de configuración, elementos estáticos, desvío o enrutamiento de las peticiones http...

Los frameworks MVC más populares actualmente son:

- **Ruby on Rails**. Se trata de un marco de trabajo muy exitoso por la facilidad que tiene de programar y sus buenos resultados visuales. Se puede ejecutar en casi cualquier servidor web, basta con instalar el componente correspondiente. El lenguaje en el que se basa es Ruby.
- **Apache Struts**. El marco de trabajo más famoso para la creación de aplicaciones J2EE. Muy preparado para utilizar con servidores web Apache.
- **Spring**. Otro marco para trabajar en Java J2EE que tiene bastante éxito. Tiene incluso una versión para las aplicaciones .NET.
- **Django**. Escrita en **Python** y pensada para utilizar en ese lenguaje, dice de sí misma no ser MVC, pero permite perfectamente programar en ese paradigma.
- **Zend**. Framework escrito para PHP. El más popular para este lenguaje.
- **Angular.js**. Marco de trabajo para JavaScript creado por Google que cada vez tiene más adeptos. Es el más utilizado para la creación de aplicaciones web SPA de página única (**SPA, Single Page Applications**) que permiten que la página vaya mutando para ir cargando solo los elementos de la misma que se requieren. Realmente su paradigma es una variante de MVC conocida como **MVVM** (*Modelo-Vista-Vista Modelo*). Es el marco que más crece actualmente.
- **Backbone.js**. Marco ligero de programación de aplicaciones web usando JavaScript como único lenguaje; es uno de los primeros que se crearon para JavaScript.

ACTIVIDAD 1.2: TECNOLOGÍAS PARA CREAR APLICACIONES WEB

- La página **W3Techs** ofrece numerosas estadísticas sobre la implantación de tecnologías en Internet.
- Por ejemplo la dirección:

  http://w3techs.com/technologies/overview/programming_language/all

  contiene información sobre el uso de lenguajes de programación para implementar aplicaciones web.
- Observa los diferentes apartados y toma conclusiones sobre los lenguajes utilizados en la web (aunque es una muestra parcial, es bastante significativa).
- En el menú ***Technologies*** disponible en la portada podremos obtener acceso a todas las estadísticas sobre la tecnología implementada en las aplicaciones web. Elije ***Server-side Languages*** y veremos cuáles son los lenguajes de servidor más comunes en las aplicaciones web actuales.
- Después, si hacemos clic en el apartado ***Historical Trend*** veremos la tendencia que han tenido los lenguajes de servidor en los últimos meses. Si elegimos **Yearly** en el menú lateral izquierdo veremos la tendencia por años.
- Lo mismo podremos hacer sobre cualquier otra de las estadísticas que la página nos ofrece.

1.7 APLICACIONES EN LA NUBE. *CLOUD COMPUTING*

Se trata de uno de los términos más influyentes en el mundo de la computación de los últimos años. Consiste en ofrecer, en forma de servicio, cualquier prestación que se pueda programar en un sistema informático. Este servicio es accesible desde cualquier dispositivo con conexión a Internet ya que se distribuye en una serie de servidores, cuyo número puede ser enorme y cuya localización geográfica puede abarcar el planeta entero.

Es un nuevo modelo de trabajo para los usuarios que ha requerido de un cambio drástico en la forma de crear aplicaciones en general, de modo que las mismas puedan atender a una demanda ingente de peticiones de servicio, gracias a una altísima tolerancia a fallos en los servidores.

De esta forma el usuario puede conectarse desde cualquier parte del planeta y recibir servicio de una forma eficiente, haciendo que no necesite instalar aplicaciones en su ordenador. Lo que subyace en este modelo, es la idea de que cualquier necesidad informática se convierta en un servicio al que se accede a través de la red.

Empresas como **Google**, **Facebook** o **Amazon** fueron las primeras que tuvieron que atender a gran velocidad a las peticiones de millones de usuarios; Amazon, concretamente, fue la primera en permitir a los desarrolladores de aplicaciones alquilar su propia nube para que pudieran disponer de una infraestructura virtual en la nube y así desarrollar sus aplicaciones distribuidas.

En la práctica, esta idea permite que se utilice Internet como la base de trabajo, sustituyendo así al propio ordenador personal, el cual puede ser un equipo de menor potencia al delegar en la nube el proceso de las tareas y el almacenamiento de la información.

Algunos ejemplos de servicios en la nube son:

- **Servicios de almacenamiento en la nube**. Se trata de servicios que permiten al usuario almacenar datos que, por lo general, se suben a la nube sincronizando alguna carpeta de la máquina del usuario. Esto permite que los usuarios tengan copia de seguridad instantánea (incluso algunos servicios permiten almacenar varias versiones de cada documento), además de acceso a sus archivos desde todo tipo de dispositivos y ubicaciones. Ejemplos de servicios concretos de este tipo son **DropBox, Microsoft OneDrive, Google Drive** o **SugarSync**.

- **Aplicaciones de ofimática web**. Se trata de aplicaciones en la nube que permiten crear, modificar, compartir y almacenar documentos típicos de oficina (textos, hojas de cálculo, presentaciones...) Entre las más conocidas están **Google Docs** y **Microsoft Office 365**.

- **Copias de seguridad en línea**. Al estilo del disco virtual, pero pensado para que las empresas tengan un respaldo en caso de pérdida de información. En este caso la seguridad es crítica, así como disponer de todas las versiones previas de copia. Los servicios de copia de seguridad más conocidos son **Carbonite, Backblaze** y **CrashPlan**.

- **Calendarios**. Permiten disponibilidad permanente de los datos de agenda personal. Los calendarios de Google, Microsoft y Apple son los más populares.

- **Contactos.** Basados en la misma idea, se almacenan en un servicio en la nube y podemos acceder a ellos desde cualquier ordenador. Incluso la propia agenda de contactos de los móviles ha pasado a ser una base de datos en la nube (con lo que cambiar de teléfono no implica perder los contactos).

- **Sistemas operativos web**. Permiten utilizan un ordenador virtual disponible a través de Internet utilizando un sistema operativo virtual. El más conocido es **EyeOS** pero incluso el sistema **ChromeOS** de Google se le considera sistema en la nube, al instalar un sistema ligero y utilizar aplicaciones web en lugar de aplicaciones de escritorio instaladas en la propia máquina.

- **Redes sociales.** Como **Facebook**, **Instagram**, **twitter** o **LinkedIn**. Almacenan miles de imágenes, textos y vídeos de sus usuarios que las suben para que sus contactos las puedan ver y comentar.

- **Bibliotecas multimedia.** Como **Youtube** o **Flickr** que permiten una enorme cantidad de vídeos o fotos que los usuarios pueden consultar.

- **Marcadores en línea**. Como **del.icio.us** o **Digg**, haciendo que, aunque estemos en otra máquina distinta a la nuestra habitual, tengamos acceso a nuestros marcadores.

Existen detractores de este tipo de servicios, puesto que al final la información personal o empresarial queda en manos de empresas privadas. E incluso hay quien afirma (especialmente algunos movimientos a favor del software libre) que supone un retroceso en los derechos de los usuarios, ya que se asemeja el funcionamiento de los sistemas de los años 70 en los que los

trabajadores utilizaban un terminal muy ligero que se conectaba a un ordenador central de cuya potencia y software dependían; sin poder el usuario mejorar esa potencia por su cuenta.

1.7.5.1 MODELOS DE CLOUD COMPUTING

Los servicios en la nube se suelen dividir en tres tipos:

- **SaaS**, acrónimo de ***Software as a Service*** (**Software como Servicio**) que hace referencia a las aplicaciones creadas, de forma que se accede a ellas como un servicio en la red. Se trata en definitiva de una aplicación que no requiere su instalación en el equipo del usuario, sino que accede a ella mediante el protocolo **http**.

  Por ejemplo, **Google Calendar** es un ejemplo de SaaS ya que es un software que permite almacenar nuestros contactos en la nube de Google. Podemos acceder desde la página web https://www.google.es/contacts, desde una aplicación de contactos de nuestro móvil (especialmente de Android) o incluso mediante aplicaciones de correo de escritorio, como por ejemplo **Mozilla Thunderbird**.

  Juegos en línea, escritorios virtuales, correo, redes sociales, software de creación de presentaciones online... Cada vez hay más servicios de este tipo cubriendo absolutamente todas las necesidades de los usuarios.

  En definitiva un servicio es de tipo SaaS, cuando el servicio parece ser una aplicación normal que podríamos tener instalada en nuestro equipo.

- **PaaS**, acrónimo de ***Platform as a Service*** (**Plataforma como Servicio**). Se trata de un servicio que proporciona una plataforma preparada para desarrollar aplicaciones. Por plataforma se entiende una la completa de elementos: sistema operativo, librerías, compiladores, APIs de desarrollo,etc. necesarias para crear software en una determinada tecnología.

  Un ejemplo de **PaaS** sería **Heroku**, que es un servicio en la nube que permite el desarrollo de aplicaciones web completas. Un programador de Heroku tendrá a su disposición una base de datos (**Heroku** usa como base de datos una versión de **PostgreSQL**) y un servidor de aplicaciones web (**node.js**, **Ruby on Rails...**) perfectamente configurado para que simplemente el programador suba el código a la nube y ya funcione. Rivales de Heroku son **Google App Engine**, **Red Hat Open Shift** o **Cloud Foundry**.

  La ventaja es que los desarrolladores pueden trabajar en su aplicación desde diferentes estaciones de trabajo, ya que el entorno de trabajo y la propia aplicación realizada está en la nube.

- **IaaS**, ***Infrastructure as a Service*** es un servicio en la nube que nos permite virtualizar hardware de todo tipo: un servidor, una red de ordenadores, un disco duro... Podemos acceder a ese hardware como a un servicio más de Internet.

  La ventaja es que dispondremos de máquinas que no tendremos físicamente instaladas nosotros, y a las que accederemos desde un navegador o una aplicación móvil.

El ejemplo más conocido de IaaS es **Amazon Web Services** (AWS) que nos permite instalar y gestionar servidores virtuales en la red de Amazon (además de otro tipo de servicios). Otros ejemplos son **Google Compute Engine** o **Microsoft Azure**.

A veces esta división se realiza para tener en cuenta las necesidades de una organización. Un servicio cloud computing de tipo SaaS soluciona una necesidad muy concreta. Un PaaS permite solucionar problemas más complejos, porque la propia organización puede crear sus propias aplicaciones. Finalmente IaaS es una solución aun más completa (y compleja) ya que puede atender y personalizar todas las necesidades computacionales de una organización.

1.7.5.2 IMPLEMENTACIÓN DE *NUBES* PARA EL CLOUD COMPUTING

Podemos entender que una nube realmente es un conjunto de servidores conectados que trabajan de forma distribuida. Cuando implementamos un servicio en la nube, realmente lo que tenemos es una aplicación que se distribuye en una nube de servidores ofreciendo el servicio que deseemos implementar.

Se considera que hay tres tipos de nubes para implementar un servicio:

- **Nubes públicas**. Son nubes ofrecidas por empresas externas. Ofrecen alguno de los servicios comentados en el Apartado 1.7.5.1 *"Modelos de Cloud Computing"*, en la página 36 . Ofrecen facilidad para implementarlos ya que toda la gestión es externa; el problema es que se delega, en terceros, información que puede ser crítica; además de que la personalización es muy limitada, al depender de las posibilidades que nos otorga los propietarios de la nube.
- **Nubes privadas**. Son implementadas por la propia empresa que necesita la nube. Requiere una inversión más grande, pero nos permite personalizar absolutamente el servicio, así como ser nosotros quienes gestionamos nuestros datos sin intervención de terceros. Esta es la forma adecuada si necesitamos servicios en la nube que utilicen información crítica o estratégica.
- **Nubes híbridas**. Son una mezcla de las anteriores, parte de nuestro negocio se delega a una nube pública y parte lo gestionamos en nuestra nube privada. Intenta aunar lo mejor de ambas opciones.

1.8 APLICACIONES WEB Y APLICACIONES MÓVILES (APPS)

El éxito de los teléfonos inteligentes (***smartphones***) ha permitido al usuario lograr que desde el móvil también se acceda a los servicios disponibles en Internet. Las aplicaciones presentes en los móviles (conocidas popularmente como ***Apps***) aúnan sencillez de uso y acceso directo para resolver problemas complejos o, al menos, muy interesantes para el usuario.

El éxito de estas aplicaciones ha sido tal que ha hecho palidecer a las aplicaciones web, de tal forma que si el modelo, antes comentado, de funcionamiento de las aplicaciones web era la instalación de un único software (el navegador) para acceder a todos los servicios, el modelo de

las aplicaciones móviles es el contrario: la instalación de una aplicación (**app** en la terminología móvil).

Hay cierta similitud entre aplicaciones móviles y aplicaciones web. Es más, las aplicaciones que un smartphone (o una tableta) puede ejecutar, pueden ser de dos tipos:

- **Web apps**. Son aplicaciones web (se ejecutan en el navegador), pensadas para ser utilizadas en un dispositivo móvil; consideran el tamaño, resolución y orientación habitual de un móvil, así como la facilidad de estos dispositivos para cambiar la orientación (de vertical y horizontal) para construir la aplicación.

  Hoy en día cualquier aplicación web debería considerar tener al menos una versión de tipo web app, porque empieza a ser más habitual navegar desde un dispositivo móvil que desde un ordenador.

  Las web apps se crean fundamentalmente en **HTML5**, **CSS** y **JavaScript/jQuery**; es decir, las técnicas actuales habituales para crear aplicaciones web.

- **Apps nativas**. No son aplicaciones web, son aplicaciones programadas en los lenguajes nativos de los dispositivos móviles: **Java**, **C#**, **C++**, **Objective C** o **Swift**, dependiendo de la plataforma. Si queremos que nuestra app nativa sea compatible con todas las plataformas disponibles para los dispositivos móviles, habrá que programarla en cada lenguaje de cada plataforma; lo que es más costoso, ya que nos obliga a hacer varias versiones de la misma aplicación.

  Tienen como principal ventaja que el hardware de la máquina está a nuestra disposición, ya que el lenguaje con el que se crean las *apps nativas* permite acceder al GPS, la luz LED del móvil, la cámara,etc.

  Otra ventaja importante es que se publican en un ***market*** (**Google market**, **Apple Store**...), una tienda online que permite buscar e instalar la aplicación en el dispositivo. Esos ***markets*** facilitan la difusión y promoción de la app nativa.

  Las apps nativas triunfan más que las web apps, porque les resulta incluso más fácil de manejar a los usuarios; a cambio los desarrolladores se tienen que esforzar más, al ser más compleja su programación.

  Las empresas que proporcionan servicios suelen utilizar como modelo habitual, crear una aplicación web para acceder a ese servicio y crear además aplicaciones móviles (apps nativas) acceder desde un smartphone o tableta.

También han aparecido frameworks de trabajo que permiten crear apps nativas compatibles con todas las plataformas. Las aplicaciones se crean al estilo de las aplicaciones web mediante los típicos lenguajes HTML, CSS y JavaScript. Pero utilizan librerías (**PhoneGap/Cordoba** es la más popular) especiales que permiten acceder al hardware de un móvil, incluidos el GPS, acelerómetro, teléfono, cámara de fotos, barómetro, etc. El código resultante se traduce mediante un intérprete y el resultado es una serie de apps, cada una de ellas orientada a una plataforma: Android, iOS, Windows, etc.

1.9 RESUMEN DE LA UNIDAD

- Las aplicaciones web llevan presente muchos años ya entre nosotros, pero desde la aparición de la web hasta la a actualidad cada vez los usuarios han requerido más de ellas.
- La evolución de la web ha hecho que las aplicaciones web se creen en tecnologías cada vez más sofisticadas. La velocidad en la innovación y, aparición y mejora de las tecnologías hace que una aplicación pueda utilizar numerosos lenguajes para su creación.
- Las aplicaciones web ofrecen ventajas sobre las tradicionales aplicaciones de escritorio que han provocado que cada vez se utilicen más y por lo tanto, los navegadores sean la herramienta de trabajo fundamental de los usuarios actuales.
- Las aplicaciones web tienen también ciertas desventajas sobre las aplicaciones de escritorio que hay que tener en cuenta.
- Las aplicaciones enriquecidas (***RIA***), la web social y la orientación a servicio son las bases de la web 2.0 y por lo tanto, la base del éxito de la creación de aplicaciones web. A esto hay que sumar la web semántica y la web inteligente, componentes de la llamada web 3.0.
- La creación de aplicaciones web actuales sigue un modelo de tres capas: presentación, lógica y negocio. Esto permite dividir la programación de aplicaciones en estos tres aspectos, incluso utilizando lenguajes diferentes en cada capa, salvo en los llamados modelos ***Full Stack*** en los que se utiliza el mismo lenguaje en todas las capas.
- Cada vez hay más aplicaciones y servicios en la nube cuyo modelo (***cloud computing***) se basa en que tanto la aplicación como la información que maneja se encuentran en Internet y no en el ordenador del usuario.
- ***SaaS***, ***PaaS*** e ***IaaS*** son los tres modelos de cloud computing disponibles tanto para los usuarios como para los desarrolladores de aplicaciones web, que se distinguen por el tipo de servicio que ofrecen.
- Debido a la proliferación de aparatos móviles para acceder a Internet, ha aparecido un nuevo modelo de aplicación (app) para este tipo de aparatos que rivaliza directamente con las aplicaciones web.

1.10 TEST DE REPASO

1.- ¿Cuáles de las siguientes son tecnologías del lado del servidor?

a) JavaScript
b) Python
c) JSP
d) CSS
e) J2EE
f) HTML
g) PHP
h) Flash
i) Applet Java
j) Servlet Java
k) Silverlight
l) ASP
m) jQuery
n) Perl
o) CGI
p) Ruby on Rails

2.- ¿Qué capa es la que permite acceder a las bases de datos en un arquitectura de tres capas de una aplicación web?

a) Presentación
b) Lógica
c) Negocio

3.- ¿Qué capa es la que se encarga de crear el HTML y CSS de una aplicación web datos en un arquitectura de tres capas?

a) Presentación
b) Lógica
c) Negocio

4.- En un modelo MVC ¿Qué parte es la que se encarga de recoger y encauzar las peticiones del cliente?

a) Modelo
b) Vista
c) Controlador

5.- Si queremos crear una aplicación pensada para utilizarse en un móvil y necesitamos que esa aplicación maneje la cámara del móvil ¿qué tipo de aplicación necesitamos crear?

a) Una web app
b) Una app nativa
c) Una aplicación de escritorio
d) Una aplicación *Full Stack*

6.- El front-end de una aplicación web...

a) ...está en la capa de presentación
b) ...está en la capa lógica
c) ...está en la capa de negocio
d) ...abarca la capa de negocio y la lógica

7.- Una aplicación Full Stack es...

a) Aquella que utiliza el mismo lenguaje tanto en el front-end como en el back-end
b) La que utiliza como servidor de aplicaciones web el propio servidor web
c) La cubre todas las necesidades de un usuario
d) La que se configura como un servicio completo de red

8.- Una empresa desea manejar un servidor Linux virtual en Internet y quiere acceder a ese servidor a través de una petición http. El tipo de cloud computing a utilizar es:

a) SaaS
b) PaaS
c) IaaS
d) Ninguno de los anteriores

Un programador se plantea desarrollar una aplicación web y como va a trabajar en varios dispositivos quiere un entorno listo para codificar y probar la aplicación que sea accesible a través de un servicio de red. El tipo de cloud computing a utilizar es:

- SaaS
- PaaS
- IaaS
- Ninguno de los anteriores

Una empresa decide implementar un servicio en la nube, para ello aloja el servicio en la nube alquilada de otra empresa, ya que no quiere implementar servidores propios ¿Qué tipo de nube utilizará?

- Nube pública
- Nube privada
- Nube híbrida
- Ninguno de los anteriores

11. ¿Cuáles de los siguientes conceptos se identifican con la web 2.0?

- a) RIA, Aplicaciones Enriquecidas de Internet
- b) Web semántica
- c) Web estática
- d) Web social
- e) Web inteligente
- f) SOA

12. ¿Cuáles de los siguientes conceptos se asocian a la web 3.0 más que a la web 2.0?

- a) RIA, Aplicaciones Enriquecidas de Internet
- b) Web semántica
- c) Web estática
- d) Web social
- Web inteligente
- SOA

¿Cuál de las siguientes ideas refleja mejor la situación de la web actual?

- Ningún usuario de la web crea contenido
- Pocos usuarios de la web crean contenidos para ella
- c) El número de usuarios que crean contenido y el número de los que no crean contenido es equiparable
- d) Prácticamente todos los usuarios de la web son creadores de contenidos

¿Cuál de las siguientes afirmaciones es verdadera?

- a) No se pueden crear aplicaciones web actualmente con el lenguaje PHP
- b) No se pueden crear aplicaciones web actualmente con el lenguaje JavaScript
- c) No se pueden crear aplicaciones web actualmente con el lenguaje Java
- d) En las aplicaciones web actuales es obligatorio utilizar el paradigma MVC
- e) Todas las respuestas anteriores son falsas

¿Cuáles de las siguientes son ventajas de las aplicaciones web respecto a las aplicaciones de escritorio?

- a) Son más potentes
- b) Aseguramos que el usuario disponga siempre de la última versión
- c) No se requiere instalación
- d) Requieren conectividad
- e) Aprovechan mejor el hardware
- f) Disponibles desde cualquier dispositivo

UNIDAD 2

PREPARACIÓN DEL ENTORNO DE TRABAJO

OBJETIVOS

- Reconocer los elementos necesarios para crear aplicaciones web
- Analizar las fases de desarrollo de una aplicación
- Reconocer las opciones de implementación de aplicaciones web
- Instalar servidores de aplicaciones web en diferentes sistemas
- Instalar servidores de bases de datos en diferentes sistemas
- Conectar todos los elementos de una instalación
- Instalar soluciones compactas de implementación de aplicaciones web para entornos de prueba y desarrollo
- Establecer parámetros de configuración en la instalación
- Establecer la seguridad de la instalación

CONTENIDOS

2.1 ELEMENTOS NECESARIOS PARA CREAR APLICACIONES WEB

Como se ha visto en la unidad anterior, las aplicaciones web requieren ser implementadas en un servidor en Internet. Esto complica las necesidades que se requieren para empezar a programar, en comparación con lo que se necesita para programar aplicaciones clásicas de escritorio.

Además, como veremos en el Apartado 2.2 *"Creación profesional de aplicaciones web. Modelo de tres estados"*, en la página 47, cuando se desarrollan aplicaciones a nivel profesional se suelen configurar dos entornos de trabajo: uno en un equipo local y otro en Internet.

Para el aprendizaje basta con equipar a nuestro equipo local. Los elementos que necesitaremos configurar son:

- **Servidor web**. Encargado de recibir las peticiones http relacionadas con nuestra aplicación. Más adelante veremos las opciones disponibles en el mercado. A la hora de seleccionar el elegido, hay que pensar en nuestras necesidades; si queremos que sea gratuito o no, cuál es el Sistema Operativo (por ejemplo, el servidor web **IIS** solo funciona para Windows), el número y tipo de peticiones que debe atender, cuál va a ser el lenguaje elegido (casi todos los programadores en PHP instalan Apache por su facilidad de configuración) si deseamos soporte o no, etc.

- **Instalar un servidor de aplicaciones web**. Hoy en día bastará con añadir módulos al servidor web. Por ejemplo, para programar en **PHP**, basta activar e instalar el módulo PHP en el caso del servidor web **Apache**, o instalar extensiones **FastCGI** para PHP en el caso de otros servidores.

  Hay soluciones compactas de servidores web que son además de aplicaciones, como es el caso de **Apache Tomcat**, servidor web con capacidad para implementar aplicaciones **Java Enterprise**.

- **Instalar un servidor de base de datos**. Realmente una aplicación web podría no necesitar de la instalación de un servidor de base de datos; pero francamente es rara esta situación. Normalmente necesitaremos un sistema gestor de bases de datos.

  El sistema gestor de base de datos habitual para programar en PHP es MySQL; pero se puede instalar el que deseemos (siempre que dispongamos de la forma de comunicar con él desde nuestra aplicación web).

- **Instalar/configurar conectores de base de datos**. Todos los componentes deben estar conectados. En el caso de la base de datos puede implicar instalar conectores en el servidor, de modo que nuestra lógica conecte con el negocio. Los conectores es lo que permitirá que podamos invocar al servidor de bases de datos desde el código de las aplicaciones web.

  Por ejemplo, en las aplicaciones Java se necesita instalar un conector **JDBC** y en las aplicaciones de Microsoft conectores de tipo **ODBC** u **OLE DB**. En el caso de PHP necesi-

taremos instalar el módulo PHP necesario para conectar con la base de datos y, a veces, una librería externa.

- **Entorno de desarrollo**. La creación de aplicaciones web implica escribir muchas líneas de código. El código se crea en archivos de texto, con lo que cualquier editor de texto valdría para hacerlo. Pero hay software de edición de código que aporta mejoras al escribir código como:
  - **Coloreado de sintaxis**. Que permita diferenciar las palabras clave del lenguaje.
  - **Autocorrección**. Que permita detectar los errores de código a la vez que vamos escribiendo.
  - **Depuración de código**. Que facilita la realización de pruebas con el código.
  - **Facilidades para la navegación**. Búsqueda rápida de apartados en el código, ventanas con acceso rápido a las diferentes secciones del código, lista de archivos en uso...
  - **Creación de *snippets***. Son abreviaturas que facilitan la escritura de código habitual.

  Todas estas, y muchas más, ventajas hacen que estas herramientas sean indispensables para trabajar. En unidades posteriores comentaremos las herramientas más recomendables para trabajar con PHP.

- **Depurador de código**. Se trata de un software que se instala en el servidor de aplicaciones y que permite realizar pruebas con el código. Estas pruebas sirven para detectar errores, cálculo de rendimiento, visualizar estado de variables, monitorizar el estado del servidor...

  En el caso de PHP el sistema de depuración de código más conocido y utilizado es **XDebug**; aunque también se utilizan **Zend Debugger** (especialmente para los amantes de **Zend Studio**), **FirePHP** (se utiliza como extensión del navegador **Mozilla Firefox**) y **PHP Console** (extensión del navegador **Google Chrome**).

- **Software de máquina virtual**. No es obligatorio instalar una máquina virtual pero sí es interesante, ya que si instalamos directamente el software servidor en nuestra máquina de trabajo, hará que su ejecución sea más lenta, además de que los fallos en instalaciones y pruebas pueden afectar al funcionamiento. Una máquina virtual permite hacer pruebas sin la preocupación que resulta de manipular la máquina de la que dependemos para trabajar a diario.

  Sin duda el software gestor de máquinas virtuales más popular hoy en día es **VMWare**, el problema es que es una solución de pago (aunque hay disponibles licencias educativas muy interesantes). Otra solución muy popular es **Oracle Virtual Box**, que tiene la ventaja de que es gratuita.

  Una desventaja de trabajar con máquinas locales reales es que si cambiamos de máquina para trabajar tendremos que instalar de nuevo el software de trabajo. Las máquinas virtuales se pueden copiar de un entorno a otro, aunque si hablamos de entornos con diferentes sistemas operativos, podríamos tener problemas.

Hay soluciones como **Vagrant** o **Docker**, que permiten configurar un entorno virtual de trabajo mediante un archivo de configuración en formato texto. A través de esa configuración estos sistemas generan una máquina virtual preparada para el desarrollo de aplicaciones. De modo que podremos trabajar fácilmente desde distintas ubicaciones ya que la configuración se duplica fácilmente (basta con copiar el archivo de texto) y con ella se genera una máquina optimizada para el nuevo entorno. La máquina que resulta es exactamente la misma, incluso podremos utilizar ese archivo de configuración para generar la máquina final real en el entorno de producción; lo que implica que podremos utilizar exactamente el mismo entorno tanto en la fase de desarrollo como en la de producción.

- **Sistema de control de versiones.** Una situación que suele ocurrir a menudo a los programadores, es la siguiente: tenemos una aplicación cuyo código funciona perfectamente y al hacer una mejor en la misma, el código deja de funcionar o no percibimos mejora alguna, y deseamos volver a la versión anterior del código.

  Volver a una versión anterior de nuestro código es muy fácil utilizando un sistema de control de versiones. Hoy en día el más popular es **Git**, creado por **Linus Torvalds** (creador, a su vez, del sistema Linux), pero también se utilizan **Mercurial**, **Subversial** o **SVN**.

  El funcionamiento es sencillo, cuando hemos programado nuestra aplicación y estamos conformes con lo realizado en ese punto, grabamos ese estado en nuestro sistema de control de versiones. Así iremos haciendo con cada situación que nos parezca *grabable*. Si en algún momento queremos volver a un punto anterior, podremos hacerlo.

  La mayoría de sistemas incluso admiten trabajo por ramas, en las que desde un punto concreto que hemos salvado podemos hacer varias pruebas diferentes (ramas) que den lugar a diferentes resultados. Eso mejora el trabajo en equipo, ya que podremos unir el trabajo de varias ramas, cada una de ellas realizada por una persona o equipo distinto, y compactar el trabajo completo.

  El éxito actual de estos sistemas procede de los repositorios de versiones en la nube. El más popular es **GitHub**, pero hay otros muy populares como por ejemplo **BitBucket.** En ambos casos hay disponible un uso gratuito del sistema, aunque pagando disponemos de más posibilidades. Su idea es simplemente centralizar nuestras versiones de código en la nube a través de un sistema de control de versiones. Tanto GitHub como Bitbucket admiten Git como sistema de control de versiones (Bitbucket además admite Mercurial). De modo que si cambiamos de ubicación, podremos sincronizar nuestro trabajo en la nube y así siempre disponer de todas las versiones realizadas desde cualquier ubicación.

- **Software de test.** En las aplicaciones web, la fase de pruebas resulta compleja porque requiere de la interacción de diversos elementos. Por ello se han creado herramientas que permiten automatizar esta fase. Las más conocidas, para el lenguaje PHP, son **PHPUnit** y **Selenium** (ésta se usa en otras plataformas). Hay incluso una corriente de la ingeniería del software que se basa en crear aplicaciones, empezando primero por las pruebas de test: es el modelo llamado **Test Driven Development** (TDD).

2.2 CREACIÓN PROFESIONAL DE APLICACIONES WEB. MODELO DE TRES ESTADOS

Cuando se realiza una aplicación web con la idea de que realmente sea un servicio final en Internet, se suelen configurar tres entornos de trabajo.

- ***Development*, entorno de desarrollo**. Normalmente este entorno se prepara en un entorno local de trabajo. Es, pues, un ordenador al que accedemos directamente, bien porque es un ordenador físico a nuestra disposición o bien porque es una máquina virtual almacenada en un ordenador físico. Este entorno tiene la ventaja de que es rápido y no necesita conexión a Internet para trabajar. En este manual, este es el entorno que prepararemos para empezar a trabajar.

  Cuando se trabaja en equipo, habrá varios entornos locales, por lo que las opciones comentadas anteriormente de exportación de entornos virtuales de trabajo (como **Docker** o **Vagrant**) y, sobre todo, los sistemas de control de versiones de código centralizados en la nube (como **GitHub** o **BitBucket**) son muy interesantes, ya que en cuanto los miembros del equipo modifiquen el código y lo sincronicen en la nube, estará a disposición del resto del equipo.

  En el entorno de desarrollo es donde se encuentra el código en fase de pruebas.

- ***Staging*, entorno de simulación real**. *Staging* es un término que podríamos traducir como ***puesta en escena***. Los entornos locales tienen el problema de que no pueden simular completamente el entorno en el que se ejecutará realmente la aplicación, cuando esté finalizada. Por ello, se trabaja en un segundo entorno (este sí es el mismo para todos los miembros del equipo) que residirá en Internet, simulando lo mejor posible el funcionamiento real de la aplicación.

  Lo habitual es utilizar un servicio virtualizado en la nube como **AWS** de **Amazon** o **Google Compute Engine** de **Google**, que nos permitirá configurar un servidor virtual instalando todo lo necesario (Sistema Operativo, Servidor de Aplicaciones, Sistema de Base de datos, Depurador...) y de esa forma podremos acceder a él, normalmente mediante **SSH**, como si fuera un servidor real. Amazon, por ejemplo nos permite, durante un año, configurar servidores con mínimas prestaciones en su nube de forma gratuita.

  Otra opción es utilizar servicios de tipo **PaaS**, como **Heroku** o **Google App Engine**, ambos con opciones gratuitas, que en lugar de ofrecernos un servidor completo virtualizado, nos ofrece una plataforma preparada de trabajo a la que simplemente subiremos el código y nuestra aplicación ya funcionará sin tener que instalar nada. Es una opción más cómoda, pero con menos posibilidades de personalización.

  En este entorno se implementan aplicaciones prácticamente terminadas, pero que necesitan ser probadas en las condiciones más parecidas posibles a las finales, y así pasar a la fase de producción con más tranquilidad.

- *Production*, **entorno en producción**. Este es el entorno de trabajo final, el que realmente utilizarán los usuarios. Aquí se trata de determinar las necesidades de nuestra aplicación: número de usuarios previstos, transacciones de datos por minuto, acceso concurrente, escalabilidad... para contratar el mejor servicio posible para alojar nuestra aplicación.

  El código enviado a este entorno es ya el código definitivo.

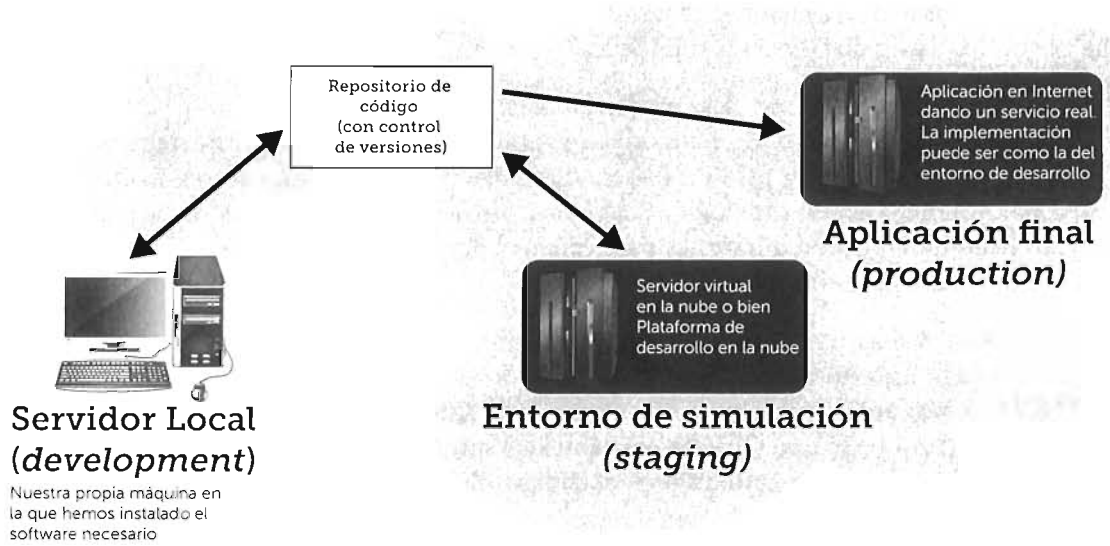


Figura 2.1: Modelo de tres estados para implementar una aplicación web; es la forma habitual de crear aplicaciones web profesionales.

2.3 INSTALACIÓN DEL SISTEMA OPERATIVO

La elección e instalación del Sistema Operativo es el primer paso, pero crucial, para preparar nuestro entorno local listo para crear aplicaciones web; tanto si lo hacemos físicamente en un equipo como si lo hacemos en una máquina virtual.

Fundamentalmente en cuanto al equipo tenemos tres posibilidades:

- **Linux**. Indudablemente en el mundo de los ordenadores personales Linux no deja de ser una rareza, ya que su cuota de mercado es muy pequeña en comparación al dominante Windows. Pero esto no ocurre en el caso de los servidores web, la cuota de servidores Linux/Unix alcanza un 70%. Lo que significa que en el mundo de las aplicaciones web, Linux es el sistema dominante.

  Solo esta razón podría ser suficiente para elegir Linux como Sistema Operativo en nuestro entorno local de trabajo; si lo hacemos, la migración a un servidor real de Internet será menos problemática.

Otra razón es el hecho de que la mayoría de distribuciones Linux son gratuitas y también es más barato contratar un servidor real en Internet en este sistema.

Finalmente hay que tener en cuenta que muchos servidores web (entre ellos Apache), se crearon pensando en ser instalados en un sistema Linux; lo mismo pasa con numerosas herramientas de trabajo. Linux posee un ecosistema de aplicaciones para los desarrolladores muy rico, además de una comunidad muy dinámica de desarrolladores.

La clara desventaja es que no es fácil utilizar un sistema Linux y que hay muchas versiones y distribuciones, lo que puede dar lugar a problemas de compatibilidad y de aprendizaje.

Las distribuciones Linux mas utilizadas son: **Debian** (muy robusta pero difícil de manejar), **Ubuntu** (basada en Debian pero con más facilidad de manejo) y **CentOS** (basada en Red Hat Enterprise). Todas ellas son gratuitas, pero **Red Hat Enterprise** y **Suse Enterprise** son versiones de pago muy utilizadas por su robustez y soporte para instalar servidores reales en Internet.

- **Windows**. Leído el punto anterior parece que es un error instalar Windows para programar aplicaciones web. Pero no es así. Sabedores de la cantidad de equipos con Windows, la mayor parte del software relacionado con el desarrollo de aplicaciones web posee versiones para este sistema. Además hay servidores web propios solo de los sistemas Windows, como es el caso de **Internet Information Server**, conocido como **IIS,** de Microsoft.

  Lógicamente para implementar un servidor, lo ideal es instalar alguna versión **Windows Server.** Pero podemos implementar servidores de aplicaciones web, prácticamente, en todas las versiones de Windows.

  Realmente las razones de instalar Windows en nuestro sistema local para crear aplicaciones web, pasan por el hecho de ser nuestro sistema habitual de trabajo y, por lo tanto, tener mucho más conocimiento sobre su gestión.

- **OSX**. El sistema operativo de los **MacIntosh** de **Apple**, es otra opción muy valorada por los desarrolladores de aplicaciones. Para ello debemos disponer de un equipo de esa marca. A pesar de su precio más elevado, es un entorno muy utilizado por los desarrolladores de aplicaciones porque para muchos aúna lo mejor de ambos mundos (Linux y Windows): facilidad de uso y de gestión, buen funcionamiento y un gran ecosistema de aplicaciones.

  No hay que olvidar que los Mac, al igual que los sistemas Linux, están basados en Unix, por lo que la configuración desde la línea de comandos es muy parecida a la de los sistemas Linux y eso facilita la migración a un servidor Linux típico en Internet.

La elección del sistema operativo puede estar determinada por la elección del resto de elementos. Por ejemplo, en el caso de PHP (que es el lenguaje elegido en este manual para programar aplicaciones web), siempre ha habido una muy buena relación con Linux. De hecho, lo que se conoce como **LAMP** (Linux+Apache+MySQL+PHP), sigue siendo la tecnología todavía dominante en el mundo de las aplicaciones web.

Bien es verdad, sin embargo, que la tendencia actual es que el sistema operativo sea poco determinante, gracias a las facilidades que tenemos actualmente para convertir nuestro entorno de una máquina a otra. Por ello, la elección del sistema operativo tendrá que ver con nuestras preferencias personales o bien con el rendimiento que nos ofrece.

2.4 INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR WEB

2.4.1 ELECCIÓN DEL SERVIDOR WEB

Una vez instalado el sistema operativo, el servidor web pasa a ser la siguiente decisión. Muchas veces estará determinada por la tecnología que hemos elegido para programar la aplicación web: por ejemplo, si decidimos programar en **.NET**, lo lógico es que el servidor sea **Internet Information Services** (**IIS**), si programamos en **PHP**, será más cómodo instalar **Apache**, si deseamos hacerlo en **JavaScript** será **Node.js**, etc.

Los servidores web más populares actualmente son:

- **Apache**. Es un servidor web muy veterano y sigue siendo el más utilizado. Es gratuito, de código abierto y francamente robusto. Actualmente está perdiendo cuota respecto a otros debido a la nueva naturaleza de las conexiones de usuario (más veloces y con requisitos continuos de modificaciones de datos).
- **nginx**. En realidad es el acrónimo de: **engine X** (que suena en español como ***enyainex***), es un servidor en alza que ofrece mejores prestaciones cuando hay que atender a numerosas conexiones. Es también gratuito y de código abierto. En solo 5 años ha pasado a instalarse en el 21% de servidores web de Internet.
- **IIS**. El servidor web de **Microsoft** es otra opción. No es gratuito aunque aparece como componente de los sistemas **Windows Server** (también en los Windows normales de tipo ***Professional*** y ***Enterprise***). Está muy preparado para programar aplicaciones web utilizando tecnologías de Microsoft, pero permite ser utilizado con otras tecnologías.
- **LiteSpeed**. Es un servidor web de código cerrado y propiedad de la empresa del mismo nombre. Permite leer las configuraciones de Apache y le reemplaza en muchos servidores por ser, según parece, más fácil y veloz. Hay una versión gratuita y abierta llamada **Open LiteSpeed**.
- **Apache Tomcat**. Servidor web pensado para ejecutar aplicaciones Java.
- **Node.js**. Servidor JavaScript con capacidad para crear servidores web especialmente con el módulo **Express.js**.

Para programar en PHP lo habitual es utilizar el servidor Apache debido a su ya larga tradición, estabilidad y gran número de usuarios. Actualmente hay servidores más rápidos y que suponen menos coste. Especialmente populares en este aspecto son nginx y Node.js, ambos se comportan mucho mejor que Apache cuando hay que atender a muchísimas conexiones y peticiones.

Mientras que node.js funciona como servidor JavaScript, nginx es un servidor web de estilo más clásico, muy utilizado para crear aplicaciones PHP.

Figura 2.2: Cuota de uso de servidores web según la web w3techs.com (abril 2015)

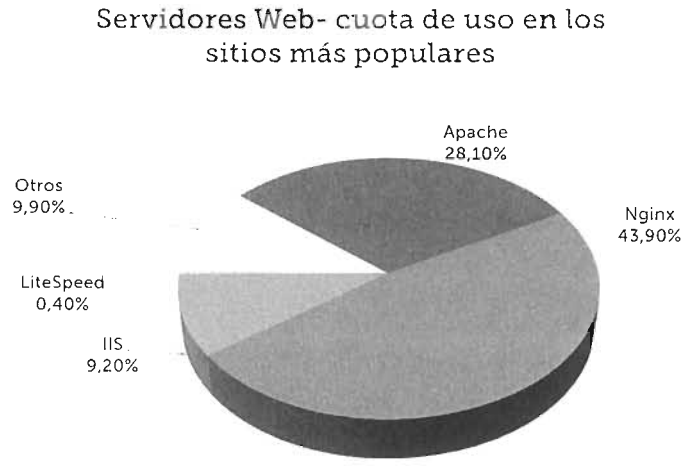


Figura 2.3: Cuota de uso de servidores web en sitios pertenecientes al top 1000 (1000 sitios más populares de Internet) según la web w3techs.com (abril 2015)

Las imágenes anteriores muestran el uso actual de servidores en los que destacan Apache y nginx. Se observa que nginx se va imponiendo en los servidores que tienen más popularidad (y por lo tanto, más visitas).

La última imagen (*Figura 2.4*) muestra el diferente uso de los servidores web. En horizontal se muestra el uso general de los servidores, siendo Apache el servidor más utilizado. En vertical se muestra el uso en servidores diferenciando por volumen de tráfico; aquí se observa que Node.js se usa mucho en los sitios con gran cantidad de tráfico, aunque su cuota general sea muy pequeña.

La elección final dependerá de las necesidades del servicio que ofrezcamos, además de considerar otras cuestiones: como por ejemplo el soporte. Los servidores que aparecen en todas las gráficas que hemos utilizado, son gratuitos y por ello muy utilizados. Pero no significa que sean los mejores, los servicios críticos están dominados por otros servidores web, como los de las

empresas **Microsoft**, **Google**, **IBM** u **Oracle** que ofrecen un soporte mucho más grande y una mayor robustez.

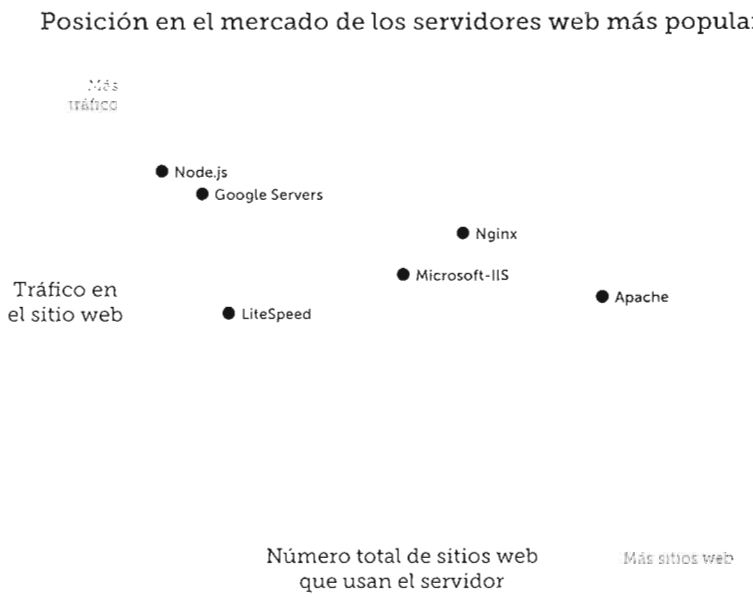


Figura 2.4: Uso de los diferentes servidores web y correlación con el uso de esos servidores según tengan los sitios web más o menos tráfico. Datos de w3techs.com (abril 2015)

Teniendo en cuenta que al final nuestro lenguaje para implementar aplicaciones web será PHP por seguir siendo el más utilizado y uno de los más sencillos de aprender, parece claro que la mejor opción es utilizar Apache.

2.4.2 INSTALACIÓN DE APACHE

2.4.2.1 LICENCIA DE APACHE

Apache Http Server (más conocido simplemente como Apache) es un software de código abierto bajo una licencia de tipo **Apache License,** que es una variante de la licencia **GPL;** lo que significa que, incluso, se puede modificar el código libremente y generar una nueva variante personal de Apache, sin obligación de ceder nuestro código de forma abierta. Aunque en todo momento deberemos de mostrar un aviso indicando que se trata de una variación hecha sobre el software **Apache Http Server** con licencia Apache. En definitiva, la licencia de Apache es una de las más libres que existen.

2.4.2.2 FORMAS DE INSTALAR APACHE

Podemos instalar Apache compilando su código fuente, y así mantener el espíritu de trabajo del software libre, a la vez que intervenimos en todos los aspectos de la instalación, realizándola absolutamente a medida. La otra opción es descargar código ya compilado y ejecutable mediante archivos binarios, que en algunos casos requiere de la ejecución de un software instalador.

Las diferentes posibles instalaciones se pueden examinar en la página oficial de Apache: http://httpd.apache.org.

2.4.2.3 ESTRUCTURA DE DIRECTORIOS DE APACHE

Una instalación típica posee los siguientes subdirectorios dentro del directorio de instalación de Apache.

DIRECTORIO	CONTENIDO / USO
bin	Archivos ejecutables. Contiene todos los programas que tiene Apache para configurar, gestionar, ejecutar o detener el servicio que ofrece.
cgi-bin	Directorio que se usa para almacenar programas del lado del servidor.
conf	Contiene los archivos de configuración de Apache.
error	Archivos que contienen los mensajes de error del servidor (en varios idiomas).
htdocs	Directorio por defecto en el que se guardan las páginas web. Raíz por defecto del sitio web principal que sirve Apache.
icons	Carpeta que contiene los iconos que usa el servidor para mostrar en algunos de sus mensajes.
include	Archivos de cabecera del código fuente de Apache.
lib	Contiene archivos de librería de Apache.
logs	Archivos de información sobre conexiones y errores acaecidos.
manual	Contiene el manual de Apache.
modules	Módulos y extensiones del servidor.

2.4.2.4 INSTALACIÓN BINARIA EN WINDOWS DE APACHE

La fundación Apache ya no distribuye instalaciones binarias para Windows en su página web, solo distribuye el código fuente. Por lo que hay que acudir a terceros para obtener archivos binarios de instalación.

La descarga más recomendada es la que se realiza a través de la fundación **Apache Lounge**, asociación (avalada por Apache) encargada de crear archivos preparados para una instalación de Apache en Windows. La página http://www.apachelounge.com/download/ (véase Figura 2.5) es la que posee tanto los archivos de instalación, como la información necesaria sobre cómo realizar el proceso.

La instalación consiste simplemente en descargar el de tipo ZIP correspondiente a nuestro sistema Windows. Ese archivo contiene la aplicación Apache ya preparada para funcionar. El código binario se ha conseguido compilando el código fuente original en lenguaje C utilizando el entorno **Visual Studio** de Microsoft; la versión concreta del compilador dependerá, a su vez, de la versión de Apache descargada; las últimas versiones utilizan la compilación llamada **VC11**, acrónimo de Visual Studio 2012. El componente Visual C++ Redistributable para Visual Studio, se consigue en la dirección http://www.microsoft.com/en-us/download/details.aspx?id=30679.

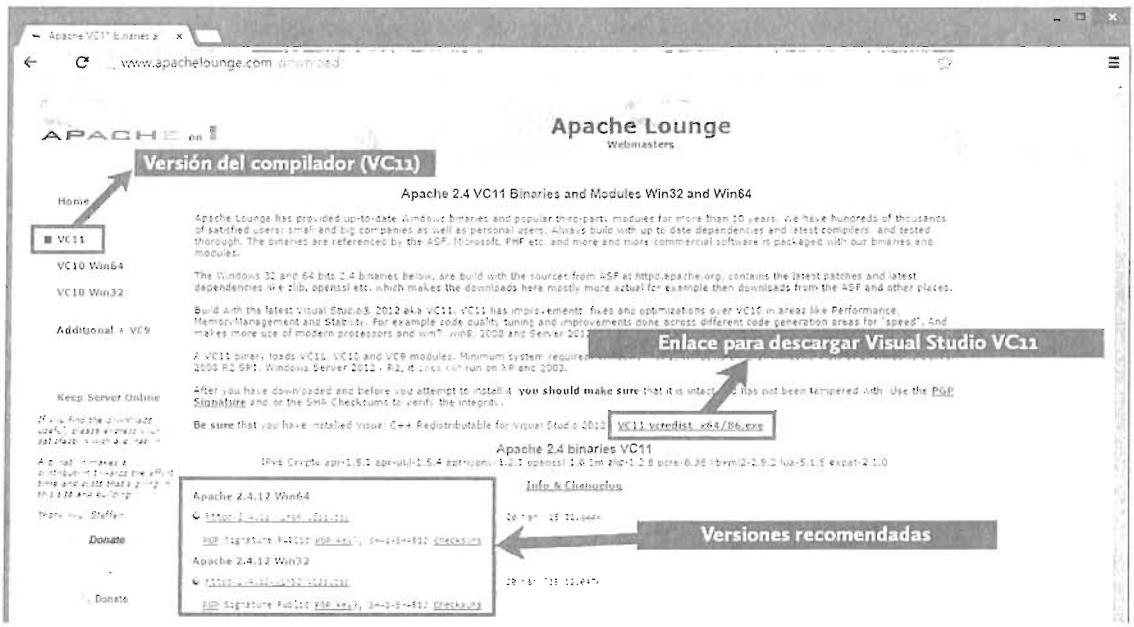


Figura 2.5: Página de descarga de los archivos binarios para Windows en Apache Lounge

ACTIVIDAD 2.1:

- Realiza la Práctica 2.1 *"Instalación binaria de Apache en Windows"*, en la página 90.
- Intenta realizar la misma práctica para un sistema de 32 bits.

2.4.2.5 INSTALACIÓN DE APACHE EN LINUX MEDIANTE CÓDIGO FUENTE

La ventaja de instalar utilizando directamente el código fuente es que no importa bajo qué distribución Linux estemos ya que funciona para todas. Además, las instalaciones por código fuente nos permiten configurar todo el proceso y decidir exactamente dónde y cómo vamos a realizar la instalación. Es la instalación recomendada en entornos de producción. La desventaja es que es una forma de instalar compleja que requiere conocer muy bien tanto el sistema operativo como el funcionamiento del propio servidor Apache.

La instalación por código fuente requiere leer la documentación oficial de Apache sobre la instalación disponible en la dirección http://httpd.apache.org/docs/ en la que se explica la forma de instalar y los problemas habituales que nos podemos encontrar. Aunque hay documentación en español e inglés, la versión en inglés siempre está más al día y es la que, si es posible, deberíamos leer.

Para instalar Apache mediante su código fuente, los pasos principales son:

[1] Descargar el código fuente de la dirección http://httpd.apache.org/download.cgi

[2] Descomprimir los archivos

[3] Instalar librerías y utilidades necesarias. Se requieren las librerías de compilación en lenguaje C y C++, la librería APR (y APR-util) y la librería PCRE

[4] Configurar y compilar el código fuente

[5] Hacer que el directorio **bin** de Apache forme parte del PATH del sistema

[6] Si se desea, configurar Apache como servicio de Linux

ACTIVIDAD 2.2:

- Realiza la Práctica 2.2 *"Instalación de Apache en Linux mediante el código fuente"*, en la página 93
- Intenta realizar esa misma práctica para un Linux tipo Red Hat y para otro tipo Debian

2.4.2.6 INSTALACIÓN EN LINUX MEDIANTE PAQUETES

Indudablemente es la opción favorita para instalar Apache en Linux. Es más fácil, más cómoda y práctica. Tiene la desventaja de que es menos personalizable y que la distribución de directorios se hace sin nuestra intervención; lo que dificulta conocer dónde se instala realmente Apache y cómo es su estructura de directorios.

Instalar Apache por paquetes en sistemas de tipo Red Hat

Actualmente la utilidad más habitual para instalar un paquete en los sistemas Linux de tipo Red Hat es **yum.** Apache se instala mediante el comando:

```
# yum install httpd
```

También podemos instalar el paquete desde el entorno gráfico del sistema.

En cualquier caso, tras la instalación todo estará configurado directamente para utilizar ya Apache. Los comandos para lanzar y detener Apache son los mismos (***httpd -k start*** y ***httpd -k stop***), la instalación varía en cuanto a los directorios de instalación. Así:

- La raíz de instalación de Apache es **/etc/httpd**. Dentro de ese directorio se encuentran los directorios: **conf**, **logs**, **modules** y **run**.
- Los archivos de configuración (concretamente el archivo **httpd.conf**) está en el directorio **/etc/httpd/conf.**
- En el directorio **/var/www** se encuentran los directorios **cgi-bin**, **error**, **icons** y especialmente el directorio **html** que es la raíz por defecto de los documentos web. Es decir, cuando escribimos en el navegador **http://localhost** se intentará mostrar el contenido del archivo */var/ww/html/index.html*.
- **/usr/sbin** contiene los ejecutables. Esa carpeta forma parte del **PATH** del sistema.

Instalar Apache por paquetes en sistemas de tipo Debian

En el caso de Ubuntu y el resto de sistemas Debian, se usa actualmente el comando **apt-get** desde la línea de comandos. Sería:

```
apt-get install apache2
```

Eso mismo se puede hacer desde el gestor gráfico de paquetes (por ejemplo **Synaptic**). En este caso los directorios involucrados son:

- **/etc/apache2** es el directorio raíz de instalación de Apache.
- **/etc/apache2/apache2.conf** es el archivo de configuración (que dentro hará referencia a **httpd.conf**).
- **/usr/sbin** contiene los ejecutables, esa carpeta forma parte del **PATH** del sistema.
- No hay archivo **httpd**, en su lugar se gestiona Apache con el comando **apachectl, que** equivalente a **httpd -k.**
- La raíz de documentos, el equivalente al directorio **htdocs** en Windows, es el directorio **/var/www**. Es decir, cuando escribimos en el navegador **http://localhost** se intentará mostrar el contenido del archivo */var/ww/index.html.*

Aunque la gestión es sencilla (quizá más que en las otras instalaciones) sus importantes diferencias con las otras instalaciones, sobre todo por la diferencia de sus rutas de instalación, son una desventaja para el aprendizaje.

2.4.3 INICIAR Y PARAR LA EJECUCIÓN DEL SERVIDOR WEB APACHE

2.4.3.1 WINDOWS

Apache se instala como servicio de Windows. El arranque y la parada por tanto se pueden realizar desde la pantalla de servicios o bien, desde un icono que Apache instala en la barra de tareas al lado del reloj, llamado **Apache Monitor**. Este icono se obtiene del archivo del mismo nombre situado en el directorio **bin** de Apache.

Desde el símbolo de sistema de Windows se pueden también hacer estas tareas mediante el programa que controla al servidor Apache, que se suele llamar **httpd**. Si hemos configurado correctamente la variable PATH del sistema, tendremos acceso a ***httpd*** desde el terminal de comandos de Windows. Abriendo el terminal con privilegios administrativos podremos:

- Instalar el servidor Apache como servicio en Windows, comando: `httpd -k install`
- Arrancar el servidor Apache: `httpd -k start` o con `net start apache2.4`, si apache2.4 es el nombre que Windows asigna al servicio de Apache

- Detener el servidor Apache: `httpd -k stop` o con `net stop apache2.4`
- Reiniciar: `httpd -k restart`
- Desinstalar el servicio: `httpd -k uninstall` o con el comando Windows: `sc delete apache2.4`

2.4.3.2 LINUX

Los comandos son los mismos que en Windows, salvo que en Linux no hace falta instalar Apache como servicio, aunque se puede hacer. Los comandos relacionados con la ejecución son (en algunas distribuciones Linux, en lugar de ***httpd -k***, se utiliza ***apachectl***):

- `httpd -k start`. Inicia el proceso ***httpd***; es decir ejecuta el servidor Apache.
- `httpd -k stop`. Detiene el servidor.
- `httpd -k restart`. Reinicia el servidor.
- `httpd` sin parámetros, lanza el servidor
- `cat /raízApache/logs/httpd.pid | xargs kill`. Elimina el proceso cuyo identificador (**pid**) se encuentra en el archivo ***httpd.pid*** (dentro de la carpeta **log** de Apache); es decir, elimina el proceso servidor de Apache. El archivo **httpd.pid** se crea cada vez que se ejecuta Apache, precisamente para grabar el identificador de proceso.

  A veces conviene realizar esta operación porque solo con detener Apache no basta para realmente pararle.

2.4.3.3 OPCIONES DEL COMANDO HTTPD

El programa que permite indicar un servidor Apache es **httpd**. Algunas instalaciones usan, en lugar de httpd, **apachectl** o **apache2** aunque no tienen exactamente las mismas opciones. Al ejecutar httpd se pueden utilizar estos parámetros:

OPCIÓN	SIGNIFICADO
-k start	Lanza el servidor Apache
-k stop	Para el servidor Apache
-k restart	Reinicia el servidor Apache
-D nombre	Define un nombre para las directivas **<IfDefine name>**
-d directorio	Permite indicar un directorio raíz alternativo para Apache
-f rutaArchivo	Permite indicar un archivo alternativo de configuración
-C "directiva"	Procesa la directiva indicada antes de leer la configuración
-c "directiva"	Procesa la directiva indicada después de leer la configuración
-v	Muestra la versión de Apache
-V	Muestra las opciones de compilación

OPCIÓN	SIGNIFICADO
-h	Ayuda para conocer las opciones de httpd
-l	Lista de módulos compilados
-L	Lista de directivas
-t	Ejecuta el analizador de sintaxis para los archivos de configuración de Apache
-T	Igual pero no comprueba la sintaxis

La ubicación de **httpd** depende de la instalación (como se ha comentado anteriormente); lo más habitual es que se encuentre en la carpeta **bin** dentro de la raíz de instalación de Apache.

2.4.4 FUNCIONAMIENTO DE LAS RUTAS EN UN SERVIDOR WEB APACHE

Salvo que modifiquemos la configuración de Apache, la carpeta por defecto donde alojar las páginas del sitio web es **htdocs** en la raíz de instalación de Apache o bien **www** (en muchas instalaciones Linux suele ser **/var/www/html**). A ese directorio se le llama **raíz de documentos** (***DocumentRoot***) y, por defecto, es la ubicación física de la raíz del sitio web.

Así, por ejemplo, si nuestra raíz de documentos es ***/var/www/html***, en esta tabla se expresa la relación entre la dirección que el usuario escribiría en el navegador y su ruta en disco:

URL EN EL NAVEGADOR	RUTA DEL ARCHIVO QUE SE MUESTRA EN EL NAVEGADOR
http://nombreServidor	/var/www/html/index.html
http://nombreServidor/pagina3.html	/var/www/html/pagina3.html
http://nombreServidor/documentos	/var/www/html/documentos/index.html
http://nombreServidor/documentos/opciones.html	/var/www/html/documentos/opciones.html
http://nombreServidor/documentos/txt/texto1.html	/var/www/html/documentos/txt/texto1.html

2.4.5 FUNCIONAMIENTO DE LA CONFIGURACIÓN DEL SERVIDOR APACHE

2.4.5.1 INTRODUCCIÓN

Para modificar el funcionamiento de Apache, se utilizan sus archivos de configuración. El principal es **httpd.conf** (en algunas instalaciones es **apache2.conf**) que se encuentra, normalmente, en el directorio **conf** de la instalación de **Apache**.

Los cambios en la configuración de apache no estarán disponibles hasta que reiniciemos e,l propio apache (por ejemplo con `httpd -k restart`).

Para indicar un archivo de configuración personal, distinto del archivo por defecto, se ejecuta el comando:

```
httpd -f rutaAlNuevoArchivo
```

Los archivos de configuración están compuestos de los siguientes elementos:

- **Directivas**. Se trata de una palabra clave a la que se sigue un valor. Por ejemplo:

  ```
  Listen 80
  ```

  Indica que el servidor Apache usará el puerto 80 para comunicarse.

  La lista completa de directivas se puede mostrar desde la línea de comandos con el comando `httpd -L`

- **Secciones o contenedores**. Permiten dividir el documento y, así, conseguir que las directivas solo se apliquen a una parte del documento. Las secciones se configuran mediante etiquetas al estilo del lenguaje XML (aunque no son realmente XML). Por ejemplo:

  ```
  <Directory /usr/local/apache/htdocs/dir2>
     Deny from all
     Allow from 192.168.4.21
  </Directory>
  ```

  En este caso se nos indica que el directorio */usr/local/apache/htdocs/dir2* estará protegido ante cualquier acceso salvo a las conexiones desde la dirección IP *192.168.4.21*.

- **Comentarios**. Se trata de un texto que comienza con el signo #. Sirven para documentar el archivo de configuración.

2.4.5.2 ARCHIVOS .htaccess

Los archivos ***htaccess*** son archivos de configuración, se colocan en un directorio. Tienen las mismas posibilidades que el archivo **httpd.conf**, pero su configuración solo se aplica al directorio en el que está ubicado el archivo.

Por ejemplo si colocamos un archivo **.htaccess** en un directorio concreto y le añadimos la directiva: `DirectoryIndex indice.html`, conseguiremos que la página de índice de ese directorio ya no será la típica ***index.html*** sino la página ***indice.html***. Es decir, hemos aplicado una configuración para ese directorio concreto.

Por defecto, Apache no tiene en cuenta el contenido de los archivos .htaccess. Para que sí se tenga en cuenta, se debe utilizar la directiva **AllowOverride**, por ejemplo, con el valor **all**. Esta directiva se explica más adelante.

2.4.5.3 CONTENEDORES O SECCIONES

Como se ha comentado anteriormente, Apache utiliza en los archivos de configuración una serie de contenedores o secciones, de modo que las directivas que se coloquen dentro de un contenedor se aplican solo a los elementos a los que se refiere dicho contenedor.

Por ejemplo, en un contenedor de directorio, las directivas dentro de ese contenedor, servirán para modificar el funcionamiento de dicho directorio.

La sintaxis de una sección o contenedor es:

```
<nombreSección argumentos>
     directivas...
</nombreSección>
```

Lista de secciones o contenedores habituales:

- **Directory**. Permite establecer una configuración para un directorio concreto. Ejemplo:

```
<Directory "C:\www2">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

- **Files**. Establece directivas para los archivos con un nombre concreto. Ejemplo:

```
<Files privado.html>
    Require all denied
</Files>
```

- **DirectoryMatch**. Aplica una configuración a varios directorios a la vez mediante una expresión regular.
- **FilesMatch**. Permite establecer una expresión regular para que la sección se refiera a un grupo de archivos.
- **Location**. Las directivas de esta sección se referirán a una URL. Ejemplo:

```
<Location /privado/>
    Require all denied
</Location>
```

  En el ejemplo, la dirección ***http://localhost/privado*** (si el servidor es local) no será accesible.

- **LocationMatch**. Igual que la anterior, pero permite especificar varias URL mediante una expresión regular.

- **VirtualHost**. Permite configurar un servidor virtual, de modo que parezca que disponemos de varios servidores web en una única instalación. Para ello, se indica en la sección como argumentos, un nombre de servidor (o su dirección IP) y el puerto por el que atenderá. Ejemplo:

```
<VirtualHost localhost:12000>
        DocumentRoot "c:\www2"
        DirectoryIndex index.html
</VirtualHost>
```

  Mediante la dirección **localhost:12000** conseguiremos tener un segundo servidor Para ello tendremos que hacer que Apache escuche por el puerto 12000 y además permitir el acceso a ese directorio con una directiva **Directory**. Más adelante se explica en detalle la creación de servidores virtuales.

- **IfDefine**. Directivas a ejecutar si se pasa al servidor Apache un comando concreto durante su ejecución.

- **IFModule**. Directivas a ejecutar si Apache ha cargado un módulo concreto.

2.4.6 PRINCIPALES DIRECTIVAS DE APACHE

Las directivas se pueden poner dentro de una sección o bien fuera de cualquier sección, lo que las convierte en globales. Muchas directivas solo pueden ser globales. Se comentan, brevemente, a continuación las directivas más importantes.

2.4.6.1 DIRECTIVAS DE CONFIGURACIÓN GENERAL DEL SERVIDOR

- **ServerName**. Nombre del servidor (para servidor local se utiliza **localhost**).

- **ServerAlias**. Solo funciona dentro de directivas **VirtualHost** y permiten establecer nombres alternativos (alias) al servidor virtual.

- **ServerRoot**. Indica la raíz de la instalación de Apache. La raíz, al menos, debe contener las carpetas **conf** y **log**.

- **Listen**. Permite modificar el puerto por el que se comunica el servidor. Podemos incluso escuchar por más de un puerto, lo cual se consigue de esta forma:

```
Listen 80
Listen 8080
```

- **DocumentRoot**. Indica la ruta raíz de los documentos de Apache. Normalmente es la ruta al directorio **htdocs**.

- **DirectoryIndex**. Indica el nombre del archivo índice del directorio. Normalmente es **index.html**, pero se puede cambiar y, además, indicar más de un archivo. Por ejemplo:

```
DirectoryIndex index.php index.html index.txt
```

En el ejemplo, se ha establecido como archivo índice a **index.php**, si no existe se buscará **index.html** y si tampoco existe a **index.txt**. Si no hay ninguno de esos archivos, entonces se mostrará el contenido el directorio o un error (dependiendo del estado de la directiva **Indexes** que se comenta más adelante).

Se pueden establecer diferentes nombres de archivos de índice para cada directorio si utilizamos esta directiva dentro de una sección **Directory**.

- **ErrorDocument**. Permite indicar qué mostrar cuando ocurra un error. A esta directiva le sigue el número de error http y luego el texto o documento a mostrar en caso de error. Por ejemplo, podemos establecer que se muestre una página en caso de error por archivo no encontrado:

```
ErrorDocument 404 /error404.html
```

- **ServerAdmin**. Establece el email del administrador del sistema, que se utilizará para que el propio servidor web envíe un mensaje en caso de problemas. Deberemos tener configurado en el sistema un servidor SMTP para que el envío de mensajes funcione.

2.4.6.2 CONFIGURACIÓN DE LOS ARCHIVOS LOG

- **ErrorLog**. Ruta al archivo LOG de errores que permite examinar los problemas acaecidos en el servidor Apache.

```
ErrorLog "logs/errorlog"
```

- **LogLevel**. Permite indicar qué eventos se almacenan en el archivo LOG de errores. Se basa en un sistema de niveles, de modo que cuanto más alto sea el nivel, menos eventos se almacenen en los log.

  La lista de niveles, en orden de más a menos nivel, es:

  - **emerg**. Solo almacena eventos que impiden el desarrollo del servidor.
  - **alert**. Eventos que requieren tomar acciones inmediatamente.
  - **crit**. Eventos de fallos críticos.
  - **error**. Condiciones de error.
  - **warn**. Avisos, no significan errores, solo advertencias de posibles errores.
  - **notice**. Cualquier evento que tenga cierta significancia. Si se elige este nivel, se almacena cualquier evento de los anteriores.

2.4.6.3 DIRECTIVAS PARA RENOMBRAR Y REDIRECCIONAR RECURSOS

- **Alias**. Permite establecer un alias para una dirección de modo que, por ejemplo, si hemos escrito en el navegador la dirección: ***http://localhost/departamentoVentas*** podamos conseguir que en lugar de entrar en el directorio ***departamentoVentas*** dentro de la raíz de

documentos, en su lugar podamos mostrar el contenido de otro directorio, que puede residir incluso fuera del directorio de documentos por defecto de Apache. Ejemplo:

```
Alias /departamentoVentas e:\otros\ventas
```

Cuando queramos ir a la ruta */departamentoVentas*, mostraremos el contenido del directorio (Windows) *e:\otros\ventas*.

- **Redirect**. Parecida a la anterior, pero en lugar de asociar un directorio a una ruta URL, se cambia automáticamente una ruta por otra. Ejemplo:

```
Redirect /usuarios/jorge http://www.jorgesanchez.net
```

Admite indicar si el salto es permanente (**permanent**) o temporal (**temporary**). Este texto se escribe antes de la ruta al directorio (es decir, justo tras la palabra **Redirect**). Por defecto el valor es ***temporary***. La diferencia está en el código que se envía al navegador (en el primer caso un código 303 y en el segundo un código número 410)

2.4.6.4 DIRECTIVAS DE PERMISO DE ACCESO

- **Allow**. Solo se admite en secciones **Directory** o dentro de archivos **.htaccess**. Permite indicar desde que host se permite el acceso al directorio. El formato es:

```
Allow from host
```

El host es un serie de nombres de dominio (o partes de dicho nombre) o de direcciones IP (sean v4 o v6) o especificaciones *CIDR* de red (como por ejemplo 10.10.0.0/16) con las que se especifica el servidor, servidores o rutas de red a las que permitimos el acceso al directorio.

Si se indica como host la palabra ***all*** entonces se permite el uso a cualquier host.

- **Deny**. Funciona como la anterior, pero ahora especifica a qué máquinas prohibimos el acceso al directorio.
- **Order**. Especifica el orden en el que Apache tiene que aplicar las directivas anteriores. Ejemplo (que es el que funciona por defecto):

```
Order Deny,Allow
```

En el ejemplo primero se aplica la directiva **Deny** y luego **Allow** (que es la que tendrá prioridad).

- **Require**. Permite especificar qué usuarios o grupos de usuarios pueden acceder a un directorio. Admite como valores:
  - **user**. Permite indicar una lista de usuarios. Será a los usuarios de esa lista a los que se permita el acceso.
  - **valid-user**. Permite el acceso a cualquier usuario válido.

- **group** . Permite indicar una lista de grupos. Será a los usuarios que pertenezcan a uno de esos grupos a los que se permitirá el acceso.
- **all denied.** Ningún usuario podrá acceder al directorio.
- **all granted.** Todos los usuarios tendrá permitido el acceso.

- **Satisfy**. Especifica el funcionamiento de las directivas **Allow** y **Require** anteriores. Sus posibles valores son:
  - **All.** Valor por defecto, indica que el cliente que ha hecho la petición tiene que cumplir tanto la directiva **Allow** como la **Require** para acceder al directorio.
  - **Any**. Al cliente le basta con cumplir cualquiera de las dos directivas
- **AuthType**. Indica la forma de autentificar a los usuarios, en el caso de restringir el acceso a un directorio en base a usuarios o grupos. Se suele usar la opción **Basic**, autentificación simple, pero es posible indicar **Digest** (autentificación avanzada).
- **AuthName**. Indica el texto que aparecerá en el cuadro que pida el usuario y contraseña al usuario para su autentificación.
- **AuthBasicProvider**. Indica la forma de autentificar el usuario. El valor por defecto es **file**, es decir, mediante un archivo de contraseñas. Sin embargo, se puede autentificar mediante otros métodos (una base de datos, un directorio LDAP...), lo cual requerirá instalar los módulos adecuados.
- **AuthUserFile**. Indica la ruta al archivo de contraseñas, encargado de autorizar los accesos a un directorio.

2.4.6.5 DIRECTIVAS PARA DIRECTORIOS

- **Options.** Aunque está pensada para directorios, se admite en cualquier contexto (servidor completo, sección de servidor virtual...). Indica qué opciones están disponibles para un directorio particular. Posibilidades (se pueden indicar varias):
  - **All**. Todas.
  - **ExecCGI**. Se admite la ejecución de scripts CGI.
  - **FollowSymLinks.** Solo funciona en secciones **Directory** y en archivos **.htaccess.** Permite el uso de enlaces simbólicos en el directorio.
  - **Indexes.** Hace que cuando se use la URL en el servidor hacia un directorio sin página índice, se muestre un listado del directorio.
  - **SymLinksIfOwnerMatch.** Sigue los enlaces simbólicos del directorio solo si pertenecen al usuario que hizo la petición.

  Además como los directorios heredan las propiedades de su directorio padre, se pueden indicar signos + o - en las opciones para que se añadan o eliminen las opciones. Por ejemplo:

```
Options -Indexes
```

Hace que se acepten las opciones heredadas, pero no funcione la de listar el directorio en caso de que no haya índice.

- **AllowOverride**. Solo se puede usar en secciones **Directory** indica qué aspectos del archivo **.htaccess** que se indique en el directorio serán respetados. Posibilidades (se puede indicar más de una, excepto si se indica el valor **All**):
  - **All**. Todo el contenido del archivo **.htaccess** se tendrá en cuenta.
  - **AuthConfig**. Se admite del archivo **.htaccess** las directivas referidas a autentificación.
  - **FileInfo**. Permite el uso de directivas de archivos.
  - **Indexes**. Permite el uso de directivas de control de la página índice.
  - **Limit**. Permite el uso de las directivas de acceso (**Allow**, **Deny** y **Order**).
  - **Options**. Permite el uso de la directiva **Options**.
  - **None**. No se tiene en cuenta el contenido.

2.4.7 TAREAS HABITUALES DE CONFIGURACIÓN DE APACHE

2.4.7.1 CAMBIAR EL PUERTO DE ESCUCHA

Se hace con la directiva **Listen** (véase 2.4.6.1 *"Directivas de configuración general del servidor"*, en la página 61)

2.4.7.2 INDICAR LA UBICACIÓN DEL DIRECTORIO RAÍZ DE APACHE

Se utiliza cuando queremos cambiar de ubicación en disco de Apache. Es el caso de querer utilizar un directorio distinto al predefinido *C:\Apache24* de la instalación tipo ZIP de Windows. Si le queremos en otro sitio habrá que modificar la directiva **ServerRoot** (véase 2.4.6.1 *"Directivas de configuración general del servidor"*, en la página 61) de esta forma, por ejemplo:

```
ServerRoot "C:/Apache"
```

Hay que tener en cuenta que un cambio en el archivo de configuración de esta índole, puede implicar cambiar más lineas, ya que puede haber muchas directivas y secciones que hagan referencia al directorio anterior (por ejemplo a *C:/Apache24*), por lo que habrá que reemplazar cada referencia por la nueva raíz.

2.4.7.3 CAMBIAR EL NOMBRE DE LA PÁGINA ÍNDICE

Se encarga de ello la directiva **DirectoryIndex** (véase 2.4.6.1 *"Directivas de configuración general del servidor"*, en la página 61)

2.4.7.4 MODIFICAR LA UBICACIÓN DE LA RAÍZ DE LOS DOCUMENTOS

Normalmente la raíz del servidor web Apache es el directorio **htdocs** (que, a su vez, se encuentra dentro de la raíz de instalación de Apache). Si deseamos indicar otro directorio como raíz del sitio, en ese caso utilizaremos la directiva **DocumentRoot** (véase 2.4.6.1 *"Directivas de configuración general del servidor"*, en la página 61).

2.4.7.5 MODIFICAR PÁGINAS DE ERROR

El protocolo http devuelve códigos de estado y error al alcanzar un recurso. Lógicamente son considerados errores cuando al acceder al recurso hay problemas: porque no se encuentra o no se ha obtenido correctamente. Los códigos más habituales son:

CÓDIGO	TIPO	SIGNIFICADO
200	**Petición correcta**	OK, se accedió sin problemas al recurso
201		Petición http realizada y provocó un nuevo recurso creado
202		Se aceptado la petición pero ha habido problemas con ella
203		Información no autoritativa
204		Sin contenido
205		Recargar contenido
206		Solo se envió contenido parcial del recurso solicitado
300	**Redirección**	Múltiples opciones. Lista con direcciones que el usuario podría seguir
301		Recurso solicitado movido permanentemente, además se indica la nueva dirección del recurso
302		Igual que al anterior, pero referido a que el recurso se ha movido temporalmente
303		Sugerencia de otra dirección
304		Indicación de que el recurso no se ha modificado desde la última vez que se accedió a él
305		Indicación de que se debe utilizar un proxy
403	**Errores de cliente**	Prohibido el acceso. En este caso el recurso no está accesible ni aun cuando nos autentifiquemos
404		Recurso no encontrado. Es el error más habitual
405		Método de acceso no permitido
406		No se acepta los datos procedentes del usuario (normalmente procedentes de un formulario
407		Autentificación proxy requerida

CÓDIGO	TIPO	SIGNIFICADO
408	**Errores de cliente**	Tiempo de espera agotado
409		Conflicto con el recurso solicitado
410		Recurso no disponible permanentemente
500		Error interno del servidor
501	**Error de servidor**	No está implementado el servicio solicitado en el servidor
502		Pasarela incorrecta
503		Servicio no disponible
504		Tiempo de espera de pasarela agotado
505		Versión de http no soportada

Las páginas de error que devuelven estos códigos son las que Apache tiene configuradas por defecto. Para cambiarlas se usa la directiva **ErrorDocumentation** en la cual se indica el código de error y la URL a la página que mostraremos cuando se produzca ese error.

2.4.7.6 REDIRECCIONAR UN RECURSO

Se trata de conseguir que cuando el usuario acceda a un recurso, el servidor Apache le lleve a otra dirección.

Hay dos posibilidades:

- Directiva **Alias** (véase 2.4.6.3 *"Directivas para renombrar y redireccionar recursos"*, en la página 62). La cual hace que cuando se usa una determinada ruta, en realidad Apache cargue los recursos de una dirección local concreta.
- Directiva **Redirect** (véase 2.4.6.3 *"Directivas para renombrar y redireccionar recursos"*, en la página 62). La cual realiza una redirección completa. Es decir, se intenta acceder a un recurso y Apache redirige la petición a otra dirección (que normalmente es una URL en otro servidor).

2.4.7.7 OCULTAR LISTADO DE DIRECTORIOS

De manera predeterminada, cuando navegamos hacia una ruta como ***www.miserver.com/docs***, correspondiente a un servidor Apache, se buscará si existe una página índice (normalmente ***index.html***) en el directorio ***docs***. Si no hay índice, Apache muestra el listado de archivos que tiene el directorio.

Esta opción no suele ser deseable, por permitir el acceso indiscriminado a todo el contenido del directorio. La razón es que, en el archivo de configuración, suele haber una directiva `Options Indexes FollowSymLinks` dentro de la sección `Directory` del directorio raíz de documentos.

```
<Directory /var/www>
 Options -Indexes FollowSymLynks
</Directory>
```

El código anterior modifica la directiva para la raíz de documentos del servidor web, que suponemos situada en */var/www*.

Otra opción es crear un archivo **.htaccess** en el directorio que deseamos proteger. Después dentro de ese archivo se indica la directiva: `Options -Indexes`.

Si lo que deseamos es que ningún directorio tenga la posibilidad de mostrar el listado de su contenido, entonces deberemos observar su configuración predeterminada, que suele ser (suponemos ahora que la raíz de documentos es *c:\Apache24\htdocs*):

```
DocumentRoot "c:/Apache24/htdocs"
<Directory "c:/Apache24/htdocs">
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

Simplemente necesitaríamos quitar la palabra **Indexes**. El resto de opciones por defecto se refieren que no permitimos el uso de archivos **.htaccess** y que a todo el mundo tendrá acceso a los directorios.

2.4.7.8 PERMISOS DE ACCESO

Consiguen que un directorio quede inaccesible o no, dependiendo de varias condiciones. Las directivas relacionadas están descritas en el Apartado 2.4.6.1 *"Directivas de configuración general del servidor"*, en la página 61. Normalmente, por defecto todo el mundo tiene acceso a todos los directorios.

2.4.7.9 AUTENTIFICACIÓN

Una de las opciones habituales de uso en Apache es el acceso a un recurso (como un directorio) para usuarios autentificados. Lo habitual es utilizar un archivo **.htaccess** en el directorio que contenga la configuración de acceso que estimemos conveniente.

Hay que recordar que necesitamos que la directiva **AllowOverride** esté al menos con el valor **AuthConfig** para que se tengan en cuenta las directivas de autentificación en los archivos **.htaccess**. **AllowOverride** se debe aplicar al directorio que contenga el .htaccess, o bien se puede haber aplicado a directorios superiores (incluso al raíz) y, por herencia, la directiva funcionará para todos los demás directorios.

Otra posibilidad es indicar las directivas directamente dentro del archivo de configuración general de Apache a través de una sección **Directory**; aunque, lo cierto es que es más difícil de mantener, especialmente si cambiamos de servidor.

Debemos de crear el archivo de contraseñas fuera de la ruta del servidor Apache, de otro modo podría quedar expuesto a los usuarios de nuestro sitio web y que alguien hiciera un uso no autorizado del mismo. La utilidad **htpasswd** que forma parte de las herramientas de Apache (estará en el directorio **bin** de Apache o en el directorio **/usr/sbin** en Linux) crea un archivo de texto pero con las contraseñas cifradas.

La creación del archivo se hace con:

```
htpasswd -c rutaArchivoContra usuario
```

Ejemplo:

```
htpasswd -c c:\passwords\pass1 alonso
```

Se utiliza el parámetro -c para crear el archivo por primera vez (si el archivo ya existiera, con este parámetro le eliminaremos); para añadir el resto de usuarios no se usa. Es decir, si añadimos el segundo usuario:

```
htpasswd c:\passwords\pass1 guzman
```

Tras cada comando **htpasswd** se nos pedirá escribir y confirmar la contraseña del usuario que estamos añadiendo al archivo. Esa contraseña queda cifrada en el archivo de texto ***pass1***. Así iremos creando las contraseñas de todos los usuarios que queramos. Así el archivo de contraseñas para los dos usuarios creados podría ser:

```
alonso:$apr1$0.cDnGTc$NJrqioRdYayfqiQoGp4yw1
guzman:$apr1$Ra5.q5YP$krQ7WzR9SENsgUefNyTCf/
```

Cuando un directorio queramos que quede accesible a solo un usuario, en el archivo **.htaccess** de dicho directorio se indican las siguientes directivas:

```
AuthType Basic
AuthName "Acceso Privado"
AuthBasicProvider file
AuthUserFile c:\passwords\pass1
Require user alonso
```

De esta forma al entrar en el directorio en el que está situado este archivo **.htaccess** se pedirá el usuario y contraseña, se cotejará con el contenido del archivo ***c:\passwords\pass1*** y solo se permitirá el acceso al directorio si somos el usuario ***alonso*** y hemos escrito correctamente su contraseña.

Si deseáramos permitir el acceso a más de un usuario la línea del **Require** sería:

```
Require user alonso guzman
```

Si queremos permitir que pueda entrar cualquier usuario presente en el archivo, se haría así:

```
Require valid-user
```

Grupos

Es posible indicar grupos de usuarios. Para ello se crea un archivo de grupo, el cual es un archivo de texto.

Formato del archivo de texto:

```
grupo1: usuario1 usuario2 usuario3 ...
grupo2: usuario4 usuario5 ...
...
```

Ejemplo de contenido de archivo de grupo:

```
editores: alonso guzman
administradores: sofia
```

Para cargar ese archivo de grupo se debe indicar la directiva:

```
AuthGroupFile c:\passwords\grupos
```

Para conseguir que un directorio solo permita el acceso a los usuarios de un grupo concreto (se podría indicar más de un grupo separados por espacios):

```
Require group editores
```

2.5 INSTALACIÓN Y CONFIGURACIÓN DE PHP PARA APACHE

2.5.1 ¿QUÉ ES PHP?

Las siglas PHP significan: ***Hipertext Pre Processor*** y se trata del **lenguaje de scripts de servidor** más popular. Un lenguaje de scripts, en general, es un lenguaje cuyo código se incrusta dentro de otro. Es el caso de **JavaScript** que es código que se incrusta dentro del código HTML de una página web. Pero, en el caso de JavaScript, está en el lado del cliente; es decir, es el navegador de Internet el que tiene que interpretar el código del lenguaje script.

Eso provoca una desventaja: los navegadores tienen que tener la capacidad de traducir ese lenguaje. En el caso de JavaScript se asume que cualquier navegador es capaz de traducirle, pero cualquier nuevo lenguaje script que se creara necesitaría añadir esa capacidad al navegador.

En los lenguajes script del lado del servidor, lenguaje script incrustado en el código HTML se interpreta en el propio servidor, que devuelve al navegador el resultado de interpretar dicho código, que siempre será un documento HTML entendible por el navegador.

PHP es gratuito y es software de código abierto que tiene una relación excelente con Apache, MySQL y Linux; aunque actualmente en Windows también se instala muchísimo ya que se puede instalar también en servidores IIS de Microsoft y en otros muchos. Además puede conectar a sistemas de bases de datos como Oracle, Informix, DB2...

2.5.2 INSTALACIÓN DE PHP

2.5.2.1 INSTALACIÓN EN WINDOWS MEDIANTE ARCHIVOS BINARIOS COMPRIMIDOS

Es la instalación recomendada en Windows. Hay posibilidades de instalar mediante instalador, pero es más versátil la instalación mediante un binario ZIP. Además es la opción compatible con la instalación de Apache mediante archivos binarios, explicada en el Apartado 2.4.2.3 *"Estructura de directorios de Apache"*, en la página 53.

Para realizar la descarga del archivo binario correspondiente, hay que ir a la dirección http://windows.php.net/download/ desde la que debemos leer la información sobre la instalación de PHP en Windows para el servidor Apache (Figura 2.6) a fin de conocer la versión a descargar para nuestro sistema.

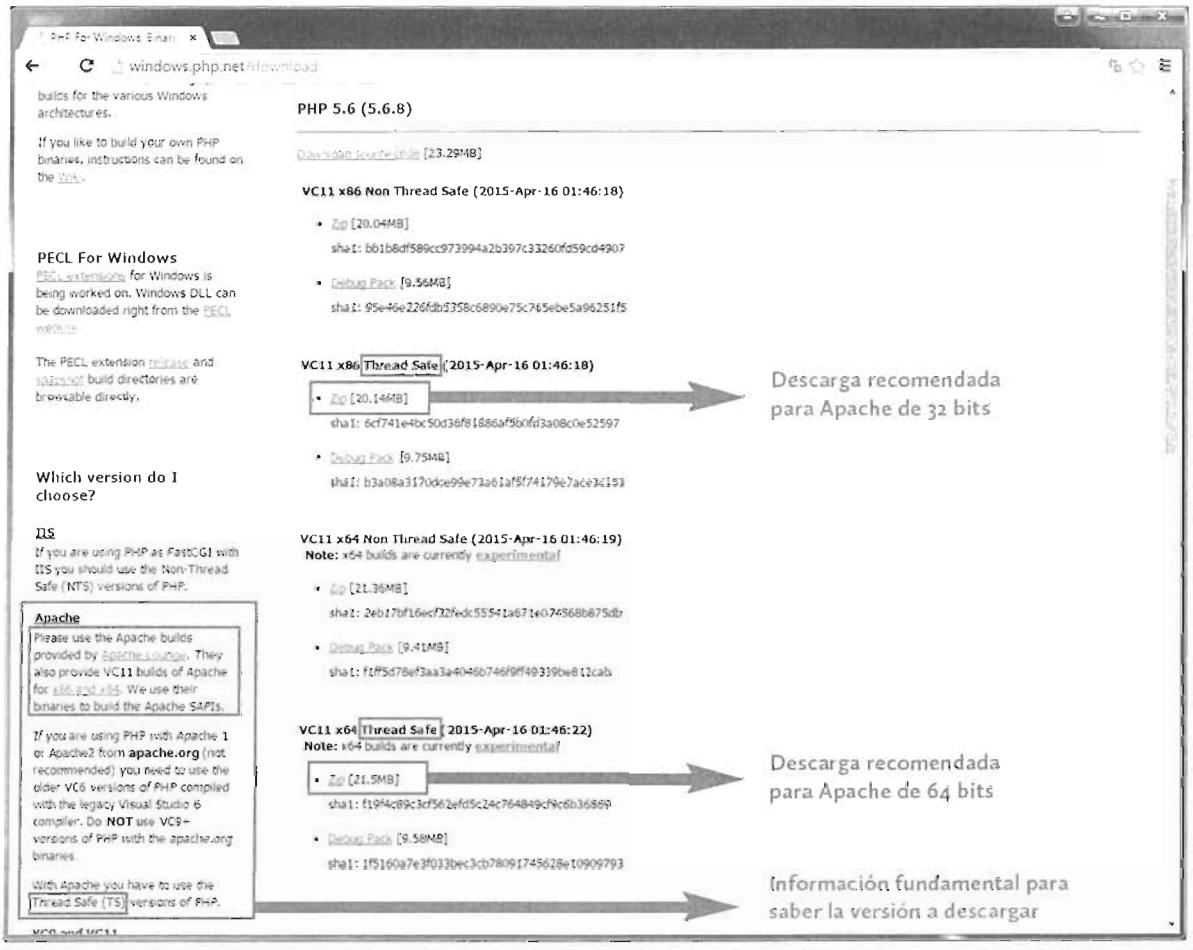


Figura 2.6: Página de descarga de PHP: windows.php.net/download en la que se ha destacado la información que debemos leer para saber qué versión descargar para nuestro servidor Apache en Windows

Para un entorno Apache en Windows, hay que descargar la versión **Thread Safe**, ya que Windows, que es un entorno multihilo (***thread*** significa hilo). El modo **Non-Thread Safe** es el que se debe escoger en el caso de usar PHP en servidores **IIS** de Microsoft porque tienen un modo de ejecución distinto.

Además hay que tener instalado el **Visual C++ Redistributable para Visual Studio 2012**, si descargamos la versión última de PHP (que es lo lógico) para VC11. La descarga e instalación de Visual C++ Redistributable se explicó en la página 90. La versión de este componente a instalar, debe de ser compatible con la versión instalada de Apache. Lo normal es que tengamos Apache para la versión VC11.

Una vez descargado el archivo correspondiente a nuestro sistema, tendremos que:

- Descomprimir el archivo ZIP en la ruta deseada.
- Añadir el directorio raíz de **PHP** y el directorio **ext** (que estará dentro del anterior) al PATH del sistema
- Generar un archivo inicial de configuración php.ini. Para ello en la raíz de PHP se suministran dos archivos:
  - **php.ini-development**. Versión de php.ini pensada para entornos de desarrollo.
  - **php.ini-production**. Versión de php.ini pensada para producción.

  Bastará con copiar el que nos interese de los dos y renombrar la copia como **php.ini**.
- Añadir las extensiones deseadas al archivo php.ini

```
; Muestra errores
display_errors = On

;Especificar la ruta de las extensiones de PHP
extension_dir = "./ext"

;Habilitar el soporte de MySQL
extension=php_mysql.dll
extension=php_mysqli.dll
```

- En el archivo de configuración de Apache (httpd.conf) añadir el módulo de PHP y el tipo apropiado para manejar archivos PHP (suponiendo que PHp está instalado en c:\php):

```
#indicación de la raíz de PHP
PHPIniDir "C:/php"
#carga del módulo de PHP
LoadModule  php5_module "c:/php/php5apache2_4.dll"
#Indicación del tipo MIME apropiado para manejar archivos PHP
<FilesMatch \.php$>
      SetHandler application/x-httpd-php
 </FilesMatch>
```

La comprobación de que la instalación es correcta pasa por crear una página con código PHP y, con el servidor Apache relanzado, cargarla en el navegador. Lo típico es crear una página que invoque a la función **phpinfo**, dicha página generaría el contenido de la Figura 2.7.

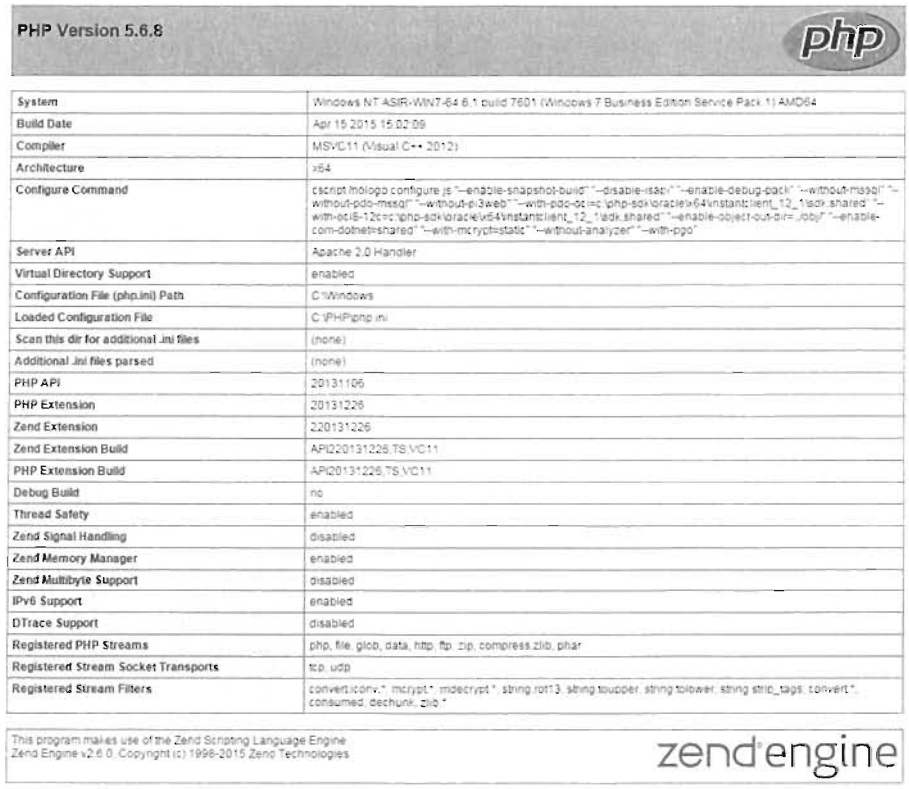


Figura 2.7: Ejemplo de resultado con PHP perfectamente instalado

ACTIVIDAD 2.3:

- Realiza la Práctica 2.3 *"Instalación de PHP en un servidor Apache en Windows"*, en la página 98
- Intenta realizar la misma práctica para un sistema de 32 bits

2.5.2.2 INSTALACIÓN DE PHP EN LINUX MEDIANTE EL CÓDIGO FUENTE

Se trata de realizar una instalación a partir del código fuente PHP. Este tipo de instalación tiene sentido si también Apache lo habíamos instalado de esta forma.

Este tipo de instalación permite personalizar completamente la instalación de PHP. Los pasos fundamentales son:

[1] Descargar el archivo comprimido con el código fuente disponible en la página http://php.net/downloads.php.

[2] Descomprimir el archivo.

[3] Instalar paquetes necesarios para la compilación y especialmente los paquetes **lib-xml2-dev** (o **lib-xml2-devel**) y el intérprete de **perl**.

[4] Configurar la instalación mediante el comando `configure` que nos permitirá decidir qué librerías compilamos. Por ejemplo, se suelen incluir las librerías **mysql** y **mysqli** para gestionar bases de datos MySQL.

[5] Compilar e instalar mediante los comandos `make` y `make install`.

[6] Crear el archivo **php.ini** a partir de una copia del archivo **php.ini-development** o del archivo **php.ini-production**, ambos dentro del directorio con el código fuente PHP.

[7] Modificar el archivo de configuración de Apache para que cargue el módulo PHP.

[8] Modificar la política del módulo de seguridad **SELinux** para que permita la ejecución del módulo de PHP. Esto es necesario en los Linux que disponen de esa capa de seguridad.

[9] Reiniciar Apache.

[10] Crear una página PHP de prueba y cargarla en el navegador.

ACTIVIDAD 2.4:

- Realiza la Práctica 2.3 *"Instalación de PHP en un servidor Apache en Windows"*, en la página 98
- Intenta realizar la misma práctica para un sistema de 32 bits

---

ACTIVIDAD 2.5:

- El paso en el que se invoca al comando **configure** es muy interesante ya que permite realizar instalaciones de PHP muy personales.
- Hay más opciones disponibles para este comando. Por ejemplo tenemos la opción `--with-config-file-path=ruta` que permite indicar la ruta en la que PHP buscará el archivo de configuración, lo que nos permite colocar el archivo php.ini en cualquier ruta del sistema. También podremos indicar opciones de compilación para añadir librerías que nos interesen.
- Lo interesante es que se puede invocar una y otra vez a este comando y después compilar el código PHP sin que necesitemos reinstalar Apache.
- Examina las posibilidades de este comando en la dirección: http://php.net/manual/es/configure.about.php

---

2.5.2.3 INSTALAR PHP USANDO PAQUETES EN LINUX

Siempre es más cómodo este tipo de instalación. La pega es que debemos conocer el gestor de paquetes del sistema Linux en el que deseamos hacer la instalación y, además, el nombre de

los paquetes necesarios. Por otro lado, los cambios que se realizan al sistema: rutas donde se han dejado archivos, ficheros de configuración modificados, etc. No son fáciles de conocer.

Se explica la instalación para sistemas Red Hat (como CentOS por ejemplo) y Debian (como Ubuntu por ejemplo) con gestores automáticos de paquetes.

2.5.2.4 INSTALACIÓN EN SISTEMAS DEBIAN

El comando de instalación de paquetes sería:

```
# apt-get install php5 php-pear php5-mysql
```

Se instala el paquete PHP, junto al módulo PHP-PEAR (imprescindible) y aprovechamos para instalar el módulo MySQL para PHP.

Tras la instalación debemos reiniciar Apache. Podemos crear un sencillo programa PHP y poner la URL del mismo en el navegador para ver si el servidor aplica correctamente el módulo PHP.

2.5.2.5 INSTALACIÓN EN SISTEMAS RED HAT

En este caso se ejecuta:

```
# yum install php php-mysql
```

Se instala el paquete PHP y también aprovechamos para instalar el módulo MySQL para PHP.

Tras reiniciar Apache, el módulo PHP ya estará funcionando. Basta crear un documento PHP y poner su URL en el navegador para probar la instalación.

2.5.3 CONFIGURACIÓN DE PHP

2.5.3.1 COMANDO PHP

Sea cual sea la instalación que hemos elegido para PHP, habrá disponible un archivo binario llamado **php** que, entre otras funciones, tiene la capacidad de interpretar código PHP para, por ejemplo, indicar si tiene errores. Además, permite modificar el funcionamiento de php.

Para poder invocar al archivo php, el directorio en el que se encuentra debe formar parte del PATH del sistema. Las opciones del archivo están disponibles invocando:

```
php --help
```

Por ejemplo, el comando:

```
php -v
```

Nos muestra la versión instalada de PHP

2.5.3.2 UBICACIÓN DE PHP.INI

El archivo de configuración para PHP es normalmente **php.ini** que se encuentra (salvo que se indique lo contrario) en la raíz de instalación de PHP. Como se ha comentado en las instalaciones, lo normal es copiar alguno de los archivos semipreparados de configuración (***php.ini-development*** o ***php.ini-production***) para tener ya preconfiguradas las opciones de trabajo, sean para un entorno de desarrollo o para producción. En el caso de las instalaciones por paquetes de Linux, dispondremos directamente del archivo php.ini con opciones ya preconfiguradas.

Para saber dónde o qué archivos de configuración usa PHP (especialmente en instalaciones automatizadas con instaladores o paquetes) basta con ejecutar desde la línea de comandos:

```
php --ini
```

Ejemplo:

```
> php --ini
Configuration File (php.ini) Path: /etc
Loaded Configuration File:         /etc/php.ini
Scan for additional .ini files in: /etc/php.d
Additional .ini files parsed:      /etc/php.d/curl.ini,
/etc/php.d/fileinfo.ini,
/etc/php.d/json.ini,
/etc/php.d/mysql.ini,
/etc/php.d/mysqli.ini,
/etc/php.d/pdo.ini,
/etc/php.d/pdo_mysql.ini,
/etc/php.d/pdo_sqlite.ini,
/etc/php.d/phar.ini,
/etc/php.d/sqlite3.ini,
/etc/php.d/zip.ini
```

En el ejemplo anterior se observa que el archivo de configuración principal es **/etc/php.ini** , pero hay más archivos cuyo contenido también influirán en la configuración.

2.5.4 MODIFICACIÓN DE PHP.INI

En el archivo php.ini, las líneas de comentarios comienzan con un punto y coma (;). Y las instrucciones de configuración se indican con la forma:

```
instrucción=valor
```

Como ocurre con el archivo de configuración Apache, en PHP hay numerosas opciones de configuración.

Ejemplo:

```
display_errors=on
```

Nos permite ver los errores de PHP para depurar nuestro código. En un entorno de producción, esta opción está en modo **off** (apagada).

Hay muchísimas opciones de configuración en el archivo, las más interesantes están comentadas en la dirección http://php.net/manual/es/configure.about.php .

2.6 INSTALACIÓN Y CONFIGURACIÓN DE MYSQL

2.6.1 INTRODUCCIÓN

MySQL es un Sistema Gestor de Bases de Datos. Tradicionalmente se ha unido a Apache y PHP para formar una pila de software capaz de dar sustrato a la creación completa de aplicaciones web. Actualmente pertenece a la empresa Oracle que con esta compra intenta asegurarse el liderazgo en el mundo de las bases de datos para el desarrollo. MySQL mantiene una licencia de código abierto GPL y otra comercial para las empresas que deseen un soporte completo por parte de Oracle.

A pesar de la aparición de numerosos sistemas de bases de datos de todo tipo, MySQL sigue siendo el sistema de bases de datos más utilizado para crear aplicaciones web, especialmente con PHP.

2.6.2 DOCUMENTACIÓN

MySQL tiene una excelente documentación, aunque en inglés. Está disponible en la dirección http://dev.mysql.com/doc/

Lógicamente aparece documentación para todos los aspectos de MySQL, para ver los aspectos relativos a la instalación de la versión 5.6 de MySQL tenemos la dirección:

http://dev.mysql.com/doc/refman/5.6/en/installing.html

2.6.3 INSTALACIÓN DE MYSQL

2.6.3.1 INSTALACIÓN EN WINDOWS MEDIANTE ARCHIVO INSTALADOR

Es la forma más fácil de instalar, aunque también la que añade más software a nuestro sistema. El instalador instala:

- El servidor MySQL.

- **MySQL Workbench**, que es un entorno de trabajo completo para gestionar y administrar nuestras bases de datos.
- Conectores para los lenguajes que quieran acceder a las bases de datos: **Java**, **Python**, **.Net**, **conectores ODBC** y, por supuesto, PHP.
- Herramientas, documentación y ejemplos para aprender a trabajar con MySQL, incluido un conector con Excel y con Visual Studio.

Durante la instalación podremos elegir exactamente qué componentes instalamos.

Antes de realizar la instalación hay que tener en cuenta que MySQL exige tener instalado el **Framework .NET** de Microsoft, cuya descarga es gratuita.

Para descargar el instalador basta acceder a la dirección http://dev.mysql.com/downloads/installer/

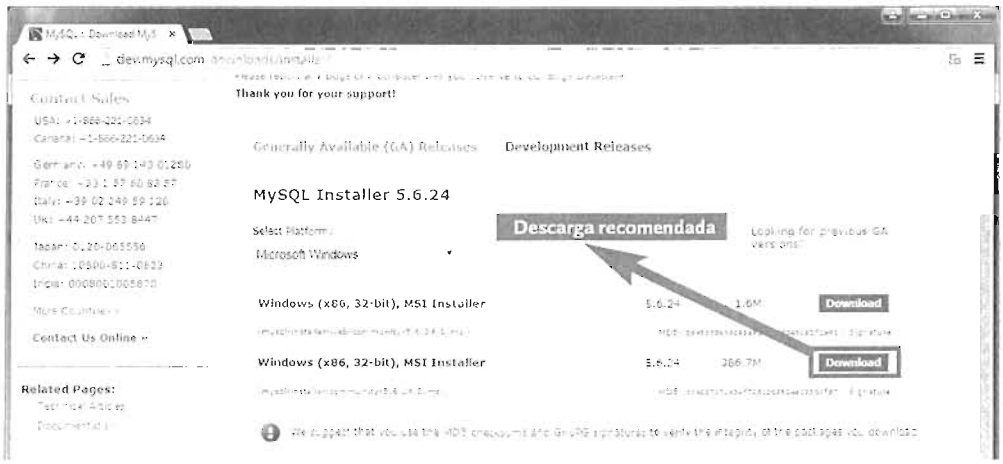


Figura 2.8: Página de descarga del instalador de MySQL en entorno Windows

Tras hacer clic en **Download**, una segunda página nos anima a utilizar nuestra cuenta Oracle (si la tenemos) o a hacerla ahora. Una tercera posibilidad es hacer clic en el texto "***No thanks, just start my download***", que se encuentra en la parte inferior y que permite iniciar directamente la descarga.

Tras la descarga bastará iniciar el instalador y seguir instrucciones. El instalador nos permitirá decidir qué componentes de MySQL instalaremos.

2.6.3.2 INSTALACIÓN EN WINDOWS MEDIANTE ARCHIVO COMPRIMIDO ZIP

Se trata de no utilizar un instalador, sino de un archivo ZIP que dentro contiene los binarios de MySQL. Realmente es una opción más aconsejable, ya que es más ligera y adecuada para usar MySQL en entornos de desarrollo en Windows.

Para instalar con binarios comprimidos tendremos que:

[1] Descargar los archivos desde la dirección http://dev.mysql.com/downloads/mysql/ correspondientes a nuestra instalación de Windows y Apache.

[2] Extraer los archivos descargados en el directorio en el que deseemos que esté la raíz de MySQL.

[3] Crear el archivo de configuración **my.ini**. Lo habitual es copiar el archivo **my-default.ini** que se encuentra en el directorio raíz de MySQL y dejar la copia en la raíz de Windows (la raíz de Windows es accesible usando la ruta **%WINDIR%**) con el nombre win.ini.

[4] En el archivo my.ini asegurar que tenemos las siguientes líneas:

```
[mysqld]
# ruta a la raíz de instalación de MySQL
basedir=c:/mysql
# ruta al directorio de datos de MySQL
datadir=c:/mydata/data
```

Si deseamos un directorio de datos (**data**) en una ruta diferente de la que tenemos por defecto, tendremos que copiar el contenido del directorio data a la nueva ubicación y luego modificar el archivo **my.ini** para indicar la nueva ruta al directorio de datos.

[6] Añadir el directorio bin de MySQL a la variable PATH de Windows.

[7] Arrancar MySQL la primera vez mediante:

```
> mysqld --console
```

A partir de ahí MySQL ya estará instalado. Si hemos instalado PHP con las librerías de acceso a MySQL, podremos acceder a las bases de datos MySQL desde PHP.

Arranque y Parada de MySQL. Configuración como servicio de Windows

Se indican los comandos de consola relacionados con el arranque y parada de MySQL.

- `mysqld`. Sin parámetros, simplemente lanza MySQL.
- `mysqladmin -u root shutdown`. Detiene MySQL. Se requerirá indicar la contraseña de root.
- `mysqld --install`. Instala MySQL como servicio. EL nombre del servicio será precisamente **MySQL** y quedará configurado para iniciarse cada vez que se arranque el sistema.
- `mysqld --remove`. Elimina MySQL como servicio.
- `mysqld --install-manual`. Instala MySQL como servicio, pero no arrancará automáticamente cada vez que se inicie Windows (instalación manual de servicio).

- `mysqld --install mysql5.6`. Instala como servicio MySQL y utiliza como nombre de servicio *mysql5.6*. También con los parámetros `--install-manual` y `--remove` podemos indicar un nombre para el servicio.

Comprobación de la instalación

Algunos comandos en la consola nos permiten saber si la instalación es correcta:

- `mysqladmin version`. Obtiene el estado y versión instalada de MySQL.
- `mysqlshow`. Muestra las bases de datos de MySQL, si el servidor no funciona, este comando no muestra nada.
- `mysqlshow -u root`. Muestra las bases de datos de MySQL accesibles por el usuario **root**, entre ellas estará el diccionario de datos.
- `mysql -u root`. Conecta con el intérprete de línea de comandos de MySQL. Una vez conectemos, podremos, por ejemplo, ejecutar el comando `show databases;` para ver la lista de bases de datos. El comando `quit` permite salir del intérprete.

2.6.3.3 INSTALACIÓN EN LINUX MEDIANTE BINARIOS GENÉRICOS

Esta forma de instalar permite instalar MySQL en cualquier distribución Linux. Solo necesitamos un prerrequisito: debe estar instalada la librería **libaio**. La forma de instalar esta librería dependerá del sistema Linux que tengamos. En concreto en Red Hat se instala con `yum install libaio` y en Debian con `apt-get install libaio1`.

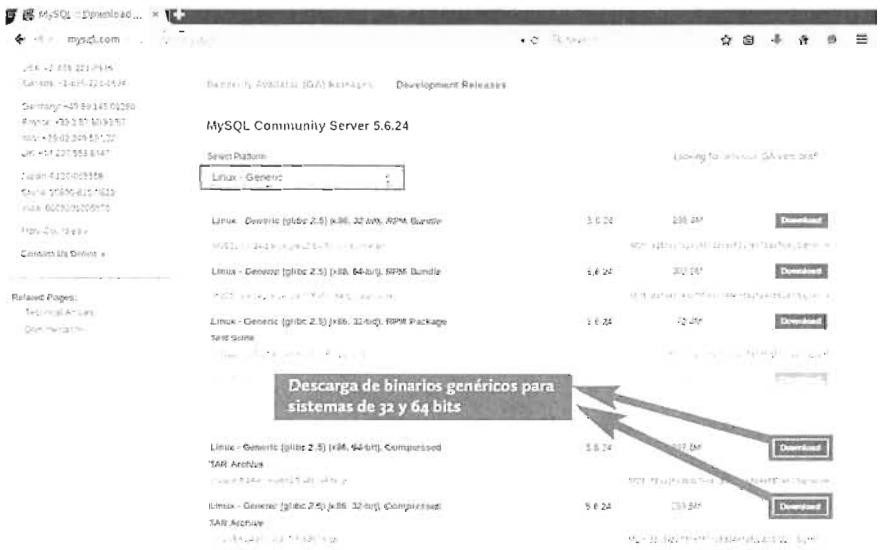


Figura 2.9: Página de descarga de los archivos binarios genéricos para cualquier instalación Linux o Unix

Para realizar la instalación, hay que realizar estas tareas:

[1] Descargar y descomprimir los archivos binarios de MySQL en el directorio que deseemos utilizar como raíz de MySQL.

[2] Disponer de un usuario y un grupo, distinto de ***root***, a los que otorgaremos la propiedad de la raíz de MySQL, así como de su directorio de datos. Tradicionalmente tanto al usuario como al grupo se les llama MySQL.

[3] Añadir el directorio **bin** dentro de la raíz de MySQL al PATH de Linux.

[4] Lanzar el comando de instalación de MySQL. Por ejemplo (suponiendo que **asir** es el nombre del usuario al que se le han otorgado los permisos de uso de MySQL):

```
# mysql_install_db --user=asir
```

[5] Lanzar por primera vez MySQL con el comando:

```
# mysqld_safe --user=asir &
```

El resto de veces que arranquemos MySQL, deberemos estar conectados con el usuario relacionado con la instalación (***asir*** por ejemplo) y ejecutar simplemente el comando `mysqld`.

Comprobar la instalación

En Linux dentro del directorio **bin** disponemos de los mismos comandos para comprobar la instalación vistos para Windows.

Arranque y Parada de MySQL

- `mysqld`. Sin parámetros, simplemente lanza MySQL.
- `mysqladmin -u root shutdown`. Detiene MySQL. Suponiendo que el usuario root no tiene contraseña.

2.6.3.4 INSTALACIÓN EN LINUX MEDIANTE PAQUETES

Como siempre es una instalación más sencilla, pero menos personalizable. Suponemos ya instalado Apache y PHP.

- Instalación de paquetes en Debian, Ubuntu y similares:

```
# apt-get install mysql-server mysql-client
```

Durante la instalación se nos pregunta la contraseña del usuario root. Con esa contraseña podremos acceder a todos los servicios de conexión y configuración.

Si hemos instalado previamente la librería **php5-mysql** ya podremos conectar con MySQL desde PHP.

- Instalación en sistemas Red Hat

```
# yum install mysql-server mysql-client
```

En muchas distribuciones tendremos que integrar previamente el repositorio oficial de MySQL en nuestro sistema.

2.6.4 MYSQL Y MARIA DB

Con la compra de Sun por parte de la empresa Oracle, la comunidad de desarrolladores de MySQL tuvo inquietud respecto a la forma de negocio que iba a tomar MySQL (que pertenecía a Sun, y que anteriormente pertenecía a una empresa independiente).

Realmente Oracle ha seguido siendo muy respetuosa con la comunidad y mantiene MySQL con licencia GPL de código abierto sin soporte, aunque también permite utilizar una licencia comercial con soporte por parte de la empresa.

Michael Widenius, fundador de MySQL, creo **Maria DB** con total compatibilidad con MySQL para asegurar que siempre hubiera disponible una versión abierta de MySQL.

La estructura de archivos, comandos, scripts... son los mismos que en MySQL. Por lo que lo comentado aquí sobre MySQL vale para MariaDB.

MariaDB ha sido muy bien acogida por muchas distribuciones de Linux; es incluso más fácil instalar en Linux MariaDB que MySQL en las últimas versiones de Linux, ya que MariaDB está incluida en la mayoría de repositorios, mientras que MySQL no.

La instalación de MariaDb por paquetes es:

- Distribuciones Debian

```
# apt-get install mariadb-server mariadb
```

- Distribuciones Red Hat

```
# yum install mariadb-server mariadb
```

En muchas distribuciones tendremos que integrar previamente el repositorio oficial de MariaDB en nuestro sistema.

2.6.5 CONFIGURACIÓN DE MYSQL

Al igual que PHP y Apache, MySQL dispone de archivos de texto para establecer la configuración del sistema. Detalles como el puerto por el que escucha MySQL o la ruta a las bases de datos, se almacenan en esos archivos.

Puede haber varios archivos, según la ruta unos se leen antes y otros después. Según el sistema operativo las rutas son:

- En Windows los archivos se llaman ***win.ini*** o ***win.cnf***. Lo aconsejable es llamarles ***win.ini***. Las rutas que busca MySQL, colocadas de las primera a las últimas que lee son:

- *%PROGRAMDATA%\MySQL\MySQL Server 5.6\my.ini*
- %WINDIR%\mi.ini
- C:\win.ini
- *raizMySQL*\my.ini

- En Linux se llaman ***win.cnf*** y el orden de búsqueda es:
  - /etc/my.cnf
  - /etc/mysql/my.cnf
  - SYSCONFDIR/my.cnf
  - $MYSQL_HOME/my.cnf
  - ~/.my.cnf

Si ejecutamos MySQL con el parámetro **--defaults-extra-file**, podemos indicar la ruta a un archivo de configuración personal, que le leerá MySQL tras los archivos globales de configuración.

2.6.6 ESTABLECIMIENTO DE LA SEGURIDAD EN MYSQL

Por defecto en la mayoría de instalaciones de MySQL, el usuario principal **root** no tiene contraseña. Eso significa que cualquier persona podría hacerse administrador de nuestro sistema MySQL y ocasionar serios destrozos fácilmente. Por ello MySQL proporciona un archivo script llamado **mysql_secure_installation** que nos permite asegurar la instalación. Al ejecutar ese archivo el proceso que realiza es:

[1] Solicitar la contraseña actual del usuario root, en la mayoría de instalaciones, por defecto root se instala sin contraseña, lo que supone un grave problema de seguridad.

[2] Cambiar la contraseña de root.

[3] Eliminar los usuarios anónimos

[4] Eliminar la base de datos **test**, a la que tiene acceso cualquier usuario. Incluso los usuarios anónimos podrían crear bases de datos que empiecen con la palabra **test**.

[5] Prohibir el acceso a usuarios desde sitios remotos. De esta forma solo podremos conectar a la base de datos desde el servidor en el que se encuentra la propia base de datos, lo cual supone una medida de seguridad muy interesante para evitar ataques. Esta opción solo tiene sentido si Apache con PHP y MySQL están en el mismo servidor, de otro modo desde PHP tampoco podríamos conectar con la base de datos.

[6] Refrescar la base de datos para que las opciones elegidas funcionen instantáneamente.

En Linux **mysql_secure_installation** reside en el directorio **bin**, dentro de la raíz de instalación de MySQL.

En Windows no existe este script tal cual, pero sí hay (dentro del directorio **bin**) un archivo llamado **mysql_secure_installation.pl** creado en lenguaje **Perl**. Para poder ejecutarle necesitamos instalar primero un intérprete de lenguaje Perl, el más popular para Windows es **Strawberry Perl**.

Para instalar Strawberry Perl, bastará con ejecutar el archivo de instalación descargado desde la dirección http://strawberryperl.com/ y, una vez instalado, desde la línea de comandos de Windows, invocar a **mysql_secure_installation.pl**. Las instrucciones que aparecen son iguales a las del script del sistema Linux.

2.7 INSTALACIÓN DE SOLUCIONES APACHE,PHP Y MYSQL INTEGRADAS

2.7.1 INTRODUCCIÓN

La tarea de instalar y configurar Apache y PHP es tediosa y compleja. Cuando una persona está en proceso de aprendizaje de PHP, lo que la interesa es la solución que le permita de la forma más rápida practicar con PHP. Eso es lo que facilitan las soluciones integradas.

Se trata de paquetes de software que instalan fácilmente Apache y PHP (e incluso MySQL y otro software interesante). No solo instalan el software, además lo dejan preconfigurado para que funcione directamente.

En la fase de producción de una aplicación web, no se debe instalar de esta forma un servidor de aplicaciones y de base de datos, vale solo para el entorno de desarrollo o de prácticas. Siempre es mejor instalar de forma más personal (de alguna de las formas vistas en los apartados anteriores) a fin de optimizar la instalación.

No obstante, todas las soluciones completas se pueden personalizar y configurar perfectamente, ya que se puede aplicar todo lo aprendido sobre la configuración de Apache, PHP y MySQL en los apartados anteriores.

2.7.2 XAMPP

Se trata de la instalación más popular de estos servicios. Es multiplataforma, funciona para Windows, Linus y OSX. Instala **Apache**, **MySQL**, **PHP**, **phpMyAdmin**, **Perl**, e incluso un servidor **FTP (FileZilla)**, otro de **Java** y otro de correo, además de otras muchas herramientas (por ejemplo **Open SSL**), lo que la hace muy versátil. Normalmente no necesitaremos instalar tanto.

La página de descarga y de documentación es:
http://www.apachefriends.org/es/xampp.html.

Desde la página (que está en español) disponemos de los enlaces de descarga del sistema y de la documentación para realizar la instalación y configuración del software.

Figura 2.10: Página de descarga de XAMPP en la dirección https://www.apachefriends.org/es/

2.7.3 INSTALACIÓN DE XAMPP EN WINDOWS

Lo más cómodo es, desde la página de descarga, simplemente hacer clic sobre el botón ***XAMPP para Windows***, se descargará el instalador del sistema para la versión de Windows que tengamos (32 o 64 bits).

No obstante haciendo clic en ***Pulsa aquí para otras versiones,*** podemos examinar otras posibles descargas. XAMPP siempre sugiere la última versión de Apache y PHP, pero podemos buscar una versión anterior. Además, podemos instalar versiones de XAMPP en formato ZIP que no requieren instalación.

Nada más descargar, podemos consultar la documentación (en español) disponible en la dirección https://www.apachefriends.org/es/faq_windows.html o en el enlace que aparecerá en la ventana.

Suponiendo que hemos descargado la versión con instalación, los pasos para instalar son los siguientes:

[1] Descargar el instalador

[2] Ejecutar el instalador y seguir los pasos del asistente. En el segundo paso se nos requerirá que elijamos los componentes a instalar (véase Figura 2.11). No necesitamos instalar todo, aunque tampoco pasa nada si lo hacemos, ya que los servicios relacionados con cada componente podemos activarlos o no según deseemos. Lo imprescindible es Apache, PHP y MySQL. Conviene también instalar **Perl** para poder ejecutar el script de seguridad de MySQL y muchos programadores instalan **phpMyAdmin** para facilitar la gestión de MySQL.

[3] Hay un paso en el que se nos anima a informarnos sobre lo que la empresa **Bitnami** nos ofrece, que concretamente es la posibilidad de integrar un CMS (**Drupal**, **Wordpress**, **Joomla**, etc.) en nuestro sistema. No es en absoluto imprescindible, pero puede ser interesante.

[4] Al final de la instalación, lo normal es que salte el cortafuegos de Windows (si está activado) e indiquemos que permitimos el acceso de Apache a nuestra red. Si tenemos un cortafuegos personal hay que tener en cuenta los puertos que necesitamos abrir.

[5] Al final de la instalación XAMPP habrá creado un directorio en la raíz del disco duro principal (normalmente *c:\xampp*) y un grupo de programas en el menú de Inicio llamado *XAMPP*.

[6] Bien desde el directorio principal o desde el grupo de programas, podemos acceder al cuadro de control de XAMPP desde el que podremos parar o iniciar los servicios deseados y modificar la de configuración de todos los servicios instalados.

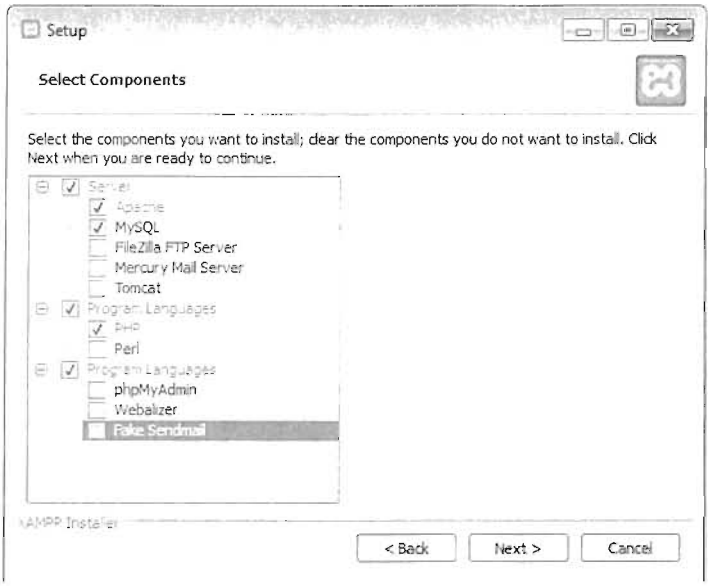


Figura 2.11: Asistente de instalación de XAMPP con las opciones mínimas seleccionadas.

2.7.4 INSTALACIÓN DE XAMPP EN LINUX

La instalación en entorno Linux de XAMPP es también muy sencilla. Los pasos serían los siguientes:

[1] Descargar el archivo de instalación correspondiente a nuestro sistema.

[2] Desde la consola acceder al directorio en el que hemos descargado el archivo y modificar sus permisos con el comando:

```
$ chmod 755 xampp-linux-*-installer.run
```

[3] Lanzar el instalador

```
$ sudo ./xampp-linux-*-installer.run
```

[4] XAMPP estará instalado en el directorio **/opt/lampp**, desde ahí accederemos a todos sus servicios.

[5] Para arrancar tanto Apache como MYSQL haremos uso de:

```
$ sudo /opt/lampp/lampp start
```

El mismo comando indicando stop al final nos permitirá detener los servicios.

2.7.5 MANEJO DE XAMPP

2.7.5.1 MODIFICAR LA CONFIGURACIÓN DE XAMPP DESDE EL PANEL DE CONTROL

Una vez instalado puede que necesitemos configurar o modificar la instalación. La configuración de Apache se realiza mediante el directorio **apache** dentro del directorio de instalación de XAMPP y la de PHP en el directorio llamado precisamente **php**.

Las páginas web se deben de colocar en el directorio **htdocs** dentro de la raíz de instalación de XAMPP (no estará dentro del directorio **apache**).

Lanzar y detener Apache (que ya tendrá configurado el módulo de PHP) se realiza mediante el programa **XAMPP control** dispone en la carpeta de instalación y en los menús de los programas del sistema. Este componente facilita el inicio y detención de los servicios, además de poseer botones directos para modificar la configuración de los mismos.

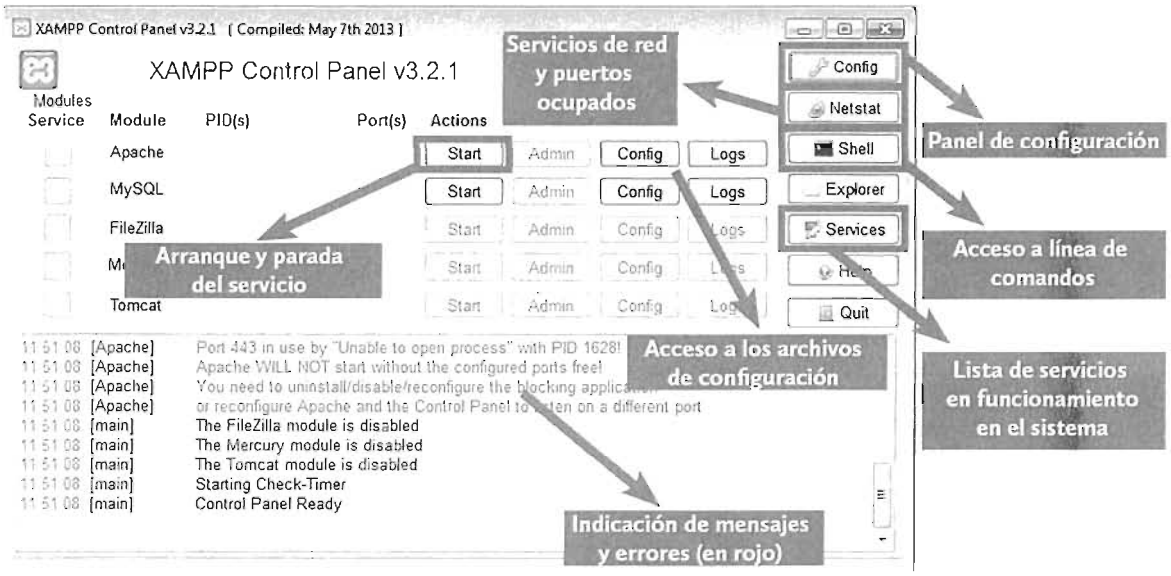


Figura 2.12: Panel de control de XAMPP desde el que se puede configurar todo el funcionamiento de los servicios y servidores instalados

En la *Figura 2.12* podemos observar las posibilidades que ofrece el panel de control de XAMPP. Se detallan a continuación los elementos más interesantes del panel:

- **Lista de servicios.** A la izquierda del panel aparece la lista de servidores instalados, junto con su estado de funcionamiento; si están de color verde es que están funcionando ahora. Además se nos indica los puertos que ocupan y sus números de proceso (PID).
- **Botones de arranque y parada.** Start y Stop, permiten arrancar o detener el servicio.

- **Botones Service.** Permite instalar el módulo (**Apache, MySQL,** etc.) como servicio de Windows. Esto permite que el servidor arranque automáticamente con el arranque del sistema.
- **Botones Config**. Nos permite acceder a los archivos de configuración de los servicios instalados. Es decir, desde ese botón accederemos al archivo **httpd.conf** de Apache, al **php.ini** de PHP o al **my.ini** o **my.cnf** de MySQL
- **Botón Logs.** Permite acceder a los archivos de registro (log) de cada módulo instalado.
- **Botón Config**, con dibujo de llave inglesa. Permite realizar de forma sencilla la mayoría de acciones de configuración básicas sobre el software instalado.

  Entre ellas:

  - Indicar el editor predeterminado para abrir los archivos de configuración. Normalmente aparece el bloc de notas (***notepad***) pero se puede elegir otro editor más avanzado que tengamos en el sistema.
  - Indicar qué módulos arrancan automáticamente cuando abramos este panel (***Autostart of modules***)
  - Cambiar el nombre de los servicios y puertos, de cada módulo instalado (***Service and Port Settings***). Indudablemente es la opción más importante de este panel.
- **Botón Shell.** Abre una ventana de terminal ya situada en la raíz de XAMPP para ejecutar los comandos que nos parezcan convenientes.
- **Botón Explorer**. Abre una ventana de navegación sobre el directorio raíz de XAMPP.
- **Botón Services**. Nos proporciona la lista de servicios de Windows que tengamos instalados.
- **Botón Help**. Ayuda de XAMPP.
- **Botón Quit**. Cierra el panel de control.

En definitiva, el panel de control es una herramienta magnífica para gestionar todos los elementos de nuestra instalación.

2.7.5.2 CONFIGURACIÓN MANUAL DE XAMPP

Hay que tener en cuenta que XAMPP se puede instalar incluso en sistemas Linux sin entorno gráfico, por lo que no siempre dispondremos de Panel de Control.

En todo caso, si sabemos manejar Apache, PHP y MySQL (con lo visto en los apartados anteriores de este tema) con XAMPP todo funciona igual, por ejemplo en una instalación Linux, este comando:

```
$ sudo  /opt/lampp/bin/httpd -k start
```

Arrancaría Apache igual que en cualquier instalación sin XAMPP. La única diferencia está en la ubicación de los directorios. Así desde la raíz de xampp que suele ser **c:\xampp** en Windows y **/opt/lamp** en Linux, se encuentran los siguientes directorios:

- **apache** o **apache2**. Raíz de instalación de Apache.
- **htdocs**. Raíz de documentos del servidor Apache.
- **php**. Raíz de PHP.
- **mysql**. Raíz de MySQL.
- **bin** y **sbin**. Solo en el caso de Linux. Contienen todos los ejecutables de los servicios instalados: ***httpd***, ***mysqld***, ***php***, ***mysql_secure_admin***, ***htpasswd***, etc.

2.7.5.3 SEGURIDAD DE XAMPP

XAMPP no es un software en el que prime la seguridad. Hay que recordar que es un buen sistema para pruebas, no para producción. Entre los problemas que conlleva una instalación de XAMPP tenemos:

- El usuario root de MySQL no tiene contraseña.
- MySQL está accesible desde la red.
- Si instalamos ***PHPMyAdmin*** también podremos acceder al servicio desde la red.
- Los ejemplos y test están accesibles.
- Las contraseñas y nombres de los usuarios administradores de servicios como ***ProFTP***, **Mercury** o **FileZilla**, si hemos instalado esos servicios, son conocidos ya que XAMPP usa la configuración por defecto.

Todos estos problemas podemos arreglarlos con lo visto durante el tema. Pero también podemos acceder a la dirección **localhost/security** desde el navegador o bien ejecutar el comando `sudo /opt/lampp/lampp security`, donde un asistente nos ayudará a paliar estos problemas.

Recientemente XAMPP ha añadido una medida de seguridad: que solo pueda acceder a las páginas de administración de XAMPP desde el ordenador local.

2.8 PRÁCTICAS RESUELTAS

Práctica 2.1: Instalación binaria de Apache en Windows

SOLUCIÓN: PRÁCTICA 2.1

En esta práctica supondremos que tenemos una máquina virtual o real con un sistema Windows de 64 bits. Esta instalación será la base de prácticas posteriores. Estos son los pasos:

[1] Desde la página http://www.apachelounge.com/download/ descargar el archivo ZIP de Apache de 64 bits VC11. La versión recomendada en el momento de escribir este libro es la versión 2.4.12 de tipo **VC11**.

[2] Descomprime el archivo en el directorio **C:\Apache24**, de esa forma Apache funcionará sin configurar nada. Si queremos realizar la instalación en otra ruta habrá que modificar los archivos de configuración de Apache.

[3] Descarga la **versión redistribuible de Visual C++ para Visual Studio 2012** (es decir: **VC11**), desde http://www.apachelounge.com/download/ :

Figura 2.13: Página de descarga del Visual C++ Redistributable, componente imprescindible para que funcione la instalación de Apache en Windows.

[4] Tras hacer clic en el botón **Descargar**, una segunda página nos muestra qué versión deseamos bajar. Ahí es donde elegimos la versión de 32 o 64 bits (también hay disponible una versión para máquinas con procesadores ARM), para esta práctica descargamos la versión de 64 bits.

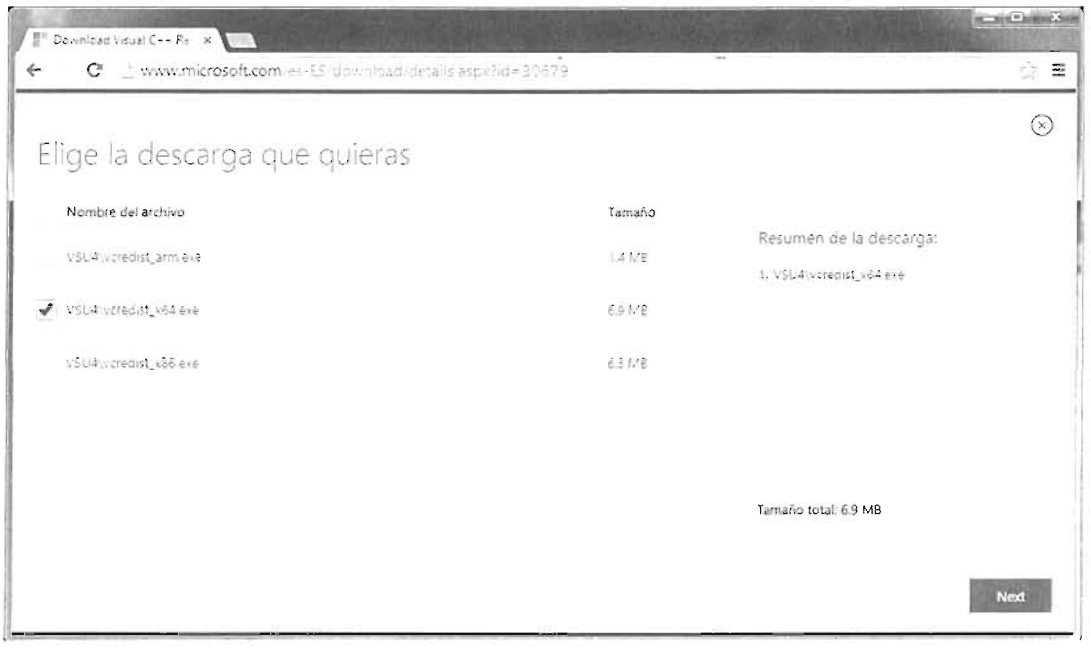


Figura 2.14: Elección de la versión a instalar del Visual Studio C++ Redistributable. En la imagen se ha seleccionado la versión de 64 bits

[5] Instala el software Visual Studio C++ descargado. Bastará con ejecutar el programa de instalación y seguir los pasos.

Figura 2.15: Inicio de la instalación de Visual C++ Redistributable para máquinas de 64 bits

[6] Tras la instalación hay que modificar el **PATH** del sistema para poder acceder a los comandos de Apache dentro del terminal del sistema.

Los pasos a realizar para modificar el PATH son (véase *Figura 2.16*):

[6.1] Vete a las propiedades del **Sistema** desde el **Panel de Control** o pulsando el botón secundario del ratón sobre el icono del Sistema y eligiendo **Propiedades**.

[6.2] Elige **Configuración avanzada del sistema.**

[6.3] Vete al apartado **Opciones avanzadas.**

[6.4] Haz clic en el botón **Variables de entorno.**

[6.5] Marca la variable **Path** en el apartado **Variables del Sistema.**

[6.6] Haz clic en **Editar.**

[6.7] Añade la ruta *C:\Apache24\bin*, en la que estarán ubicados los ejecutables de Apache. Observar que cada ruta en el cuadro se separa con un punto y coma. En ningún caso hay que borrar las rutas ya grabadas, por lo que hay que obrar con precaución.

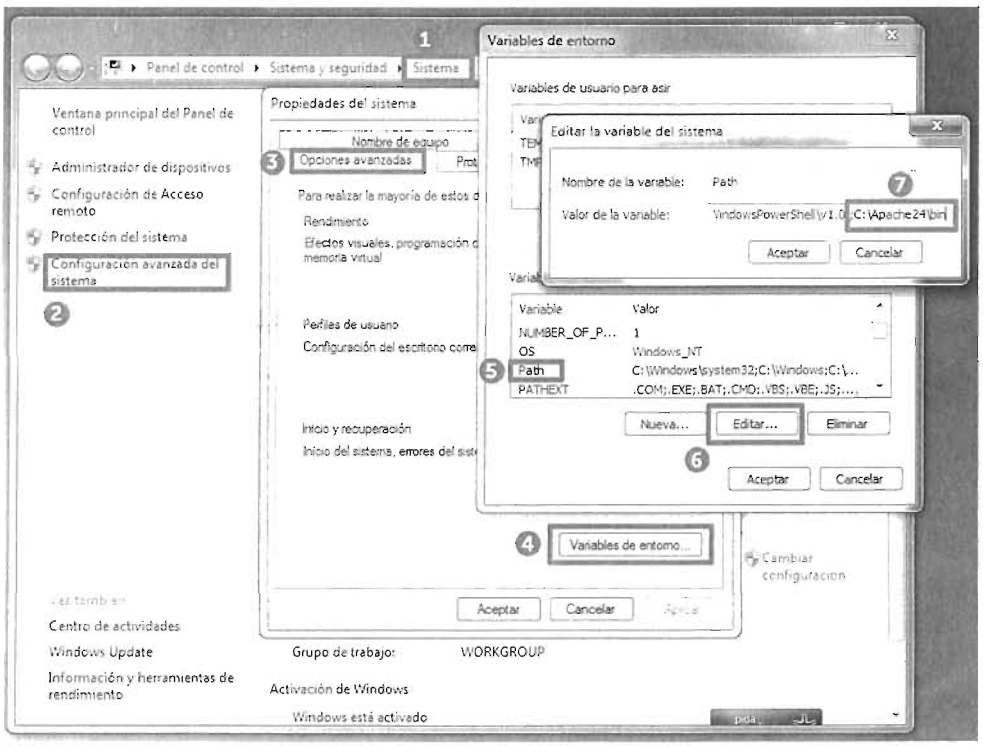


Figura 2.16: Modificación del PATH del sistema para que aparezca el directorio con los programas ejecutables de Apache

[7] En Windows, Apache debe configurarse como servicio. Para ello deberemos realizar estos pasos:

[7.1] Desde la línea de comandos de Windows (abriéndola con permisos de administración) ejecuta el comando **httpd -k install.**

[7.2] Aparecen mensajes indicando el resultado de la acción. Si aparece el texto *The 'Apache2.4' service is successfully installed*, es que el servicio está correctamente instalado.

Puede aparecer un error con el texto ***httpd: Could not reliably determine the server's fully qualified domain name, ... Set the 'ServerName' directive globally to suppress this message*** que simplemente se refiere a que no hemos indicado un

nombre para el servidor. Lo cual se puede arreglar más tarde en el archivo de configuración de Apache; en todo caso no afecta a la ejecución del servidor web Apache.

[7.3] Ejecuta el comando `httpd -k start`, Con este comando Apache debería arrancar.

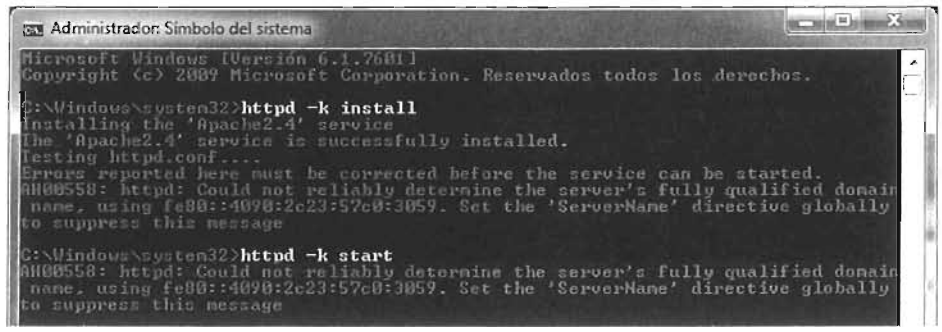


Figura 2.17: Comandos iniciales en Windows para lanzar el servidor Apache

[8] Prueba que Apache funciona correctamente desde nuestro navegador favorito escribiendo la dirección: **http://localhost**. Si todo sale bien, veremos la famosa página inicial de portada del servidor de Apache con el texto ***It works!***

Figura 2.18: Comprobación del funcionamiento del servidor. Si aparece esta imagen, la configuración es correcta

Práctica 2.2: Instalación de Apache en Linux mediante el código fuente

SOLUCIÓN: PRÁCTICA 2.2

Se trata de la instalación recomendada cuando deseamos tener todo el control sobre Apache. En entornos de producción es la habitual. Estos son los pasos que debes hacer:

[1] **Actualiza los repositorios.** Hay varias acciones a tomar que tienen que ver con instalar librerías. La forma habitual de instalarlas es mediante el comando de instalación de paquetes, el cual acude a un repositorio para buscar el programa de instalación. Por eso conviene tener al día los paquetes instalados. La forma es:

- Linux tipo Debian:

```
# apt-get update
```

- Linux tipo red Hat:

```
# yum update
```

[2] **Descarga el código fuente.** Vete a la dirección http://httpd.apache.org/download.cgi y descarga el archivo comprimido en formato **gz** (podriamos utilizar otro formato comprimido también) .

En el caso de encontrarnos en un servidor Linux en modo consola, podemos descargar el código fuente de la versión 2.4.12 de Apache mediante el comando:

```
$ wget http://ftp.cixug.es/apache//httpd/httpd-2.4.12.tar.gz
```

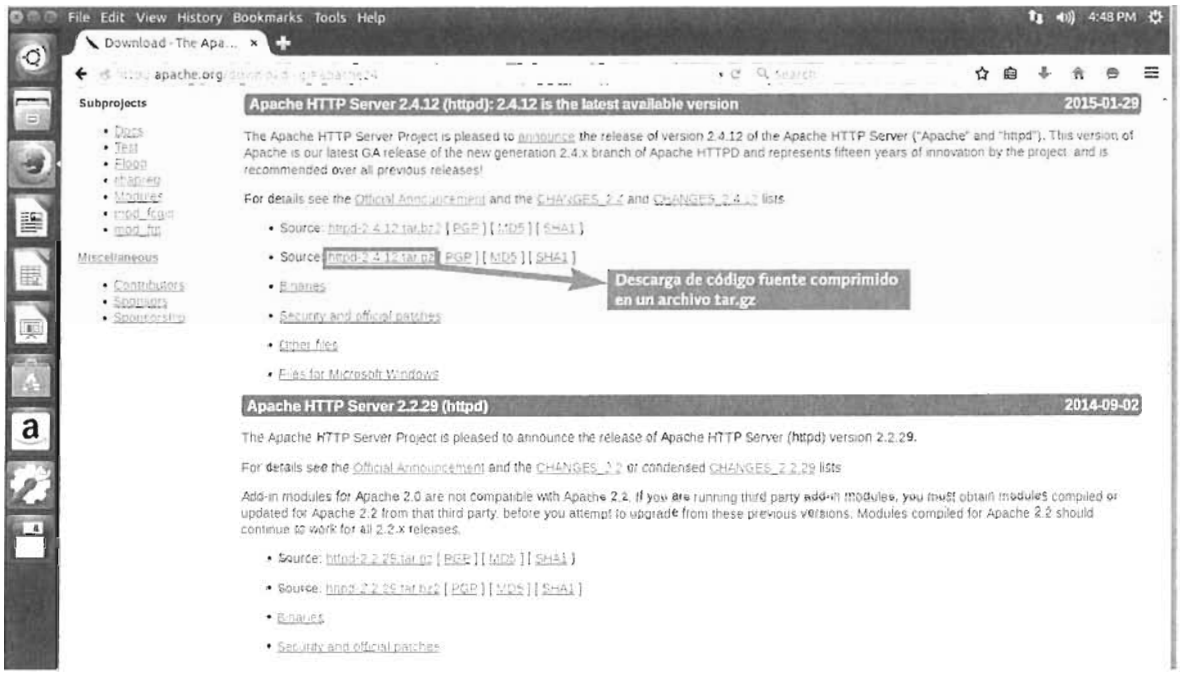


Figura 2.19: Página de descarga de Apache, se resalta la instalación mediante código fuente de la versión 2.4.12 en formato comprimido tar.gz

[3] **Descomprime el código fuente.** Por ejemplo, con el comando:

```
$ tar vzxf httpd-2.4.12.tar.gz
```

El resultado es un directorio que contendrá los archivos fuente de Apache. Borramos el archivo comprimido original.

```
$ rm httpd-2.4.12.tar.gz
```

[4] **Instala las las librerías de compilación.** Puesto que estamos realizando una instalación que requiere compilar el código, se necesita que estén instaladas las herramientas de compilación de lenguaje C y C++. Es muy posible que este paquete ya esté instalado; sino es así, necesitamos instalar los paquetes correspondientes.

En el caso de los Linux de tipo Debian (como Ubuntu o Mint) se puede hacer (usando privilegios de súper usuario) mediante:

```
# apt-get install build-essential
```

En Linux de tipo **Red Hat** no existe ese paquete, por lo que se haría mediante:

```
# yum install gcc gcc-c++ make openssl-devel
```

Podría ser necesario tener los repositorios al día, para lo cual, antes de estos comandos, tenemos que ejecutar ***apt-get update*** (en el caso de Debian) o **yum check-update** (en el caso de Red-Hat).

En otras distribuciones deberemos comprobar que tenemos esos paquetes instalados.

[5] **Instala las librerías APR y APR-Util.** Son las librerías **Apache Portable Runtime**, disponibles en la dirección de Internet http://apr.apache.org/download.cgi.

Si no están instaladas (que será lo normal) hay que realizar la descarga yendo a la dirección http://apr.apache.org/download.cgi

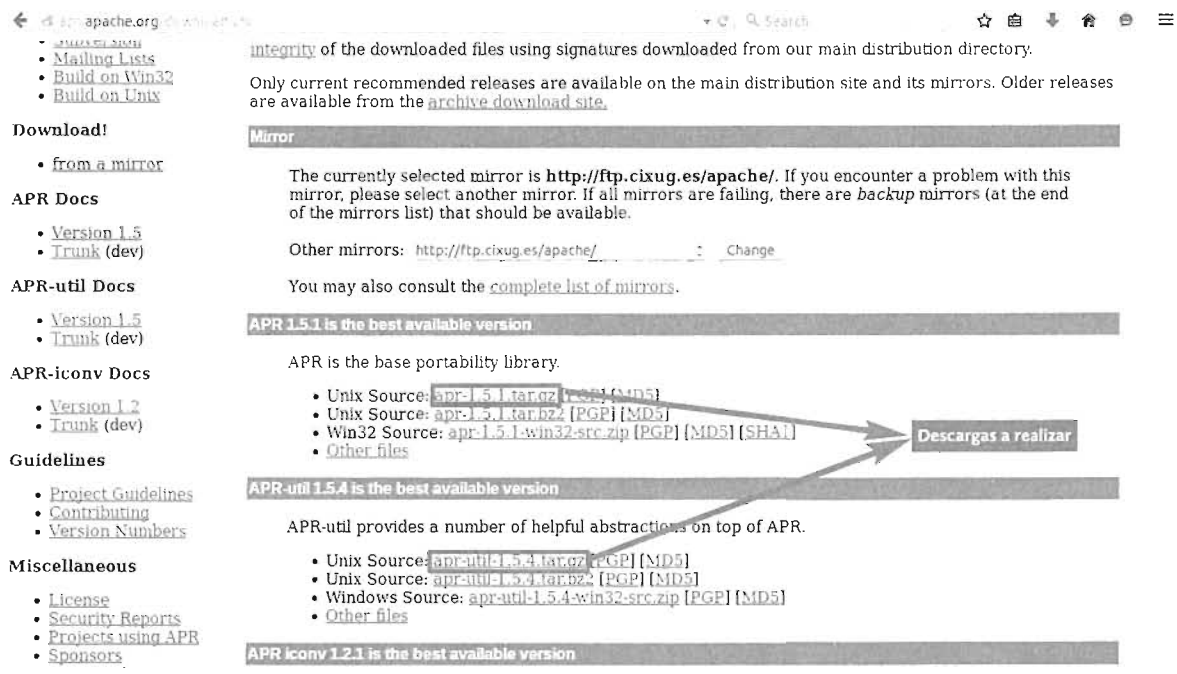


Figura 2.20: Página de descarga de las librerías apr y apr-util

Ambas librerías se deben colocar en el directorio **srclib** dentro del código fuente de Apache. Se detallan a continuación los pasos para instalar estas librerías desde la consola:

[5.1] Descarga los archivos con el código fuente de **APR** y **APR-util** (véase Figura 2.20). Si, para ello, utilizas el comando wget desde la consola, el comando para descargar las versiones 1.5.2 y 1.5.4 de las librerías sería el siguiente:

```
$ wget http://ftp.cixug.es/apache//apr/apr-1.5.2.tar.gz
$ wget http://ftp.cixug.es/apache/apr/apr-util-1.5.4.tar.gz
```

[5.2] Descomprime ambos archivos, por ejemplo con los comandos:

```
$ tar vzxf apr-1.5.2.tar.gz
$ tar vzxf apr-util-1.5.4.tar.gz
$ rm apr*.gz
```

Tras la ejecución del código anterior, se crearán los directorios *apr-1.5.1* y *apr-util-1.5.4*. Además se borran los archivos originalez tar.gz.

[5.3] Mueve ambos directorios dentro del directorio **srclib**, que se encuentra en el directorio con el código fuente de Apache. Si suponemos que tanto las librerías APR como el código fuente de Apache se descomprimieron en el mismo directorio y que el directorio de Apache se llama ***httpd-2.4.12***, los comandos son:

```
$ mv apr-1.5.2/ httpd-2.4.12/srclib/apr
$ mv apr-util-1.5.4/ httpd-2.4.12/srclib/apr-util
```

[6] **Instala de la librería PCRE.** Además de las librerías anteriores, necesitamos instalar la librería de expresiones regulares de Perl, disponible en http://pcre.org y más cómodamente en la página de descargas **SourceForge** http://sourceforge.net/projects/pcre/files

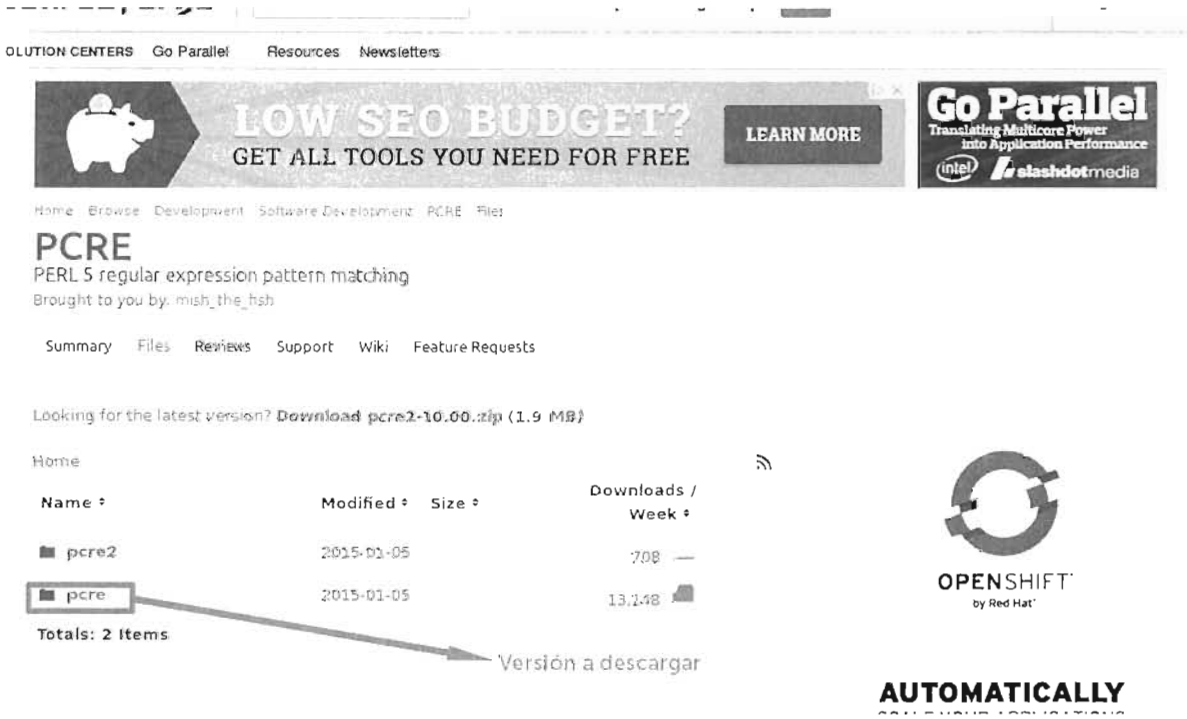


Figura 2.21: Página de descarga del paquete PCRE

Los pasos para la instalación de esta librería son:

[6.1] Descarga con el comando wget la versión deseada de la librería. Por ejemplo para descargar la versión 8.36 de la librería:

```
$ wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.36.tar.gz
```

[6.2] Descomprime el archivo descargado, se creará un directorio (*pcre-8.36*) que contiene el código fuente de la librería.

```
$ tar vzxf pcre-8.36.tar.gz
$ rm pcre*.gz
```

[6.3] Entra en el directorio y, en modo administrador, ejecuta los siguientes comandos (suponemos que la librería la instalamos en la ruta **/usr/local/pcre**):

```
$ cd pcre-8.36
$ sudo ./configure --prefix=/usr/local/pcre
$ sudo make
$ sudo make install
```

[7] **Configura el código fuente de Apache.** Tras haber cumplido los requisitos anteriores, podemos compilar el código fuente de Apache. Bastará con entrar en el directorio de Apache que contiene el código fuente y, en modo administrador, lanzar el siguiente comando (con privilegios administrativos):

```
$ sudo ./configure --prefix=/usr/local/apache2
       --enable-so --with-pcre=/usr/local/pcre
       --with-included-apr
```

Mediante la ejecución de este comando preparamos la instalación de Apache en la ruta */usr/local/apache2*, indicamos que permitiremos añadir módulos a la configuración de Apache (parámetro --**enable-so**), que la librería **pcre** (instalada en el paso anterior) se encuentra en la ruta */usr/local/pcre* y que estamos incluyendo el código de la librería **apr** (comentada en el paso *4*).

[8] **Compila e instala de Apache.** Se realiza mediante los comandos:

```
$ sudo make
$ sudo make install
```

[9] **Añade la carpeta bin al PATH del sistema.** De esa forma los comandos de Apache serán accesibles desde la línea de comandos de Linux. Para ello, basta encadenar la ruta a la carpeta bin de Apache en el PATH, mediante este comando:

```
export PATH=$PATH:/usr/local/apache2/bin
```

Para que ese comando sea permanente y al iniciar sesión ya esté funcionando, esa línea debería estar en el archivo **.bash_profile,** en **.bash_rc** o en **.profile** (dependerá de la distribución de Linux) en el directorio *home* de nuestro usuario. Si deseamos modificar el PATH de todos los usuarios del sistema, se coloca la línea en el archivo **/etc/profile**.

Para hacer que los cambios en los archivos de inicio (*/etc/profile .bash_profile*, etc.) se ejecuten al instante, se pueden lanzar con el comando **source**. Por ejemplo:

```
$ sudo source /etc/profile
```

[10] Lanza Apache.

```
$ httpd -k start
```

[11] Prueba la instalación en el navegador que deseemos introduciendo la dirección *http://localhost.* Si todo sale bien veremos la famosa página inicial de portada del servidor de Apache con el texto *It works!*

Práctica 2.3: Instalación de PHP en un servidor Apache en Windows

SOLUCIÓN: PRÁCTICA 2.3

Para realizar esta práctica, partimos de que tenemos Apache instalado en Windows y funcionando correctamente. Supondremos que lo hemos instalado en el directorio c:\Apache24 como se sugiere en la Práctica 2.1 *"Instalación binaria de Apache en Windows"*, en la página 90. Los pasos que debes realizar son:

[1] Comprueba la versión instalada de Apache. Lo cual se hace simplemente con el comando `httpd -v`:

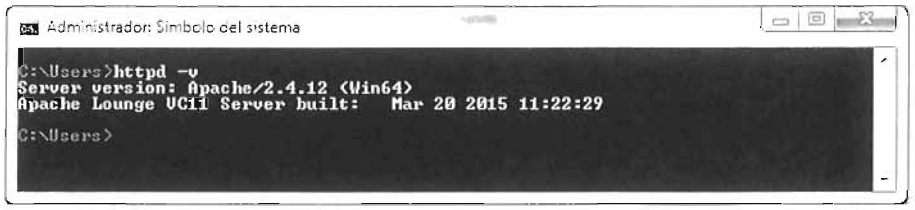


Figura 2.22: Comprobación de la versión instalada de Apache

Mediante este comando tendremos la versión de Apache, se nos indicará además si es de 32 o 64 bits y además la versión de Visual C++ con la que se ha compilado Apache.

En la imagen superior se indica que tenemos la versión 2.4.12 de Apache en 64 bits y compilada con la versión VC11.

[2] Comprueba que tenemos instalado el software **Visual C++ Redistributable para Visual Studio**. Lo lógico es que esté ya instalado, porque es un requisito previo para implementar Apache. De no ser así, tendremos que descargar e instalar ese software.

[3] Descarga el archivo ZIP comprimido con los archivos de PHP de la dirección http://windows.php.net/download. Descargaremos la versión **Thread Safe** que es la apropiada para Apache en Windows (observa la Figura 2.6, página 71). El resto depende de

nuestra versión concreta de Apache, si hemos instalado Apache siguiendo los pasos de la práctica Práctica 2.1, deberemos descargar la versión Thread Safe de 64 bits y de tipo VC11.

[4] Descomprime los binarios en la ruta ***C:\php***. Por supuesto, podríamos elegir otra ruta, pero en el resto de apartados haremos referencia a esa ruta como raíz de PHP

[5] Añade las rutas **C:\php** y **c:\php\ext** a la variable PATH del sistema (el proceso para cambiar el PATH se explicó en la página 91).

[6] Copia el archivo **php.ini-development** de la raíz de PHP y renombra la copia con el nombre **php.ini**.

[7] En el archivo php.ini creado anteriormente, busca y modifica o añade (si no existen) estas líneas:

```
display_errors = On
extension_dir = "./ext"
extension=php_mysql.dll
extension=php_mysqli.dll
```

[8] Edita el archivo de configuración (normalmente **httpd.conf**) de Apache y añade o modifica estas líneas:

```
PHPIniDir "C:/php"
LoadModule  php5_module "c:/php/php5apache2_4.dll"
<FilesMatch \.php$>
      SetHandler application/x-httpd-php
 </FilesMatch>
```

[9] Reinicia Apache.

[10] Crear en la raíz de documentos de Apache (***c:\apache24\htdocs***) el archivo ***prueba.php*** con este contenido:

```
<?php
      phpinfo();
?>
```

[11] Desde el navegador, arrancar Apache y escribir la dirección http://localhost/prueba.php

[12] Comprobar que aparece la página de información de PHP.

Práctica 2.4: Instalación de PHP mediante código fuente en Linux

SOLUCIÓN: PRÁCTICA 2.4

Esta es una práctica coherente con la Práctica 2.2. Así conseguiremos una instalación completa de Apache con PHP que permite su total personalización.

Como en todas las instalaciones mediante código fuente, podemos recompilar el código las veces que estimemos convenientes. Estos son los pasos que has de realizar para este tipo de instalación:

[1] Descarga el archivo comprimido con el código fuente en formato **tar.gz**, disponible en la página http://php.net/downloads.php. Desde la consola podríamos hacerlo con el comando (descargamos la versión 5.6.8 de PHP):

```
$ wget -O php5.tar.gz http://es1.php.net/get/php-5.6.8.tar.gz/from/
this/mirror
```

Observa que en este caso al archivo descargado le hemos llamado **php5.tar.gz** (de otro modo se llamaría *mirror*).

[2] Descomprime el archivo descargado

```
$ tar vzxf php5.tar.gz
```

[3] Crea el directorio en el que se instalará PHP:

```
# mkdir /usr/local/php
```

[4] Puede ser necesario instalar el paquete **libxml2-dev** y el intérprete de Perl. Para ello se ejecuta este comando:

- En Linux tipo **Debian**:

```
# apt-get install libxml2-dev perl
```

- En Linux de tipo **Red Hat**:

```
# yum install libxml2-devel perl
```

[5] Entra en el directorio en el que están los archivos, ya descomprimidos, de PHP y ejecuta el siguiente comando:

```
# ./configure --with-apxs2=/usr/local/apache2/bin/apxs
    --with-mysql  --with-mysqli
    --prefix=/usr/local/php
```

Este comando sirve para configurar la instalación. Hay que indicar dónde está el directorio **apxs**, que estará dentro del directorio **bin** en la raíz de Apache.

La opción **--with-mysql** y **--with-mysqli** permite compilar el código con soporte para las librerías *mysql* y *mysqli* de acceso a MySQL. La opción **--prefix** permite elegir dónde se realizará la instalación (si no se usa *prefix*, también se instala en **/usr/local/php**).

[6] Compila e instala el código mediante estos comandos:

```
# make
# make install
```

[7] Vete al directorio con el código fuente PHP. Copia el archivo ***php.ini-development*** y llamada a la copia: ***/usr/local/php/lib/php.ini:***

```
# cp php.ini-development /usr/local/php/lib/php.ini
```

[8] Modifica (o añade si no estuvieran) estas líneas en el archivo general de configuración de Apache:

```
LoadModule php5_module modules/libphp5.so
<FilesMatch \.php$>
      SetHandler application/x-httpd-php
 </FilesMatch>
```

[9] Si estamos en una versión de Linux (por ejemplo un Linux tipo **Red Hat Enterprise**) en el que esté activado el paquete **SELinux,** ejecuta el comando:

```
# chcon -t textrel_shlib_t
        /usr/local/apache2/modules/libphp5.so
```

[10] Reinicia el servidor Apache (puede ser necesario incluso reiniciar la máquina)

```
# httpd -k restart
```

[11] Crea en la raíz de documentos de Apache (***/usr/local/apache2/htdocs***) el archivo ***prueba.php*** con este contenido:

```
<?php
        phpinfo();
?>
```

[12] Desde el navegador, arranca Apache y escribe la dirección http://localhost/prueba.php

[13] Comprueba que, en el navegador, aparece la página de información de PHP.

Práctica 2.5: Instalación binaria mediante archivo ZIP de MySQL en Windows. Conexión a PHP

SOLUCIÓN: PRÁCTICA 2.5

Seguimos los siguientes pasos para instalar MySQL en nuestro entorno Windows.

[1] Vete a la dirección http://dev.mysql.com/downloads/mysql/. Si hemos realizado las prácticas de instalación de Apache y PHP en Windows: Práctica 2.1 y Práctica 2.3, además podremos conectar MySQL con PHP. Se puede descargar la versión de MySQL de 32 o 64 bits, pero es más recomendable instalar la versión de 64 bits.

Pulsa el botón **Download** correspondiente al archivo de instalación elegido.

Tras pulsar ese botón una segunda pantalla nos pedirá darnos de alta como usuarios (o introducir nuestro usuario y contraseña si ya estamos dados de alta) o bien hacer clic en el texto el texto "*No thanks, just start my download*" para iniciar la descarga inmediatamente.

[2] Extrae el archivo ZIP en la ruta *C:\mysql*.

[3] Copia el archivo **my-default.ini** situado en *C:\MySQL* y deja la copia con el nombre **my.ini** en el directorio raíz de Windows, cuya ruta será *%WINDIR%*).

[4] Modifica el archivo my.ini para que tenga estas líneas:

```
[mysqld]
# ruta a la raíz de instalación de MySQL
basedir=c:/mysql
# ruta al directorio de datos de MySQL
datadir=c:/mydata/data
```

[5] Coloca el directorio **bin** de MySQL en el PATH de Windows.

[6] Arranca por primera vez MySQL:

```
> mysqld --console
```

Tras este comando (si la configuración está bien), se quedará la consola con el mensaje: *Version: '5.6.24' socket: '' port: 3306 MySQL Community Server (GPL)* (suponiendo que hemos instalado esa versión de MySQL). El servidor MySQL está lanzado y escuchando por el puerto 3306, el habitual para MySQL.

Práctica 2.6: Instalación mediante archivos binarios de MySQL en Linux

SOLUCIÓN: PRÁCTICA 2.6

Estos son los pasos:

[1] Instala la librería *libaio*

- Linux de tipo Red Hat

```
# yum install libaio
```

- Linux de tipo Debian

```
# apt-get install libaio1
```

[2] Descarga los archivos binarios de MySQL

[2.1] **En sistemas gráficos**. Vete a la página dev.mysql.com/downloads, elegir como Sistema **Linux - Generic** y haz clic en el botón **Download** de la parte inferior, referido a binarios comprimidos en formato TAR genéricos (no RPM).

En la página siguiente podremos darnos de alta o usar un usuario y contraseña de la red de Oracle o de MySQL, si disponemos de él, o simplemente hacer clic en "*No thanks, just start my download*", que se encuentra en la parte inferior y que permite iniciar directamente la descarga.

[2.2] **Desde la consola**. Podemos, como siempre, utilizar el comando **wget**. Para instalar MySQL 5.6 en un sistema de 32 bits la descarga de los archivos se hace con:

```
# wget -O mysql.tar.gz
http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.24-linux-glibc2.5-i686.tar.gz
```

En el caso instalar la versión MySQL de 64 bits, la ruta cambia:

```
# wget -O mysql.tar.gz
http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.24-linux-glibc2.5-x86_64.tar.gz
```

En ambos casos tendremos un archivo llamado mysql.tar.gz resultado de la descarga

[3] Descomprime el archivo descargado:

```
$ tar vzxf mysql.tar.gz
```

[4] Mueve el directorio que resulta de descomprimir a la ruta */usr/local/mysql*

```
# mv mysql-5.6.24-linux-glibc2.5-x86_64/ /usr/local/mysql
```

[5] Indica el usuario y grupo propietarios de la instalación. Podemos crear un usuario o utilizar uno ya existente.

[5.1] Si queremos crear un nuevo usuario para controlar MySQL. El usuario se llamará **asir** y el grupo **mysql**:

```
# groupadd mysql
# useradd -g mysql asir
```

[5.2] Pero, es más lógico usar un usuario ya existente, bastará con añadirle al grupo creado. Así si nuestro usuario se llama **asir**, las instrucciones serían:

```
# groupadd mysql
# usermod -g mysql asir
```

[6] Otorga la propiedad del directorio al usuario root del sistema y al grupo creado anteriormente. Además el directorio de la base de datos (data) será propiedad del usuario ***asir***, que se supone será el encargado de lanzar el servidor.

```
# cd /usr/local/mysql
# chown -R root .
# chown asir data
# chgrp -R mysql .
```

[7] Lanza el script de instalación de MySQL:

```
# ./scripts/mysql_install_db --user=asir
```

[8] Lanza por primera vez MySQL. Es importante que lo hagamos sin ser el usuario root del sistema operativo. Ir al directorio **bin** dentro del directorio de instalación de MySQL y ejecutar el comando:

```
$ ./bin/mysqld_safe --user=asir &
```

La ventana de consola se queda esperando a que finalicemos el proceso

[9] Comprueba el funcionamiento de MySQL. En otra ventana de consola (con el mismo usuario **asir**), ejecutar:

```
$ /usr/local/mysql/bin/mysqldadmin version
```

Si se ejecutó correctamente MySQL, aparecerá un mensaje en el que se nos indica la versión de MySQL, la forma de conexión e incluso el tiempo que lleva en ejecución (*Uptime*). De otro modo se nos indicará un error.

[10] Desde esa misma ventana de consola, deten la ejecución de MySQL:

```
$ /usr/local/mysql/bin/mysqldadmin -u root shutdown
```

Si funciona correctamente, en la primera ventana de consola, se detendrá la ejecución de MySQL

[11] Modifica la configuración de MySQL. Para ello, edita el archivo de configuración */usr/local/mysql/my.cnf*.

```
$ sudo nano /usr/local/mysql/my.cnf
```

[12] En el archivo confirma que has indicado correctamente la ruta a la raíz de MySQL y a sus archivos de datos.

```
basedir = /usr/local/mysql
datadir = /usr/local/mysql/data
```

Si queremos otra ruta de datos, debemos indicarlo, pero previamente deberíamos copiar el contenido de */usr/local/mysql/data* a la nueva ruta de datos.

[13] En la primera ventana de consola ejecuta el comando, que esta y las siguientes veces, permitiría lanzar el servidor MySQL:

```
$ ./bin/mysqld
```

[14] Prueba la conexión desde la segunda ventana. Conecta como *root* de la base de datos:

```
$ /usr/local/mysql/bin/mysql -u root
```

[15] Ejecuta el siguiente comando SQL, para mostrar las bases de datos actuales:

```
mysql> show databases
```

[16] Sal del entorno de mysql con el comando `exit`.

[17] Asegura la instalación ejecutando la utilidad `mysql_secure_installation`. Para ello seguimos esta secuencia:

```
$ cd /usr/local/mysql
$ ./bin/mysql_secure_installation
```

Tras ejecutar el script se realizan los siguientes pasos:

[17.1] Si el usuario root ya tiene contraseña, se nos pedirá introducirla y se nos preguntará si queremos cambiarla. De otro modo (que es lo habitual) se nos pedirá una contraseña que tendremos que introducir dos veces.

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR
ALL MySQL SERVERS IN PRODUCTION USE!  PLEASE READ EACH
STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the
current password for the root user.  If you've just
installed MySQL, and you haven't set the root password
yet, the password will be blank, so you should just press
enter here.

New password for root>
```

[17.2] A continuación se nos indica si queremos eliminar a los usuarios anónimos. Escribimos *Y* (yes) para eliminarlos.

```
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to
have a user account created for them.  This is intended
only for testing, and to make the installation go a bit
smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]
```

[17.3] Se nos pregunta si desactivamos el acceso remoto a MySQL. También es recomendable responder *Y*.

```
Normally, root should only be allowed to connect from
'localhost'.  This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? [Y/n]
```

[17.4] Se nos permite eliminar la base de datos **test**, eliminando cualquier acceso indebido a ella.

```
By default, MySQL comes with a database named 'test' that
anyone can access.  This is also intended only for test-
ting, and should be removed before moving into a produc-
tion environment.

Remove test database and access to it? [Y/n]
```

[17.5] Finalmente nos pregunta si deseamos que las acciones se lleven a cabo inmediatamente. Lo lógico es aceptar escribiendo *Y*.

```
Reload privilege tables now? [Y/n]
```

[18] Edita el archivo **php.ini** (que estará por ejemplo en /usr/local/php/lib/php.ini) para asegurarnos de que existen las siguientes líneas (lo normal es que aparezcan comentadas y que simplemente debamos quitar el punto y coma que encabeza la línea):

```
extension = php_mysql.dll
extension = php_mysqli.dll
```

[19] Crea la página **prueba_mysql.php** en la raíz de documentos de Apache (la ruta habitual será */usr/local/apache2/htdocs/prueba_mysql.php*) y escribe este código:

```
<?php
$con=mysql_connect("localhost","root","libro-iaw");
if($con) echo "Conexión correcta";
else echo "Conexión incorrecta";
?>
```

Si tras ir a la dirección localhost/prueba_mysql.php aparece el mensaje *Conexión correcta*. Tendremos ya lista una instalación a medida de Apache, PHP y MySQL.

Práctica 2.7: Instalación por paquetes de Apache, PHP y MySQL en Ubuntu 14.04 Trusty.

SOLUCIÓN: PRÁCTICA 2.7

Suponemos instalada esta versión de Ubuntu. Lo ideal sería que tuviéramos la versión Server, que funciona por defecto solo en modo consola para optimizar las labores de servidor. Suponemos que estamos conectados al sistema con un usuario normal, pero que tenemos posibilidad con el comando sudo de pasar a usuario superadministrador. Estos son los pasos:

[1] Actualiza los repositorios de Linux.

```
$ sudo apt-get update
```

[2] Instala de Apache:

```
$ sudo apt-get install apache2
```

[3] Nuestro servidor web funcionará por defecto por el puerto 80. Probamos a acceder al servidor por el puerto 80 desde un navegador. Debería aparecer la página de portada de Apache sobre Ubuntu con el texto *Apache2 Ubuntu Default Page.*

[4] Instala MySQL:

```
$ sudo apt-get install mysql-server mysql-client
```

Durante la instalación se nos pedirá la contraseña del usuario **root** de la base de datos. Es **fundamental poner una contraseña.** Como contraseña pondremos **libro-iaw**

[5] Asegura la instalación de MySQL ejecutando el script de seguridad:

```
$ mysql_secure_installation
```

Se nos pedirá la contraseña actual del usuario root, introduciremos **libro-iaw**. Después nos preguntará si deseamos cambiar la contraseña, escribiremos ***n (No)*** y pulsaremos Intro.

A las tres preguntas siguientes responderemos pulsando **Intro**, lo cual contesta afirmativamente todas las preguntas para asegurar al máximo nuestra instalación.

[6] Instalar PHP:

```
$ apt-get install php5 php-pear php5-mysql
```

[7] Reiniciamos Apache:

```
$ sudo service apache2 restart
```

[8] Crea en el directorio */var/www* la página *test.php* con este contenido:

```
<?php
$con=mysql_connect("localhost","root","libro-iaw");
if($con) echo "Conexión correcta";
else echo "Conexión incorrecta";
?>
```

[9] Accede al servidor web por el puerto 80 (le suponemos abierto) y añade la ruta a la página anterior. Por ejemplo si la IP del servidor es la *192.168.100.1*, escribiremos en nuestro navegador *http://192.168.100.1/test.php.*

Si la instalación es correcta aparecerá el texto *Conexión correcta*

Práctica 2.8: Instalación por paquetes de Apache + PHP + MariaDB en CentOS 7

SOLUCIÓN: PRÁCTICA 2.8

En este caso instalaremos MariaDB. Suponemos instalado CentOS Server en su versión 7 (sin instalar modo gráfico para realmente simular un servidor real). Pasos a realizar:

[1] Instala Apache:

```
$ sudo yum install httpd
```

[2] Arranca Apache:

```
$ sudo httpd -k start
```

Tras esta instrucción, Apache estará funcionando

[3] Nuestro servidor web funcionará por defecto por el puerto 80. Probamos acceder al servidor por el puerto 80 desde un navegador. Debería aparecer la página de portada de CentOS 7 sobre Apache con el texto ***Testing 1 2 3***. Si no es así hay que examinar el cortafuegos por si está cerrado el puerto 80.

[4] Instala MariaDB:

```
$ yum install mariadb-server mariadb
```

[5] Inicia MariaDB

```
$ mysqld
```

[6] Asegura la instalación de MariaDB ejecutando el script habitual de seguridad de MySQL:

```
$ mysql_secure_installation
```

Nos pedirá la contraseña actual; pulsaremos Intro sin más ya que no habrá contraseña. Después nos pide cambiar la contraseña del usuario root, responderemos *Y (yes)* e introduciremos como contraseña del usuario ***root***, el texto ***libro-iaw*** .

A las tres preguntas siguientes responderemos pulsando **Intro**, lo cual contesta afirmativamente todas las preguntas para asegurar al máximo nuestra instalación (ver paso 17 de la práctica anterior).

[7] Haz que MariaDB y Apache se carguen en el inicio del sistema automáticamente:

```
$ sudo systemctl enable httpd.service
$ sudo systemctl enable mariadb.service
```

[8] Reinicia el sistema

```
$ sudo shutdown -r now
```

[10] Instala PHP

```
$ yum install php php-mysql
```

[11] Reiniciar Apache:

```
$ sudo systemctl restart httpd.service
```

[12] Crear en el directorio /var/www/html la página test.php con este contenido:

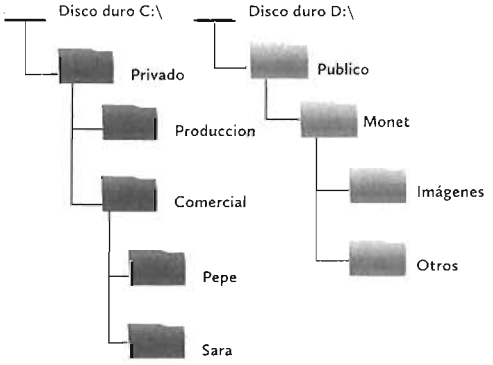
```
<?php
$con=mysql_connect("localhost","root","libro-iaw");
if($con) echo "Conexión correcta";
else echo "Conexión incorrecta";
?>
```

[13] Acceder al servidor web por el puerto 80 (le suponemos abierto) y acceder a la página anterior. Por ejemplo si la IP del servidor es la *192.168.100.1*, escribiremos en nuestro navegador ***http://192.168.100.1/test.php.*** Si la instalación es correcta aparecerá el texto ***Conexión correcta.***

2.9 PRÁCTICAS PROPUESTAS

Práctica 2.9: Instalación de Apache en Windows con restricción de acceso a directorios

- Observa la siguiente estructura de directorios.



- Utiliza un ordenador físico o una máquina virtual con Windows instalado en la que dispongas de dos unidades (C: y D:) de disco duro, desinstala cualquier instalación previa de Apache si la hubiera.
- Haz una nueva instalación Con Apache en el directorio **d:\servidorweb** y prueba que funciona correctamente en el puerto 80.
- Crea la estructura anterior de directorios.

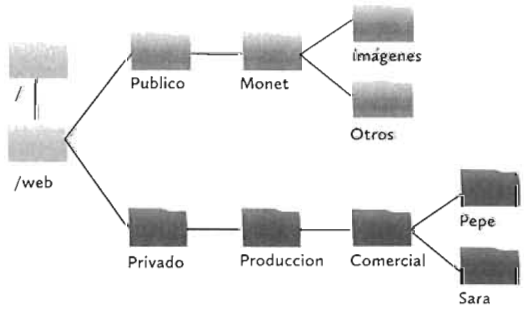
- Consigue que las carpetas **Privado** y **Publico** sean raices de dos servidores web virtuales. El público atenderá por el puerto **80** y el privado por el **8080**.Ten en cuenta la siguiente estructura de usuarios

USUARIO	GRUPO
Antonio	Produccion
Carmen	Produccion
Henar	Dirección
Pepe	Comercial
Sara	Comercial

- Consigue que solo los usuarios de la lista tengan acceso a las carpetas privadas (ponles una contraseña a cada uno), además que las carpetas de cada grupo solo las vean los usuarios de ese grupo y las personales, solo la persona en concreto. Finalmente las personas de dirección pueden ver cualquiera de las carpetas en la zona privada.
- En la zona publica, los archivos y carpetas dentro de Monet están disponibles en la dirección http://www.jorgesanchez.net/libro-iaw.

Práctica 2.10: Instalación de Apache en Linux con restricción de acceso a directorios

- Observa la siguiente estructura de directorios



- Utiliza un ordenador físico o una máquina virtual con Linux instalado (no importa la distribución).
- Desinstala cualquier instalación previa de Apache.
- Instala una nueva instalación de Apache en el directorio **/sw/apache** y prueba que funciona correctamente en el puerto 80.Instala el módulo PHP en el directorio **/sw/php** y asegúrate de que funciona.
- Crea la estructura anterior de directorios.
- Consigue que las carpetas **Privado** y **Publico** sean raices de dos servidores web virtuales. El público atenderá por el puerto **80** y el privado por el **8080.**

- Observa la siguiente estructura de usuarios

USUARIO	GRUPO
Antonio	Produccion
Carmen	Produccion
Henar	Dirección
Pepe	Comercial
Sara	Comercial

- Consigue que solo los usuarios de la lista tengan acceso a las carpetas privadas (ponles una contraseña a cada uno), además que las carpetas de cada grupo solo las vean los usuarios de ese grupo y las personales, solo la persona en concreto. Finalmente las personas del grupoi de ***dirección*** pueden ver cualquiera de las carpetas en la zona privada.
- En la zona publica, los archivos y carpetas dentro de Monet están disponibles en la dirección http://www.jorgesanchez.net/libro-iaw.
- Personaliza los mensajes de error correspondientes a los códigos de ***archivo no encontrado*** y ***acceso prohíbido***.
- Prohíbe el acceso en cualquiera de los dos servidores a los archivos que tengan extensión **.sql**

Práctica 2.11: Instalación de PHP y Apache en Windows con creación de servidores virtuales

- Utiliza una máquina local o virtual con Windows instalado. Desinstala cualquier instalación previa de Apache.
- Instala Apache en **c:\apache** y PHP en **c:\php5**. Probar que funcionan ambas cosas.
- Consigue que el servidor responda a la dirección www.prueba.local.
- Haz que la página de portada muestre el mensaje **Bienvenido a mi servidor** y luego un enlace para ir a la zona privada (que será el destino http://www.privado.local), el enlace no funcionará al principio.
- Consigue que al escribir la dirección ww.privado.local aparezca la página **index.html** que habrás creado en la dirección **c:\www.privado.**
- Logra que cuando el usuario escriba **www.servidor.local** en su navegador, aparezca lo mismo que con **www.prueba.local.**
- Hacer que al escribir **www.prueba.local/jorge** saltemos automáticamente a la dirección **www.jorgesanchez.net**.
- Prohíbe que al usar una URL referida a un directorio, si en ese directorio no hay página index, que aparezca un error y no se muestre el listado del directorio.

- Haz que el mensaje **Forbidden** que aparece cuando accedemos a un directorio sin index, se muestre el texto **Contenido no permitido. Fallo en la autentificación.** Y cuando no se encuentre un determinado contenido que se muestre el mensaje **Contenido no encontrado.**
- Consigue que Apache grabe en los logs absolutamente todos los errores y warnings.
- Haz que el acceso al servidor privado realmente sea privado, de modo que solo puedan acceder el **usuario1** (contraseña **12345**) y el usuario2 (contraseña **54321**).

Práctica 2.12: Instalación de XAMPP en un sistema de consola Ubuntu Server

- Consigue una máquina con una instalación de Ubuntu Server.
- Instala en ella XAMPP con al menos Apache, PHP y MySQL.
- Haz que la página de índice por defecto se llame **index.php** y si no, **index.html.**
- Asegura la instalación de MySQL de modo que el usuario root tenga como contraseña el texto **libro-iaw,** elimina los accesos remotos y las bases de datos de prueba y test.
- Consigue que los datos de MySQL se guarden en el directorio **/opt/data.**
- Crea un servidor virtual en el puerto 8080 que muestre el contenido del directorio **/var/www.**
- Prohíbe que Apache se muestre el contenido de un directorio al que le faltan los archivos de índice nos muestre el contenido del directorio.
- Haz que solo se pueda acceder al servidor web por el puerto 8080 si el origen de la petición procede de nuestra red local.
- Haz que los errores de código PHP no se muestren en pantalla.

Práctica 2.13: Instalación de un sistema LAMP en una máquina virtual Vagrant Ubuntu

- Vagrant es una excelente herramienta que merece una práctica para hacernos idea de su funcionamiento. En Internet encontrarás instrucciones paras solucionar problemas parecidos al que se propone.
- Consigue que funcione Vagrant en tu máquina
- Prepara una máquina Vagrant que utilice el box ***Hashicorp/Precise32***
- Prepara un archivo de configuración que instala Apache, PHP y MySQL por paquetes y que comparta el directorio /var/www de la máquina Vagrant con un directorio que llamarás html y que crearás en el directorio de la máquina vagrant.
- En el archivo de configuración, consigue también mapear el puerto 9000 para que a través de la dirección localhost:9000 puedas acceder al puerto 80 de la máquina virtual Vagrant.

2.10 RESUMEN DE LA UNIDAD

- Los desarrolladores de aplicaciones web necesitan instalar numerosas herramientas como son: el sistema operativo, los servidores necesarios (web, aplicaciones y base de datos), conectores, herramientas de edición de código, depuradores, software de máquina virtual, sistemas de control de versiones y de test.
- En la creación profesional de aplicaciones se utiliza un modelo de tres estados en el que se distingue el entorno de prueba local (***Development***), un entorno de simulación lo más real posible (***Staging***) y el entorno final en producción (***Production***). De esa forma la aplicación se publica con mayores garantías.
- Para implementar una aplicación web se debe elegir el Sistema Operativo, Servidor de aplicaciones y Servidor de bases de datos en base a las necesidades que tengamos. El marco más habitual sigue siendo la combinación Linux+Apache+PHP+MySQL (LAMP).
- **Apache** se puede instalar de diversas formas. La forma apropiada depende del entorno en el que estemos. Durante la unidad se ha instalado la instalación por código fuente en Linux (las más recomendable en un entorno de producción), la instalación por paquetes y la instalación mediante un ejecutable en Windows.
- La configuración de Apache permite delimitar su funcionamiento y es fundamental conocer las principales directivas y configuraciones, para asegurar y personalizar la instalación.
- **PHP** se puede instalar también de diferentes formas: por paquetes, por código fuente y mediante archivos binarios comprimidos, todas ellas se han estudiado durante la unidad. El archivo php.ini es el encargado de configurar su funcionamiento.
- El servidor **MySQL** (o su versión totalmente abierta, **MariaDB**) se puede instalar también por paquetes (en Linux) o mediante instalador (Windows).
- Hay soluciones que facilitan la instalación de todos los elementos que, aunque no son las recomendables en un entorno de producción, son muy cómodas para trabajar en la fase de desarrollo o para el aprendizaje. La más popular actualmente es XAMPP que se puede instalar tanto en Windows como en Linux. La instalación en ambos sistemas se ha visto a lo largo de esta unidad.

2.11 TEST DE REPASO

1.- **¿Qué son Git, Subversial y Mercure?**

a) Software de máquina virtual
b) Sistemas de control de versiones
c) Servidores de aplicaciones web
d) Entornos de desarrollo

2.- **¿Cuál es el servidor web más utilizado en Internet?**

a) Apache
b) nginx
c) IIS
d) LiteSpeed
e) Node.js

3.- **¿Cuál es el servidor web más utilizado en los sitios más populares de Internet?**

a) Apache
b) nginx
c) IIS
d) LiteSpeed
e) Node.js

4.- **Entre los servidores web de Internet ¿Qué sistema operativo es el más popular?**

a) Windows
b) Linux/Unix
c) Mac OSX
d) HP UX
e) Solaris

5.- **Tanto en Windows como en Linux ¿Cuál es el comando habitual para lanzar el servidor Apache?**

a) apache start
b) httpd
c) httpd -k start
d) net start apache2

6.- **¿Qué software necesitamos como prerrequisito en una instalación Windows de Apache mediante archivos binarios comprimidos?**

La plataforma .NET de Microsoft
Un intérprete de Perl, como Strawberry Perl
El compilador de Visual C++ Redistributable para Visual Studio
Las librerías APR y APR-util
La librería PCRE

7.- **¿Cuáles de estas librerías son necesarias para instalación de Apache mediante su código fuente en Linux?**

APR
APR-util
PCRE
make
gcc-c++
gcc

8.- **¿Cuál es el nombre del comando que permite instalar un nuevo paquete en el sistema Ubuntu?**

a) yum install
b) zypper install
c) pacman -S
d) apt-get install

9.- **¿Qué nombre tienen los archivos que permiten establecer configuraciones de Apache para un solo directorio?**

a) httpd.conf
b) httpd.dir
c) directory.conf
d) .htaccess

10.- **¿Cuáles de estos elementos son realmente imprescindibles en un sistema que implemente aplicaciones web?**

a) Servidor web
b) Sistema de control de versiones
c) Entorno de desarrollo
d) Servidor de aplicaciones web
e) Servidor de bases de datos
f) Conector a base de datos
g) Software de máquina virtual

Supongamos que deseamos que nuestro servidor escuche por los puertos 80 y 8080 ¿Qué directiva del archivo httpd.conf lo conseguiría?

Listen 80, 8080

b) Listen 80
Listen 8080

c) Port 80,8080

d) Port 80
Port 8080

¿Qué directiva permite que el acceso a un determinado recurso del servidor esté restringido a un determinado grupo de usuarios?

a) Allow

b) Deny

c) Require

d) AuthType

AuthUserFile

¿Qué directiva permite indicar que el archivo índice del directorio sea aquel que se llame index.php y si no que se utilice index.html?

DirectoryIndex index.php index.html

DirectoryIndex index.html index.php

DirectoryIndex index.php, index.html

DirectoryIndex index.html, index.php

¿Cómo se llama el archivo de configuración de PHP?

php.ini

php.conf

php.ini-development

php.conf-development

En una instalación de software por código fuente ¿Qué comando nos permite establecer la configuración de la instalación?

make

make install

configure

detail

En una instalación de MySQL en Linux,¿Cuál es el último archivo de configuración que se lee?

/etc/my.cnf

/etc/mysql/my.cnf

SYSCONFDIR/my.cnf

$MYSQL_HOME/my.cnf

~/.my.cnf

En el caso de algunos Linux, MySQL no forma parte de los repositorios estándar ¿Qué opción de base de datos abierta nos serviría igualmente por ser compatible con MySQL?

Oracle Database

b) Maria DB

c) SQLite

d) Tiny SQL

e) Open MySQL

18.- ¿En qué sistemas operativos se puede instalar XAMPP?

a) Linux

b) Windows

c) Mac OSX

d) Android

19.- ¿Cuál es el comando que permite asegurar una instalación de MySQL en Linux?

a) mysql_admin

b) mysqld

c) mysql_secure_install

d) mysql -o install

20.- En Windows existe ese mismo comando, pero ¿en qué formato de archivo está?

a) Python

b) Perl

c) Shellscript

d) .bat

UNIDAD 3

PROGRAMACIÓN BÁSICA DE APLICACIONES CON PHP

OBJETIVOS

- Describir el funcionamiento del lenguaje PHP
- Describir las herramientas necesarias para programar en PHP
- Entender los tipos de datos y el funcionamiento de las variables en el lenguaje PHP
- Entender y aplicar las estructuras de control de flujo en PHP
- Comprender el funcionamiento de la recogida de datos de un formulario desde una página PHP
- Crear páginas PHP sencillas que utilicen los elementos básicos del lenguaje

CONTENIDOS

3.1 ¿QUÉ ES PHP?

3.1.1 LENGUAJES DE SCRIPT DE SERVIDOR

Desde sus inicios, las páginas web se crean mediante el lenguaje de etiquetado HTML. El problema es que HTML es un lenguaje muy limitado para atender a los requerimientos que actualmente se exigen en las páginas web (interactividad, personalización, almacenamiento de información, etc.). Por ello se ha necesitado incorporar nuevos lenguajes a HTML. Entre ellos CSS para dar formato al contenido o JavaScript para dar interactividad al mismo.

Aunque estos lenguajes permiten dotar de una gran interactividad y apariencia a las páginas web, tienen el inconveniente de que su código es visible, lo que les incapacita para muchas tareas; por ejemplo no son válidos para el acceso a bases de datos, ya que dejarían a la vista los datos de conexión. Otro inconveniente es que dependen de la potencia y capacidades del ordenador del cliente.

Por ello aparecieron una serie de lenguajes que añaden código incrustado en el HTML, pero que se interpreta en el lado del servidor. A esos lenguajes se les conoce como **lenguajes de script de servidor**.

La idea es que el cliente reciba una página HTML normal y que sea un servidor (que deberá tener la capacidad de entender e interpretar este lenguaje) el que traduzca el código script. Al cliente no se le exige ningún plugin especial en el navegador; de hecho, no distingue que estas páginas sean especiales, salvo por la extensión del documento que en lugar de ***.html*** será, por ejemplo, ***.php***.

Todo el proceso en el lado del servidor queda oculto al usuario que en ningún momento ve el código script incrustado, dotando al código de una capa de seguridad muy interesante.

3.1.2 PHP

Se trata indudablemente, todavía a día de hoy, del lenguaje para crear contenidos web en el lado del servidor más popular. Fue el primero en aparecer, aunque realmente empezó a imponerse en torno al año 2000, por delante de ASP que era la tecnología de servidor reinante hasta ese momento.

Hoy en día se pueden instalar módulos para interpretar PHP en casi todos los servidores de aplicaciones web. En especial, PHP tiene una gran relación con el servidor web Apache.

PHP es un lenguaje con una sintaxis basada principalmente en C y en Perl, que se ha diseñado pensando en darle la máxima versatilidad y facilidad de aprendizaje, evitando la rigidez y coherencia semántica de los lenguajes clásicos. Es decir, se ha potenciado su facilidad por encima del seguimiento de reglas estrictas, propias de lenguajes más estructurados, como por ejemplo **Java**.

Eso hace que PHP sea el lenguaje favorito de miles de desarrolladores en todo el mundo y también que sea criticado por desarrolladores procedentes de lenguajes más formales, a los que no les gusta esa falta de rigidez propia de la sintaxis de PHP.

Actualmente, aunque otros muchos lenguajes crecen en número de programadores, lo cierto es que sigue siendo el lenguaje favorito para programar en el lado del servidor. La página **w3techs**[1] indica que es utilizado por el 82% de aplicaciones web programadas en el lado del servidor.

3.1.3 VENTAJAS DE PHP

[1] **Multiplataforma.** A diferencia de otros lenguajes de script, como por ejemplo **ASP** y **ColdFussion**, se trata de un lenguaje que se puede lanzar en casi todos los servidores y sistemas (**Windows**, **Unix**, **Linux**, **Solaris**, **Mac**...)

[2] **Abierto y gratuito.** PHP posee licencia de uso de tipo **GNU**, la licencia del sistema Linux; lo que permite su distribución gratuita y que su extensa comunidad de desarrolladores mejore el código sin depender de ninguna compañía privada.

[3] **Gran comunidad de usuarios.** La popularidad de PHP, junto con la gran defensa que de él hacen la comunidad de programadores de código abierto, permite tener una amplia y dinámica red de consulta a la que acudir en caso de necesidad.

[4] **Apache, MySQL.** Apache es el servidor web y de aplicaciones más utilizado en la actualidad. MySQL es el servidor de bases de datos relacionales más popular en Internet para crear aplicaciones web (aunque poco a poco va perdiendo relevancia). Puesto que PHP tiene una gran relación y compatibilidad con ambos productos (está, de hecho, muy pensado para hacer tándem con ellos), esto se convierte en una enorme (y a veces determinante) ventaja.

[5] Esta combinación unida a usar un servidor Linux se conoce como **LAMP** y es la combinación ***servidor web-servidor de aplicaciones-servidor de base de datos*** más popular, por lo menos a día de la escritura de este libro.

[6] **Extensiones.** PHP dispone de un enorme número de extensiones que permiten ampliar las capacidades del lenguaje, facilitando la creación de aplicaciones web complejas.

[7] **¿Fácil?** Es un punto muy controvertido. Los programadores PHP entusiastas, defienden esta idea. Es indudable que fue uno de los objetivos al crear este lenguaje. Sin embargo, Microsoft defiende con energía que ASP permite crear aplicaciones web complejas con mayor facilidad y parece indudable que el lenguaje **ColdFussion** de Macromedia (ahora propiedad de la empresa **Adobe**) es más sencillo de aprender (aunque tiene otros inconvenientes).

Las características de PHP correspondientes a la libertad de creación y asignación de valores a variables, tipos de datos poco restrictivos, y otras ausencias de reglas rígidas son los puntos que defienden los programadores de PHP para valorar su facilidad de aprendizaje. Los programadores de lenguajes formales como C, C++ o Java, seguramente se encontrarán con más problemas que ventajas al aprender este lenguaje.

Lo que parece indudable, es que PHP es un buen lenguaje para aquellas personas que desean crear aplicaciones web y no tienen conocimientos de programación.

1 http://w3techs.com/technologies/overview/programming_language/all

3.2 HERRAMIENTAS PARA LA ESCRITURA DE APLICACIONES EN PHP

Entendiendo que ya tenemos un servidor de aplicaciones PHP instalado según lo visto en el Apartado 2.5 *"Instalación y configuración de PHP para Apache"*, en la página 70, escribir código PHP solo requiere un editor de texto. Con lo que programas como el **bloc de notas** de Windows o los editores **nano** o **vi** de Linux, serían suficientes.

No obstante, lo lógico, es utilizar software más especializado en crear documentos PHP. En este sentido tenemos las siguientes opciones:

- **Editores de código multipropósito.** Son editores de texto especializados en crear código fuente para diversos lenguajes. Son capaces de adaptarse al tipo código escrito. Por ejemplo si escribimos código **JavaScript** el coloreado del texto se adapta a este lenguaje; si escribimos código PHP, el coloreado será distinto porque reconoce la sintaxis particular de este lenguaje.

  Tienen la ventaja de ser software muy ligero y rápido, que podríamos utilizar para otros fines, aparte de escribir aplicaciones web; por ejemplo, editar los archivos de configuración de un sistema o escribir texto puro. No incorporan muchas utilidades para PHP, pero en la mayoría de casos podemos instalar plugins y componentes para mejorar nuestro trabajo en PHP con ellos.

  Ejemplos de este software son **Sublime Text**, **Atom**, **Text Mate**, **Coda** o **Notepad++.**

- **Entornos de desarrollo integrado (IDE).** Se trata de un software más completo que incluye todo lo necesario para escribir aplicaciones PHP: coloreado del código, corrección automática, ayuda con la sintaxis, depuración, publicación de la aplicación en los tres entornos de trabajo (desarrollo, prueba y producción), integración con sistemas de control de versiones, etc.

  Los IDEs más populares para programar en PHP son:

  - **Basados en Eclipse.** Eclipse es un entorno de programación, gratuito y de código abierto, de aplicaciones creado para programar fundamentalmente en el lenguaje **Java,** pero que dispone de muchas versiones y extensiones que le sitúan como plataforma para programar en diversos lenguajes. Se ha convertido en una de las plataformas de programación más populares, aunque en estos últimos años está un poco más en declive. Las extensiones o aplicaciones basadas en Eclipse más populares para programar en PHP son:

    - **Aptana.** Sea la versión solo para PHP o la completa llamada **Studio Pro** con posibilidad de escribir código en varios lenguajes, es una de las más populares. Es muy utilizada para escribir el código de aplicaciones web. Es un software gratuito.
    - **PHP Designer Toolkit (PDT).** Considerada la extensión oficial de Eclipse para programar en PHP, ha sido la más utilizada por los programadores de este lenguaje. Ahora está siendo claramente superada por la anterior. También es gratuito.

- **Zend Studio**. Dispone de un **framework** (una plataforma de trabajo) muy famosa para crear aplicaciones PHP de forma más cómoda usando el patrón MVC. Su entorno compite con los anteriores en prestaciones. Es una herramienta de pago, para estudiantes se puede obtener por 29$.

- **Netbeans**. Se trata del entorno libre de programación competidor con Eclipse y con una filosofía parecida, pero en este caso auspiciada por la empresa **Oracle** que es su propietaria. Como Eclipse se diseñó para programar en Java, pero se ha extendido como plataforma de trabajo en numerosos lenguajes.

  NetBeans dispone de una extensión para PHP e incluso se puede descargar una versión ligera de NetBeans solo para PHP. Es una de las mejores opciones para programar en PHP. Aunque pertenece a la empresa **Oracle,** es también una herramienta gratuita y de código abierto.

- **IntellyJ Idea** y **PHPStorm**. **IntellyJ Idea** es un entorno de programación para Java que compite con Eclipse y Netbeans en popularidad y potencia. Pertenece a la empresa **JetBrains** y es un software de pago. Integra todo tipo de herramientas para programar.

  De hecho, es, sin duda, uno de los mejores entornos de trabajo para crear aplicaciones web del tipo que sea, gracias a que incorpora infinidad de plugins que permiten crear código HTML 5, JavaScript, CSS, Ruby, SQL, etc. Prácticamente cualquier tecnología de creación de aplicaciones web está presente en IntelliJIDEA: por supuesto también en PHP. Por lo que es una excelente opción de trabajo.

  Si deseamos concentrarnos solo en crear aplicaciones PHP, basado en el motor de Intelli J Idea, disponemos de **PHPStorm**. Es una versión más ligera, pero solo porque se ha centrado en la creación de aplicaciones web PHP. Reconoce perfectamente HTML, CSS y JavaScript, y también casi todos los marcos (frameworks) de uso con estos lenguajes como Node.js, Angular.js, LESS, SASS, CoffeScript, Zend, etc.

  PHPStorm no es gratuita, pero dispone de opciones para obtener licencias gratuitas para profesores, estudiantes (incluso para centros educativos enteros) y para proyectos de código abierto.

- **Visual Studio**. El popular IDE de Microsoft fundamental está pensado para crear aplicaciones web fundamentalmente de tipo .NET. Reconoce los lenguajes fundamentales HTML, CSS y JavaScript. De forma nativa no trabaja con PHP, pero existen extensiones que lo permiten. Así los amantes de esta plataforma la pueden utilizar también para crear aplicaciones PHP..

- **Herramientas de desarrollo visual**. Se trata de software que permite crear documentos web dibujando. Muchos traen herramientas para poder trabajar con PHP.

  - **Adobe Dreamweaver**. Se trata del software comercial de creación de páginas web más famoso del planeta. Tiene capacidad para escribir código PHP e incluso facilidades gráficas para hacerlo. Es inferior a los anteriores en cuanto a escritura de código, pero muy superior cuando queremos concentrarnos en el formato visual.

  - **Microsoft Expression Web**. Software comercial competidor del anterior (aunque por ahora mucho menos popular) y con capacidad de usar con PHP.

3.3 BASES DE PHP

El código PHP se escribe en un documento de texto al que se debe indicar la extensión **.php**. En realidad se trata de una página web HTML (o XHTML) normal a la que se le añaden etiquetas especiales.

3.3.1 AYUDA DE PHP

La dirección http://php.net nos lleva al sitio oficial de ayuda de PHP. Además, la dirección http://php.net/manual/es/ nos muestra el manual oficial en español. Cualquier función o elemento del que deseemos saber su funcionamiento podremos buscarlo en esta página.

Hay que tener en cuenta que, a pesar del enorme esfuerzo que se está haciendo en traducir los contenidos, siempre está más completa la ayuda en inglés.

ACTIVIDAD 3.1:

- Desde tu navegador vete a la dirección http://php.net/manual/es/ y examina la documentación oficial de PHP en español.
- Esta página tiene que ser una de las referencias imprescindibles para tu trabajo con PHP.

3.3.2 ETIQUETA <?PHP ... ?>

El código PHP que incluyamos en un documento web debe de estar indicado de esta forma:

```
<?php
        ...código PHP
?>
```

Una etiqueta especial **<?php** se coloca donde empieza el código. Esa etiqueta se cierra con **?>**. El código PHP se coloca en la zona de la página web donde más nos interese hacerlo. Así un primer documento PHP podría ser (suponiendo que incrustamos PHP en un documento de tipo HTML 5. Ejemplo:

```
<!doctype html>
<html lang="es">
    <head>
      <meta charset="utf-8" />
        <title>Título</title>
    </head>
    <body>
     <?php
        echo  "Hola desde PHP";
     ?>
    </body>
</html>
```

La página muestra el texto *Hola desde PHP*, pero este texto ha sido escrito mediante la función de escritura **echo** del lenguaje PHP.

ACTIVIDAD 3.2:

- Descubre, buscando por Internet, qué etiquetas se usan para incrustar código en otros lenguajes de script de servidor como JSP, ASP o ColdFussion

3.3.3 HTML USA PHP Y PHP USA HTML

Como se ha visto anteriormente el código PHP se incrusta dentro del código HTML. Lo interesante es que se pueden incrustar etiquetas HTML dentro del código PHP y tiene sentido, ya que las funciones de escritura (**echo** y **print**) en realidad escriben hacia el resultado final, que en realidad es una página HTML por lo que se puede hacer algo como:

```
<?php
  echo "Estoy estudiando <strong>programación</strong>";
?>
```

3.3.4 COMENTARIOS

Dentro del código PHP se pueden hacer tres tipos de comentario:

- **Comentarios de varias líneas.** Utilizan la sintaxis del lenguaje C. Comienzan con /* y terminan por */. Ejemplo:

```
<?php
  /*
    Soy un comentario de
    varias líneas
  */
$x=10;
?>
```

- **Comentarios de una línea, estilo C++.** Se ponen tras la barra doble //. Ejemplo:

```
<?php
  $x=10; //esto es un comentario
?>
```

- **Comentarios de una línea, estilo ShellScript.** Se ponen tras la almohadilla y solo tienen efecto en la línea en la que se colocan

```
<?php
  $x=10; #esto es un comentario
?>
```

3.3.5 BASES DE ESCRITURA

Las normas básicas para escribir lenguaje PHP se basan en las de sus lenguajes padres; es decir, C y **Perl**. Son:

- Todas las líneas de código deben de finalizar con un punto y coma.
- Se puede agrupar el código en bloques que se escriben entre llaves.
- Una línea de código se puede partir o sangrar (añadir espacios al inicio) a voluntad con el fin de que sea más legible, siempre y cuando no partamos una palabra o un valor.
- PHP obliga a ser estricto con las mayúsculas y las minúsculas en algunos casos, como en el nombre de las variables. Sin embargo, con las palabras reservadas del lenguaje no es estricto. Es decir, para PHP los términos ***WHILE***, ***while*** e incluso ***wHiLe*** son sinónimos, al ser variaciones de la misma palabra reservada. Sin embargo ***$var*** y ***$VAR*** no son iguales al ser nombres de variables.

  El consejo sobre esta norma, es siempre utilizar los nombres de la misma forma y escribir el código preferentemente en minúsculas. Las mayúsculas se utilizan para dar nombre a las constantes.

3.3.6 ESCRIBIR EN LA SALIDA

Aunque hay muchas funciones de escritura (para escribir en lo que será la página final) las fundamentales son **echo** y **print**.

La función **echo** es la más utilizada y en realidad es un comando del lenguaje. Tras echo se pasa uno o más textos (más adelante diremos ***expresiones de cadena***) que cuando son literales se escriben entre comillas. Si se usa más de un texto, se separan con comas:

```
<?php
  echo "Primer texto ", "segundo texto"
?>
```

El texto saldrá seguido en la página. Hay una forma abreviada de usar ***echo*** que es:

```
<?= "Primer texto ", "segundo texto"
?>
```

Permite evitar la etiqueta **<?php** y tener que escribir el nombre de la función echo. <?= sobreentiende que lo que viene después es texto a escribir, como si tuviera la palabra echo delante. No se puede colocar código PHP de otro tipo, solo expresiones imprimibles.

Hay otra función de escritura, llamada **print** que funciona casi igual a ***echo*** , pero tiene dos importantes diferencias:

- Devuelve un valor verdadero o falso dependiendo de si se pudo escribir el texto o no (en aplicaciones complejas puede ser muy útil)

- No admite varios textos, solo uno; aunque se pueden encadenar varios con el operador punto (.)

```
print "Primer texto ", "segundo texto"; //¡error!
print "Primer texto "."segundo texto"; //arreglado
```

Cuando PHP se incrusta dentro del código HTML, hay que tener en cuenta que **echo** o **print** están escribiendo dentro del código de esa página. Por lo que hay que conocer muy bien cómo funciona HTML para entender lo que ocurrirá en la página.

3.4 VARIABLES

3.4.1 INTRODUCCIÓN A LAS VARIABLES

En todos los lenguajes de programación (y PHP no es una excepción) Las variables son contenedores que sirven para almacenar los datos que utiliza un programa. Dicho más sencillamente, son nombres que asociamos a determinados datos.

La realidad física es que cada variable ocupa un espacio en la memoria RAM del servidor. Es decir, cuando utilizamos el nombre de la variable realmente estamos haciendo referencia a un dato que está en memoria.

El nombre de las variables se conoce con el término de **identificador** de variable. Un identificador en PHP tiene que cumplir estas reglas:

- Tiene que empezar con el símbolo **$**. Ese símbolo es el que permite distinguir a una variable de cualquier otro elemento del lenguaje PHP.
- El segundo carácter puede ser el guion bajo (_) o bien una letra. La recomendación es empezar siempre con una letra, el guión bajo se utiliza para variables estándares y superglobales del lenguaje.
- A partir del tercer carácter se pueden incluir números, además de letras y el guion bajo.
- No hay límite de tamaño en el nombre.
- Por supuesto, el nombre de la variable no puede tener espacios en blanco (de ahí la posibilidad de utilizar el guion bajo).

Es conveniente que los nombres de las variables indiquen de la mejor forma posible su función. Es decir: ***$saldo*** es un buen nombre, pero ***$x123*** no lo es, aunque sea válido.

También es conveniente poner a nuestras variables un nombre en minúsculas. Si consta de varias palabras el nombre podemos separar las palabras con un guion bajo en vez del espacio o empezar cada nueva palabra con una mayúscula. Ejemplos de ambos casos son: ***$saldo_final*** y ***$saldoFinal***, ambos buenos nombres de variable.

3.4.2 DECLARAR VARIABLES

La primera sorpresa para los programadores de lenguajes estructurados es que en PHP no es necesario declarar una variable. Simplemente se utiliza y ya está. Es decir, si queremos que la variable ***$edad*** valga *15*, haremos:

```
$edad=15;
```

No es necesario indicar de qué tipo es esa variable. Esto es muy cómodo, pero también complica la tarea de depurar nuestros programas, ya que provoca que haya errores difícilmente detectables.

Aunque no hay una instrucción de declaración como tal. Sí que se deben definir las variables antes de usarlas; de otro modo, en su primer uso tendríamos un error. Por ejemplo, si $x no había sido definida con un primer valor, este código falla:

```
echo $x;
```

Con la configuración de PHP en estado de mostrar errores, que es lo normal, veríamos en pantalla el error ***Notice: Undefined variable: $x*** seguido del número de línea en el que se encuentra la instrucción.

3.4.3 ASIGNACIÓN DE VALORES

Esta operación consiste en dar valor a una variable y se realiza con el símbolo =. Ejemplo:

```
$x=12;
```

Como se comentó anteriormente, en PHP no es necesario declarar una variable antes de su uso.

En el caso de asignar números se escriben tal cual (como en el ejemplo), los decimales se separan con el punto decimal. Los textos se encierran entre comillas (simples o dobles). Ejemplo:

```
$nombre="Anselmo";
```

Se pueden asignar resultados de aplicar fórmulas mediante operadores, como:

```
$beneficios=($gastos-$ingresos) * 0.6;
```

Otra cosa sorprendente, comparado con los lenguajes más formales, es que una variable puede cambiar el tipo de datos que almacena. Así, este código es perfectamente válido:

```
$x=18; //asignamos un número
$x="Hola"; /asignamos un texto
```

3.4.4 VARIABLES SIN ASIGNAR VALORES

Un problema surge cuando queremos utilizar variables a las que no se les ha asignado ningún valor. Como:

```
<?php
   echo $x; //es la primera vez que se usa $x. Error
   $y=$x+2; //error se usa $x sin haberla definido, aunque y valdrá 2
   echo $y;
?>
```

Ocurrirá un error al hacer ese uso de la variable. Aunque en realidad la directiva **error_reporting** del archivo **php.ini,** permite modificar los errores de aviso que se lanzan. Si bajamos su nivel de aviso, no detectará estos fallos. También podemos usar una función con ese mismo nombre, al inicio de nuestro código para indicar fallos. Su uso sería:

```
error_reporting(E_ALL); //avisa de todos los errores
```

Si deseamos que no nos avise de estos fallos. Habría que pasar a ***error_reporting*** otra constante; por ejemplo **E_USER_ERROR** avisa solo si hay errores de nivel usuario (o más graves) en el código.

Hay una función llamada **isset** a la que se le pasa una variable e indica si dicha variable ha sido definida o no. Ejemplo:

```
<?php
   echo isset($x); //escribe falso, no sale nada por pantalla, porque
                   //$x no ha sido definida
   $x=12;
   echo isset($x); //devuelve verdadero, escribe 1
?>
```

3.4.5 TIPOS DE DATOS

3.4.5.1 NÚMEROS ENTEROS

A las variables se les puede asignar valores enteros. Los números enteros se usan tal cual. Pueden ser positivos o negativos:

```
$n1=17;
$n2=-175;
```

Se puede usar sistema octal si se coloca un cero antes de la cifra entra:

```
$octal=071;
echo $octal; //escribe 56, equivalente decimal a 71 en octal
```

Además pueden estar en sistema hexadecimal si a la cifra se la antecede de un cero y una equis:

```
$hexa=0xA2BC;
echo $hexa; //escribe 41660, equivalente a A2BC en hexadecimal
```

3.4.5.2 NÚMEROS EN COMA FLOTANTE

Los números decimales en PHP son de tipo coma flotante. Este es un formato para almacenar números decimales utilizado en máquinas digitales. La ventaja es que se manejan muy rápido por parte de un ordenador o cualquier otra máquina electrónica. Además ocupan poco en memoria. La gran desventaja es que no son exactos. Solo nos pueden asegurar un cierto nivel de precisión. Si hacemos operaciones complejas o utilizamos muchos decimales, el resultado ya no será exacto.

Para asignar decimales hay que tener en cuenta en PHP que el formato es el inglés, por lo que las cifras decimales se separan usando un punto como separador. Además es posible usar notación científica. Ejemplos de variables con valores decimales:

```
$n1=234.12;
$n2=12.3e-4; //significa 12,3·10⁻⁴, es decir 0,00123
```

Los números decimales en coma flotante de PHP son equivalentes al formato **double** del lenguaje C.

ACTIVIDAD 3.3:

- Averigua cuáles son los tipos numéricos permitidos en lenguaje C y compáralos con el lenguaje PHP
- Haz la misma averiguación con el lenguaje Python y toma nota de cuál de los tres lenguajes (C, Python o PHP) te parece que es más sencillo para trabajar con números.

3.4.5.3 CADENAS O *STRINGS*

Se denomina así a los textos que se utilizan en el código. Se asignan a las variables entrecomillando (en simples o dobles) el texto que deseamos almacenar en la variable. Ejemplo:

```
$nombre="Jorge Sánchez";
```

Si el propio texto requiere comillas, se pueden utilizar combinaciones de las comillas para evitar el problema. Por ejemplo:

```
$frase = 'Antonio dijo "Hola" al llegar';
```

Como queremos almacenar en ***$frase*** el texto con ***Hola*** entre comillas, entonces englobamos todo el texto con comillas simples. Otra opción es usar comillas mediante secuencias de escape:

```
$frase = "Antonio dijo \"Hola\" al llegar";
```

De esa forma, las comillas se toman carácter literal y no como el inicio o el fin de un texto.

Las secuencias de escape son útiles para representar códigos de texto que no están presentes en el teclado. Las que permite PHP son:

SECUENCIA DE ESCAPE	SIGNIFICADO
\t	Tabulador
\n	Nueva línea
\f	Alimentación de página
\r	Retorno de carro
\"	Dobles comillas
\'	Comillas simples
\\	Barra inclinada (backslash)
\$	Símbolo dólar

Las secuencias de escape solo funcionan si están encerradas entre comillas dobles. Es decir, si escribimos:

```
echo 'Antonio dijo \"Hola\" al llegar';
```

Las barras saldrán por pantalla también (no se consideran secuencias de escape).

Un hecho de PHP muy interesante es que en un string se puede incluir el valor de una variable, aunque solo si el texto está entrecomillado con comillas dobles. Por ejemplo:

```
$días=15;
$texto="Faltan $días días para el verano";
echo $texto;//escribe: Faltan 15 días para el verano
```

Por lo que si necesitamos incluir el signo $ en un texto, debemos utilizar su secuencia de escape:

```
$texto="Su precio es de 15\$";
echo $texto;//escribe: Su precio es de 15$
```

Los textos se pueden concatenar con ayuda del operador punto (.). Ejemplo:

```
$nombre="Jorge";
$apellidos="Sánchez Asenjo";
echo "Nombre completo: ".$nombre." ".$apellidos;
//escribe Nombre completo: Jorge Sánchez Asenjo
```

3.4.5.4 BOOLEANOS

Los booleanos son variables que se utilizan para almacenar resultados de comprobaciones. Solo pueden tomar como valores **true** (verdadero) o **false** (falso). En muchas casos se utilizan como ***centinelas***, variables cuyo valor indica una determinada situación.

Una simple variable booleana se utiliza así:

```
$verdadero=true;
echo $verdadero; //escribe 1
```

Sorprende que **echo** escriba el valor uno; la explicación es que el valor **true** está asociado a valores positivos, mientras que **false** se asocia al cero. En concreto, hay una relación entre todos los tipos de datos:

- **Enteros**: cero=***false***, resto=***true***
- **Coma flotante**: 0.0=***false***, resto=***true***
- **Cadenas**: ***false*** si están vacías
- **Arrays**: ***false*** si no contienen ningún elemento
- **Recurso**: ***false*** si el recurso no es válido.

3.4.5.5 CONVERSIONES

Debido a que el lenguaje PHP permite que las variables puedan tomar valores de diferente tipo cuando se nos antoje, resulta que tiene que intentar que todas las expresiones tengan sentido, y así este código:

```
$v1=18;
$v2="3 de Diciembre";
echo $v1+$v2; //Escribe 21
```

Hace que por pantalla aparezca el texto ***21*** (suma el 18 y el tres). Pero, sin embargo:

```
$v1=18;
$v2="3 de Diciembre";
echo $v1.$v2; //Escribe: 183 de Diciembre
```

El texto se encadena como si fueran dos strings normales.

Aunque, en general, PHP convierte adecuadamente las expresiones, podemos convertir de forma forzosa los valores al tipo deseado; de esta forma, elegiremos nosotros cómo realizar las conversiones. Para ello se puede utilizar el ***operador de casting,*** viejo conocido de los programadores en lenguaje C. Ejemplo:

```
$x=2.5;
$y=4;
$z=(int)$x * $y; //z vale 8
```

$z vale 8 ya que se toma de ***$x*** su parte entera, es decir el número ***dos***, y dos por cuatro resulta ocho.

Podemos usar los siguientes operadores de casting:

- **(int)** o **(integer)**. Convierte a entero
- **(real)**, **(double)** o **(flat)**. Convierte a coma flotante
- **(string)**. Convierte a forma de texto
- **(array)**. Convierte a array.
- **(object)**. Convierte a un objeto

3.4.6 REFERENCIAS &

PHP también ha tomado prestado del lenguaje C++ el operador **&**. Mediante este operador, conseguimos referencias. Las referencias son variables que en lugar de almacenar valores, se convierten en sinónimos de otras variables. Su uso principal lo veremos cuando lleguemos al Apartado 4.1.4 *"Paso de parámetros por referencia"*, en la página 170.

Con un ejemplo, intentaremos explicar su uso:

```
$nombre="Antonio";
$ref=&$nombre; //$ref es una referencia a la variable $nombre
echo $ref."<br />"; //escribe Antonio
$ref="Marisa"; //ahora se asigna el valor Marisa, a $ref
echo $nombre,"<br />"
//escribe Marisa, a través de la referencia se ha cambiado el
//valor de la variable $nombre
```

En definitiva, las referencias son variables que representan a otras variables.

3.4.7 CONSTANTES

Las constantes almacenan valores que no cambian en el tiempo. La forma de definir constantes es gracias a la función **define**. Que funciona indicando el nombre que tendrá la constante, entre comillas, y el valor que se le otorga. Ejemplo:

```
define("PI",3.141592);
```

Las constantes no utilizan el signo **$** de las variables, simplemente utilizan el nombre. Es decir, escribir el valor de la constante PI tras haberla declarado con la instrucción anterior, sería:

```
echo PI;
```

Desde la versión 5.3 de PHP es posible definir una constante de una forma más estructurada. Se trata de utilizar la palabra clave **const**, habitual en la mayoría de lenguajes de programación. Ejemplo de uso:

```
const PI=3.141592;
```

Aunque no es obligatorio, es conveniente usar mayúsculas para dar nombre a las constantes y así, diferenciarlas de las variables (aunque en PHP el signo dólar $ ya hace esa diferenciación).

3.4.8 VARIABLES DE VARIABLES

Permiten utilizar en PHP lo que se conoce como **macrosustitución de variables**. Se trata de utilizar dos veces el signo dólar ($$), de modo que se toma el contenido de la variable que está definida mediante el segundo signo de dólar y con ese contenido se da nombre a una variable. Se entiende mejor con un ejemplo:

```
$x="variable1";
$$x=5;
echo $variable1; //escribe 5
```

La línea clave es **$$x=5**, en ella el intérprete de PHP sustituye ***$x*** por su valor (***variable1***); por lo que esa expresión es equivalente a ***$variable=5***.

Aunque de entrada puede parecer una rareza de PHP sin utilidad, lo cierto es que facilita mucho la creación de ciertas variables. Por ejemplo, para conseguir crear una variable de cada elemento de un array (especialmente de **$_GET**, **$_POST**, **$_SESSION** o **$_COOKIE**).

3.4.9 OPERADORES

Lo habitual al programar en PHP es utilizar expresiones que permitan realizar comprobaciones o cálculos. Las expresiones dan un resultado que puede ser de cualquiera de los tipos de datos comentados anteriormente (enteros, decimales, booleanos, strings...)

3.4.9.1 ARITMÉTICOS

OPERADOR	SIGNIFICADO
+	Suma
-	Resta
*	Producto
/	División
%	Módulo (resto)

Ejemplo de uso:

```
$x=15.5;
$y=2;
echo $x+$y,"<br />"; //escribe 17.5
echo $x-$y,"<br />"; // escribe 13.5
echo $x*$y,"<br />"; // escribe 31
echo $x/$y,"<br />"; //escribe 7.75
echo $x%$y,"<br />"; //escribe 1, solo coge la parte entera
```

3.4.9.2 OPERADORES CONDICIONALES

Sirven para comparar valores. Siempre devuelven valores booleanos (*true* o *false*). Son:

OPERADOR	SIGNIFICADO
<	Menor
>	Mayor
>=	Mayor o igual
<=	Menor o igual
==	Igual, devuelve verdadero si las dos expresiones que compara son iguales
===	Equivalente, devuelve verdadero si las dos expresiones que compara son iguales y además del mismo tipo
!=	Distinto
!	No lógico (NOT)
&&	"Y" lógico
AND	"Y" lógico
||	"O" lógico
OR	"O" lógico
XOR	"OR" exclusivo

Los operadores lógicos (**AND**, **OR** y **NOT**), sirven para evaluar condiciones complejas. NOT sirve para negar una condición. Ejemplo:

```
$edad = 21;
$mayorDeEdad = $edad >= 18; //$mayorDeEdad será true
$menorDeEdad = !$mayorDeEdad; //$menorDeEdad será false
```

El operador && (**AND**) sirve para evaluar dos expresiones de modo que, si ambas son ciertas, el resultado será **true** si no el resultado será **false**. Ejemplo:

```
$carnetConducir=true;
$edad=20;
$puedeConducir= ($edad>=18) && $carnetConducir;
//$puedeConducir es verdadero, puesto que $edad es mayor de 18
 //y $CARNET es true
```

El operador || (**OR**) sirve también para evaluar dos expresiones. El resultado será true si al menos uno de las expresiones es true. Ejemplo:

```
$nieva =TRUE;
$llueve=FALSE;
$graniza=FALSE;
$malTiempo= $nieva||$llueve||$graniza;//malTiempo es TRUE porque nieva
```

La diferencia entre la igualdad y la equivalencia se puede explicar con este ejemplo:

```
$resultado= (18 == 18.0); //resultado es verdadero
$resultado= (18 === 18.0); //resultado es falso
```

En el ejemplo *18===18.0* devuelve falso porque aunque es el mismo valor, no es del mismo tipo.

3.4.9.3 OPERADORES DE ASIGNACIÓN

Ya se ha comentado el operador de asignación que sirve para dar valor a una variable. Ejemplo:

```
$y=3;
$x=$y+9.8;    //$x vale 12.8
```

Existen operadores que permiten resumir asignaciones. Facilitan la escritura de muchas operaciones habituales al escribir código. Un ejemplo de este tipo de asignaciones es:

```
$x += 3;
```

Es equivalente a:

```
$x = $x + 3;
```

En definitiva, lo que se hace es sumar *3* al valor que ya tenía la variable *$x*. Así tenemos todos estos operadores de asignación:

+=	-=	*=	/=
&=	|=	^=	%=
>>=	<<=	.=	

Además, disponemos de los operadores de incremento (++) y decremento (--) que añaden uno y restan uno, respectivamente, al valor de la variable. Ejemplo:

```
$x++; //equivalente a $x=$x+1;
$x--; //equivalente a $x=$x-1;
```

Pero hay dos formas de utilizar el incremento y el decremento. Se puede usar por ejemplo *x++* (post-incremento) o *++x* (pre-incremento). La diferencia estriba en el modo en el que se comporta la asignación. Ejemplo:

```
$x=5;
$y=5;
$z=$x++; // Primero asigna, luego incrementa: z vale 5, x vale 6
$z=++$y; // Primero incrementa, luego asigna: z vale 6, x vale 6
```

3.4.9.4 OPERADORES DE BIT

Permite utilizar operaciones binarias. Aplicados a números (especialmente enteros), usan su forma binaria para realizar la operación BIT a BIT.

Los operadores disponibles son:

símbolo	significado
&	AND
|	OR
~	NOT
^	XOR
>>	Desplazamiento a la derecha
<<	Desplazamiento a la izquierda

Ejemplo de uso:

```
$x=97;          //$x en binario vale 01100001
$y=74;          //$y en binario vale 01001100
echo $x & $y; //resulta:---------->01000000=64
echo "\n";
echo $x | $y; //resulta:---------->01101101=107
echo "\n";
$x = $x >>2;  //ahora $x vale 00011000 = 24
echo $x;
echo "\n";
$y = $y << 2; //ahora $y vale 100110000 = 296
echo $y;
```

3.4.9.5 CONCATENACIÓN

El punto (.) es un operador que permite encadenar Strings (textos). Su uso es muy sencillo. Ejemplo:

```
$x = "Hola ";
$y = "a todo el mundo";
$z = $x.$y; //$z vale "Hola a todo el mundo"
```

Hay operador de asignación y concatenación de texto, se trata del operador .=. Ejemplo de uso:

```
$texto  = "Hola ";
$texto .= "amigos y ";
$texto .= "amigas"; //$texto es Hola amigos y amigas
```

3.5 ESTRUCTURAS DE CONTROL

Hasta ahora las instrucciones que hemos visto, son instrucciones que se ejecutan secuencialmente; es decir, podemos saber lo que hace el programa leyendo las líneas de izquierda a derecha y de arriba abajo.

Las instrucciones de control de flujo permiten alterar esta forma de ejecución. A partir de ahora, habrá líneas en el código que se ejecutarán o no dependiendo de una **condición**. Esa condición se construye utilizando lo que se conoce como **expresión lógica**; esto es, cualquier tipo de expresión que retorne un resultado verdadero o falso.

Las expresiones lógicas se construyen a través de los operadores relacionales (==, >, <,...) y lógicos (&&,||, !,...) vistos anteriormente.

3.5.1 SENTENCIA CONDICIONAL IF

3.5.1.1 SENTENCIA CONDICIONAL SIMPLE

Se trata de una sentencia que, tras evaluar una expresión lógica, ejecuta una serie de instrucciones en caso de que la expresión lógica sea verdadera. Si la expresión tiene un resultado falso, no se ejecutará ninguna expresión. Su sintaxis es:

```
if(expresiónLógica) {
     instrucciones
}
```

Las llaves solo se requieren si va a haber varias instrucciones, de otro modo podemos obviarlas.

```
if(expresiónLógica)
     instrucción;
```

Un ejemplo de **if** típico podría ser este:

```
if($nota>=5){
   echo "Aprobado";
   $aprobados++;
}
```

Que significa que en el caso de que la variable ***$nota*** valga 5 o más, mostramos el texto ***Aprobado*** e incrementamos el valor de la variable **$aprobado**. La idea gráfica del funcionamiento del if sería:

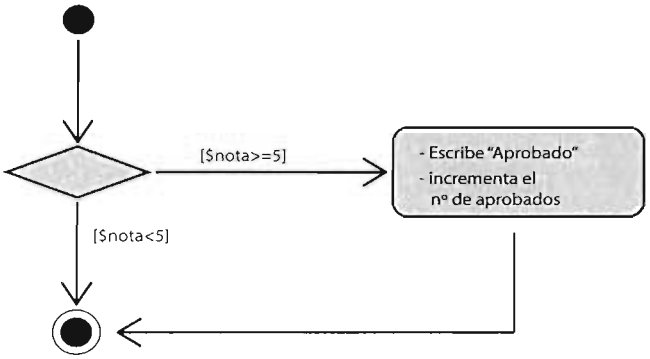


Figura 3.1: Diagrama representativo de una estructura **if** simple

Hay otra posibilidad de sintaxis para la instrucción *if*

```
if(expresión lógica) :
     instrucciones
     ...
endif;
```

En lugar de las llaves, cierra el bloque **if** con la palabra **endif**. Los dos puntos tras el paréntesis son obligatorios. Se trata de una forma simplificada que, ciertamente, hay que usar con cuidado.

3.5.1.2 SENTENCIA CONDICIONAL COMPUESTA

En ella se añade a la instrucción **if** un apartado **else** que contiene instrucciones que se ejecutarán si la expresión evaluada por el **if** es falsa. Sintaxis:

```
if(expresión lógica){
  instrucciones a ejecutar si la expresión es verdadera
  ...
}
else {
  instrucciones a ejecutar si la expresión es falsa
  ...
}
```

Como en el caso anterior, las llaves son necesarias solo si se ejecuta más de una sentencia en el bloque.

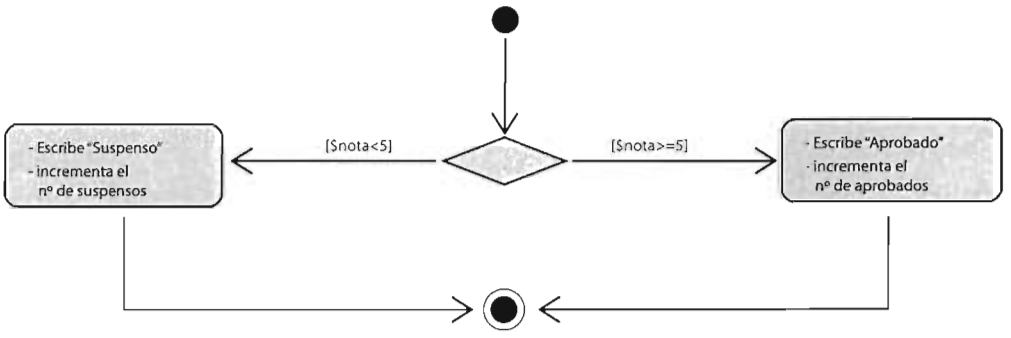


Figura 3.2: Diagrama representativo de una estructura **if** compuesta

Ejemplo de sentencia **if-else**:

```
if($nota>=5){
   echo "Aprobado";
   $aprobados++;
}
```

```
else {
   echo "Suspenso";
   suspensos++;
}
```

La forma simplificada (sin llaves) de esta sentencia sería:

```
if($nota>=5):
   echo "Aprobado";
   $aprobados++;
else :
   echo "Suspenso";
   $suspensos++;
endif;
```

3.5.1.3 SENTENCIA CONDICIONAL MÚLTIPLE

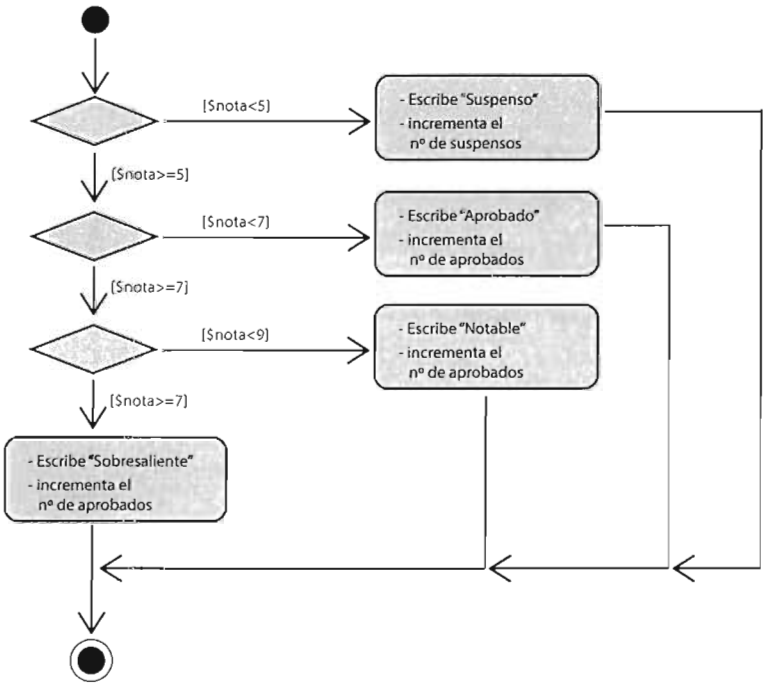


Figura 3.3: Diagrama representativo de una estructura **if** múltiple

Permite unir varios **if** en un solo bloque. Ejemplo:

```
if($nota<5){
    echo "Suspenso";
    $suspensos++;
}
```

```
elseif($nota<7){
    echo "Aprobado";
    $aprobados++;
}
elseif($nota<9){
    echo "Notable";
    $aprobados++;
}
else{
    echo "Sobresaliente";
    $aprobados++;
}
```

Cada sentencia **elseif** permite añadir una condición que se examina si la anterior no es verdadera. En cuanto se cumpla una de las expresiones, se ejecuta el código del **if** o **elseif** correspondiente. El bloque **else** se ejecuta si no se cumple ninguna de las condiciones anteriores.

La versión simplificada sería:

```
if($nota<5):
    echo "suspenso";
    $suspensos++;
elseif($nota<7):
    echo "aprobado";
    $aprobados++;
elseif($nota<9):
    echo "notable";
    $aprobados++;
else:
    echo "sobresaliente";
    $aprobados++;
}
```

SENTENCIA SWITCH

```
switch (expresión) {
  case valor1:
     instrucciones del valor 1
        [break]
  [case valor2:
     instrucciones del valor 1
     [break]
      ]
  [...]
```

Los corchetes, en la sintaxis de las instrucciones, indican contenido opcional. Es decir, los **break** son opcionales y la cláusula **default** también.

```
    [default:
       instrucciones que se ejecutan en cualquier otro caso
    ..]
}
```

Esta instrucción se usa cuando tenemos instrucciones que se ejecutan de forma diferente según evaluemos el conjunto de valores posible de una expresión. Cada **case** evalúa un posible valor de la expresión; si, efectivamente, la expresión equivale a ese valor, se ejecutan las instrucciones de esa sección case y de las siguientes. Para que no se ejecuten las instrucciones de los case adyacentes se emplea la instrucción **break,** que abandona inmediatamente el switch; de tal modo que, si queremos que para un determinado valor se ejecuten las instrucciones de un apartado **case** y solo las de ese apartado, entonces, habrá que finalizar ese **case** con un **break.**

Por otra parte, el bloque **default** sirve para ejecutar instrucciones para los casos en los que la expresión no cumpla ninguna de las condiciones de los ***case*** de la instrucción.

Ejemplo de instrucción switch:

```
switch ($diaSemana) {
   case 1:
      $dia="Lunes";
      break;
   case 2:
      $dia="Martes";
      break;
   case 3:
      $dia="Miércoles";
      break;
   case 4:
      $dia="Jueves";
      break;
  case 5:
      $dia="Viernes";
      break;
  case 6:
      $dia="Sábado";
      break;
   case 7:
      $dia="Domingo";
      break;
   default:
      $dia="?";
}
```

No usar break para salir del switch puede permitir crear estructuras switch que ejecuten unas mismas instrucciones para diferentes valores.

En el código siguiente el texto ***Laborable*** se muestra cuando tenemos un día 1, 2, 3, 4 o 5. Si es 6 o 7 se escribe ***Festivo***. Finalmente, ante cualquier otro valor se escribe una interrogación.

```
switch ($diaSemana) {
   case 1:
   case 2:
   case 3:
   case 4:
   case 5:
      echo "Laborable";
      break;
   case 6:
   case 7:
        echo "Festivo";
      break;
   default:
      echo "?";
}
```

3.5.2 BUCLES

Un bucle es un conjunto de sentencias que se repiten mientras se cumpla una determinada condición. Su uso es prácticamente imprescindible para escribir aplicaciones.

3.5.2.1 BUCLE WHILE

Sintaxis:

```
while (expresión_lógica) {
     sentencias que se ejecutan si la condición es true
}
```

Un bucle **while** sigue estos pasos:

[1] Se evalúa la expresión lógica.

[2] Si la expresión es verdadera ejecuta las sentencias, si no el programa abandona la sentencia **while.**

[3] Tras ejecutar las sentencias, volvemos al paso 1.

Ejemplo, bucle que escribe números del 1 al 100:

```
$i=1;
while ($i<=100){
   echo $i."<br />";
   $i++;
}
```

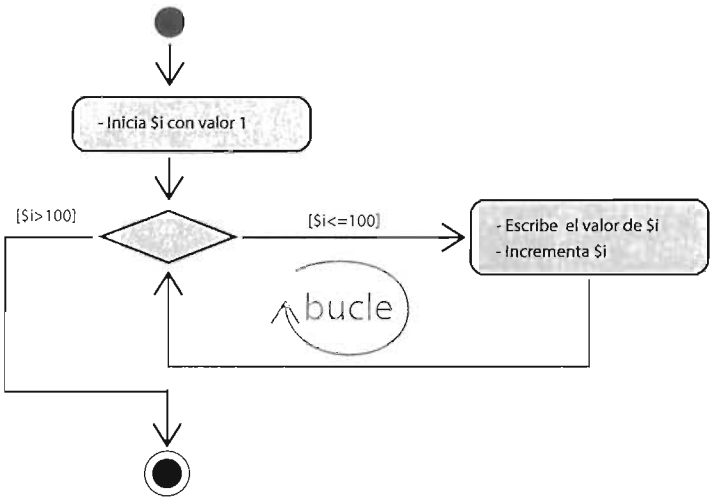


Figura 3.4: Diagrama representativo de una estructura **while**

También la instrucción **while** tiene también una versión sin llaves. En ella, el ejemplo anterior se escribiría:

```
$i=1;
while ($i<=100):
        echo $i."<br />";
        $i++;
endwhile;
```

Los bucles **while** son los fundamentales. Aunque PHP dispone de otros bucles, todos ellos se pueden realizar mediante sentencias while. Por ello, es fundamental saber utilizar este tipo de bucle. A continuación, se comentan los dos principales tipos de bucles y su implementación mediante la estructura while.

Bucles de contador

Se llaman así a los bucles que se repiten una serie determinada de veces. Dichos bucles están controlados por un **contador** (o incluso más de uno). El contador es una variable que va modificando su valor, de uno en uno, de dos en dos,... o como queramos, en cada vuelta del bucle. Cuando el contador alcanza un límite determinado, entonces termina.

En todos los bucles de contador necesitamos saber:

[1] Lo que vale la variable contadora al principio. Antes de entrar en el bucle

[2] Lo que varía (lo que se incrementa o decrementa) el contador en cada vuelta del bucle.

[3] Las acciones a realizar en cada vuelta del bucle

[4] El valor final que debe de tomar el contador para abandonar el bucle. Dicho valor se pone como condición del bucle, pero a la inversa; es decir, la condición mide el valor que tiene

que tener el contador para que el bucle se repita y no para que termine. Si un bucle no define bien la condición de salida, podría convertirse en un **bucle infinito**.

Ejemplo:

```
$i=10; // Valor inicial del contador, empieza valiendo 10

while ($i<=200){  /* condición del bucle, mientras $i sea
                  menor de 200, el bucle se repetirá,
                  cuando $i rebase este
                 valor, el bucle termina */

   echo $i."<br />";     /* acciones que ocurren en cada
                  vuelta del bucle. En este caso
                  simplemente se escribe el valor
                  del contador y un elemento HTML
                  de salto de línea */

   $i+=10; /* Variación del contador, en este caso
            cuenta de 10 en 10*/
}
/* Al final el bucle escribe:
10
20
30
...
y así hasta 200
*/
```

En la práctica la mayoría de programadores optan por un estructura **for** (que se explica más adelante) para codificar este tipo de bucle.

Bucles de centinela

Es el segundo tipo de bucle básico. Una condición lógica llamada **centinela**, que puede ser desde una simple variable booleana hasta una expresión lógica más compleja, sirve para decidir si el bucle se repite o no. De modo que cuando la condición lógica se incumpla, el bucle termina.

Esa expresión lógica a cumplir es lo que se conoce como centinela y normalmente la suele realizar una variable booleana. La diferencia con los bucles de contador, es que no sabemos de antemano las veces que se repite el bucle. Si el centinela jamás toma el valor de salida, el bucle será infinito.

Son bucles que, inicialmente, son más difíciles de aplicar. Sin embargo, cuando se domina su uso, se convierten rápidamente en un tipo de sentencia fundamental para resolver problemas complejos de programación.

Ejemplo de bucle centinela:

```
$salir=false; /* En este caso el centinela es una
              variable booleana que inicialmente
              vale falso */
while($salir==false){ /* Condición de repetición:
          que $salir siga siendo
          falso. Ese es el centinela.
          También se podía haber escrito simplemente:
          while(!salir)
          */
   $n=mt_rand(1,500); /* En cada vuelta, calculamos
   echo($n);             un número aleatorio de 1 a 500
                         y le escribimos */
   $salir=($n%7==0); /* El centinela vale verdadero
                        si el número es múltiplo de 7 */
}
```

En definitiva, el código anterior muestra continuamente números aleatorios entre 1 y 500 hasta que aparezca un múltiplo de 7.

Comparando los bucles de centinela con los de contador, podemos señalar estos puntos:

- Los bucles de contador se repiten un número concreto de veces, los bucles de centinela no.
- Un bucle de contador podemos considerar que es seguro que finalice, el de centinela puede no finalizar si el centinela jamás varía su valor; aunque, si está bien programado, siempre finalizará.
- Un bucle de contador está relacionado con la programación de algoritmos basados en series.

Un bucle podría ser incluso mixto: de centinela y de contador. Por ejemplo sería el caso de un código que escriba números de uno a 500 y se repita hasta que llegue un múltiplo de 7, pero que como mucho se repita ocho veces.

El código sería el siguiente:

```
$salir = false;  //centinela
$i=1;            //contador
while ($salir == false && $i<=8) {
    $n = mt_rand(1,500);
    echo $n."<br />";
    $i++;
    $salir = ($n % 7 == 0);
}
```

3.5.2.2 DO...WHILE

La única diferencia, respecto a los bucles while, está en que la expresión lógica se evalúa después de haber ejecutado las sentencias. Es decir, el bucle al menos se ejecuta una vez. Su funcionamiento consiste en

[1] Ejecutar sentencias

[2] Evaluar expresión lógica

[3] Si la expresión es verdadera volver al paso 1, sino continuar fuera del while

Sintaxis:

```
do {
     instrucciones
} while (expresión lógica)
```

Ejemplo (contar de 1 a 100):

```
$i=0;
do {
   $i++;
   echo $i."<br />";
} while ($i<100);
```

Esquema:

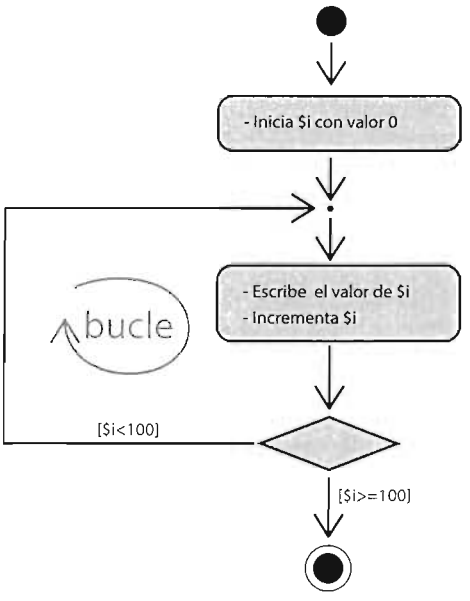


Figura 3.5: Diagrama representativo de una estructura do..while

Cualquier estructura **do..while** se puede convertir en **while**. Así, el código anterior se puede escribir usando la instrucción while de esta forma:

```
int i=0;
$i++;
echo $i."<br />";

while (i<100) {
   $i++;
   echo $i."<br />";
}
```

3.5.2.3 BUCLE FOR

Es un bucle más complejo, especialmente pensado para recorrer arrays o para implementar bucles de contador. Es el bucle más utilizado por los programadores ya que es muy cómodo de escribir y, por su estructura, es más difícil cometer errores. Puede que al principio sea difícil de entender, pero en poco tiempo se aplica perfectamente a todo tipo de aplicaciones. Sintaxis:

```
for(inicialización;condición;incremento){
     sentencias
}
```

Como en todas las estructuras de control, disponemos de una versión simplificada:

```
for(inicialización;condición;incremento):
     sentencias
endfor;
```

Las sentencias se ejecutan mientras la condición sea verdadera. Además, antes de entrar en el bucle, se ejecuta la instrucción de inicialización y en cada vuelta se realiza el incremento de la variable contadora. Es decir el funcionamiento es:

[1] Se ejecuta la instrucción de inicialización

[2] Se comprueba la condición

[3] Si la condición es cierta, entonces se ejecutan las sentencias. Si la condición es falsa, se abandona el bloque **for**

[4] Tras ejecutar las sentencias, se ejecuta la instrucción de incremento y se vuelve al paso 2

Ejemplo (contar números del 1 al 1000):

```
for($i=1;$i<=1000;$i++){
   echo $i."<br />";
}
```

La ventaja que tiene es que el número de líneas de código se reduce. La desventaja es que el código es menos comprensible.

El bucle anterior es equivalente al siguiente bucle **while:**

```
$i=1; // sentencia de inicialización
while($i<=1000) { // condición
   echo $i."<br />";
   $i++; // incremento
}
```

3.6 USO DE FORMULARIOS HTML DESDE PHP

3.6.1 ENVÍO DE DATOS DESDE UN FORMULARIO

Los formularios son el elemento de las páginas web que permiten recoger valores del usuario y enviarlos a una dirección URL para que se les procese. En nuestro caso la URL será una página PHP que recogerá los valores introducidos en el formulario y actuará en consecuencia.

HTML dispone de numerosos controles para almacenar información que se pide al usuario: cuadros de texto, cuadros numéricos, botones de radio, cuadros combinados, etc. En este manual no se enumeran dichos controles, ya que entendemos que contamos con los conocimientos de HTML necesarios para crear formularios. Pero, al menos, con los formularios debemos tener en cuenta lo siguiente:

- Todos los controles del formulario deben estar dentro de un elemento **form**, el cual nos permite configurar estos atributos:
  - **method**. Puede ser **get** o **post** y se refiere a la forma en la que se pasan los datos de formulario. Mas adelante, se explica la diferencia entre ambos métodos. Por defecto se toma como método, **get**.
  - **action**. Que indica la URL a la dirección del servicio que recoge los datos (por ejemplo ***http://www.jorgesanchez.net/anadirUsr.php***)
- Dentro del elemento form se colocan los elementos de HTML capaces de recoger información. Son los elementos **input**, **option**, **textarea**, etc. Los cuales poseen un atributo llamado **name**, en el que se indica el nombre de la variable que se envía al servidor, y otro llamado **value** que, inicialmente, permite indicar el valor que se asigna a dicha variable, dicho valor puede ser modificado por el usuario en el formulario.

3.6.2 MÉTODOS DE ENVÍO DE DATOS DEL FORMULARIO

Los formularios envían los datos mediante pares nombre/valor. Es decir, indican un nombre de variable, que se corresponderá con el indicado mediante el atributo **name** de un control del formulario, y un valor para la misma; y esto se repite para cada uno de los valores que el formulario sea capaz de enviar.

Como se ha comentado antes, un formulario puede enviar la información usando el atributo **method** del formulario y eligiendo **GET** o **POST**.

3.6.2.1 PASO DE DATOS MEDIANTE GET

El método GET lo que hace es añadir a la URL destinataria del formulario los nombres y valores recogidos en el formulario.

Ejemplo:

```
<form action="recogida1.php" method="get">
   <label for="nombre">Escriba su nombre</label>
   <input type="text" name="nombre" /> <br />
   <label for="apellidos">Escriba sus apellidos</label>
   <input type="text" name="apellidos" /> <br />
   <label for="nombre">Escriba su dirección</label>
   <input type="text" name="direccion" /> <br />
   <label for="nombre">Escriba sus teléfonos:</label>
  <br />
   <input type="text" name="telefono1" maxlength="9"
     pattern="[0-9]{9}" /> <br />
   <input type="text" name="telefono2" maxlength="9"
     pattern="[0-9]{9}" /> <br />
   <input type="submit" value="enviar"/><br />
</form>
```

El resultado del código, rellenando los campos, sería:

Escriba su nombre Jorge
Escriba sus apellidos Sanchez
Escriba su dirección Asenjo
Escriba sus teléfonos:
66666666
99999999
enviar

Suponiendo que estamos probando el código en una página web con dirección http://localhost/form11.html, al pulsar enviar, se genera la URL:

http://localhost/recogida1.php?nombre=Jorge&
apellidos=S%C3%A1nchez+Asenjo&direccion=C%2F+Los+Vientos
&telefono1=666666666&telefono2=999999999

El apellido ***Sánchez*** se convierte en **S%C3%A1nchez** por que en la URL no puede haber caracteres fuera del código ASCII y el carácter se recodifica usando los dígitos Unicode dentro del operador %.

Dejando de lado la cuestión de la codificación, la forma de una URL tipo GET es:
http://urlpágina?nombre1=valor1&nombre2=valor2&. . . . .

Es decir, usa pares entre el nombre de la variable (en el formulario asignada a través del atributo **name**) y el valor que se le dio en el formulario. Cuando un control del formulario se queda sin valor, entonces la variable se queda sin definir y, por lo tanto, la página PHP receptora deberá comprobar si se rellenó o no el control mediante la función **isset**.

Supongamos, con el ejemplo anterior, que la página receptora, en la dirección http://localhost/pruebas/recogidar.php tiene el codigo siguiente_

```
<!DOCTYPE HTML>
<html lang="es-ES">
<head>
   <meta charset="utf-8" />
   <title></title>
</head>
<body>
   <pre>
   <?php
      print_r ($_GET);
    ?>
   </pre>
</body>
</html>
```

Esa página, si recibe los datos de ejemplo de este manual, escribiría:

```
Array
(
    [nombre] => Jorge
    [apellidos] => Sánchez Asenjo
    [direccion] => C/ Los Vientos
    [telefono1] => 666666666
    [telefono2] => 999999999
)
```

Es decir, un array que contiene cada elemento pasado desde el formulario. Cada elemento tiene como clave el nombre y como valor, el valor que el formulario asocia a ese nombre. Acceder a un valor concreto del array se haría, por ejemplo, mediante: `$_GET["nombre"]`

Si intentamos acceder mediante `$_GET["nombre"]` al nombre de un valor que no hemos recibido, entonces obtendremos el error: `Notice: Undefined index:`

Para evitar este error, antes de acceder al nombre debemos comprobar si existe mediante:

```
if(isset($_GET["nombre"])
```

El paso de datos de un formulario mediante GET tiene estas connotaciones:

- Cuando se rellenan los datos en un formulario y éste envía los datos mediante una petición GET, la URL que contiene los pares nombre/valor rellenados en el formulario está a la vista. Evidentemente, no debemos pasar de esta forma contraseñas y otros datos críticos.
- Las páginas generadas con GET pueden añadirse como marcador o favorito a los navegadores y páginas de marcadores. Eso puede ser interesante, pero solo debe de ser posible si la URL no contiene datos que hagan que otro usuario que obtenga esa URL pueda acceder a información confidencial.
- No tenemos que pasar por el formulario para enviar valores a la página receptora. Ya que en la propia barra de navegación podremos escribir los valores y nombres a mano. En principio no tiene por qué ser un problema, de hecho, puede ser una forma alternativa de acceder a una página y, a la vez, enviar información que modifique su resultado; pero también es una puerta abierta para que se puedan enviar a la página datos no controlados con intenciones, digamos, malévolas.
- La depuración de nuestros programas es más cómoda si usamos GET, porque vemos lo que realmente llega del formulario en la propia URL sin tener que acudir a otras herramientas. Una forma muy habitual de trabajar es pasar parámetros de forma GET durante el desarrollo de la aplicación y luego hacerlo vía POST en la versión final.

3.6.2.2 PASO DE PARÁMETROS MEDIANTE POST

Figura 3.6: Datos enviados vía POST, tal cual se ven en el apartado *Network* de las *DevTools* del navegador Google Chrome (disponibles con la tecla F12)

Los formularios se crean igual en ambos casos, pero ahora cambiamos el atributo **method** de la etiqueta **form** para que tome el valor **post**.

Mediante POST, los nombres y valores del formulario no viajan en la URL, sino que se colocan en la cabecera del paquete http; por lo que les dotamos de una menor visibilidad. Naturalmente podemos verlos utilizando herramientas que nos permitan examinar las cabeceras http del paquete que se envía por la red. Hoy en día es sencillo, ya que desde los propios navegadores disponemos de herramientas para ello.

3.6.3 RECEPCIÓN DE DATOS DE UN FORMULARIO DESDE UNA PÁGINA PHP

PHP permite recibir los datos pasados por los parámetros usando dos variables, que son en realidad dos ***arrays*** que contendrán todos los valores del formulario. Los arrays se llaman **$_GET** y **$_POST**, cada uno de ellos dedicado a almacenar los datos enviados con el método correspondiente. Hay otro array, de uso menos recomendable, llamado **$_REQUEST** que recoge los valores sin importar si se han pasado por GET o por POST.

Puesto que son arrays (los arrays se estudian con profundidad más adelante) su manejo es distinto al de una variable normal. Podemos entender que un array es un contenedor de varios valores. Así, los arrays $_GET o $_POST son un almacén de todos los nombres y valores enviados desde el formulario.

Los controles del formulario envían los valores asociados al nombre que dimos a ese control, explicitado en el atributo **name**. De modo que si el método de paso era POST, la página PHP accederá al valor de ese control mediante:

```
$_POST["name"]
```

Donde ***name*** es el nombre del control.

Puede ser que el control no reciba ningún valor (porque el usuario deje sin uso ese control); en ese caso la invocación a **$_POST** (o a **$_GET**) usando el nombre de ese control, provocará un error de ***variable indefinida***, por lo que debemos comprobar la existencia de ese nombre mediante la función **isset**.

Otras veces lo que ocurrirá es que se recibe como definida la variable, pero no contiene valores. En ese caso habría que utilizar la función **empty** para comprobar que no está vacía.

3.6.4 USAR LA MISMA PÁGINA PARA EL FORMULARIO Y LA RECEPCIÓN

A veces se usa la misma página en la que está el formulario, para recibir los datos del mismo. En ese caso la página del formulario será PHP, en el otro caso puede ser un código HTML ya que los formularios se pueden crear sin necesidad de líneas de código PHP. Un uso habitual de esta técnica, es validar los datos desde PHP. La página recibe los datos y les valida; si están bien, entonces se dan por buenos; sino, informa del error.

Actualmente las validaciones de los datos se realizan por **JavaScript** y especialmente usando **AJAX**; esta técnica permite validar los datos de forma más directa (sin tener que reenviarles). Pero implica usar intensamente JavaScript y para validaciones complejas seguirán utilizando técnicas en el lado del servidor.

En el siguiente ejemplo se expone una página que pide un nombre de usuario y contraseña en un formulario. Si son correctos (usuario ***Jorge***, contraseña ***123456***) se indicará que lo son; sino, un enlace nos enviará de nuevo al formulario para volverlo a rellenar.

La cuestión es ¿cómo sabe la página web cuándo mostrar el formulario y cuándo no? La forma es valorar la existencia dentro del array **$_POST** (lo mismo ocurriría para **$_GET**) de las claves ***nombre*** y ***contraseña*** a través de la función ***isset***. Cuando esa función devuelve falso, es que a la página no se le ha pasado el nombre y la contraseña vía POST, con lo que mostraremos el formulario.

```
<!DOCTYPE html>
<html lang="es">
   <head>
        <meta charset="UTF-8" />
        <title>Ejemplo de formulario en la propia página</title>
    </head>
    <body>
<?php
if(isset($_POST["nombre"] && isset($_POST["contra"]) {
//si no se reciben nombre y contraseña, se muestra el formulario
?>
       <!--Este código HTML está dentro del IF de PHP -->
        <form action="form1.php" method="post">
          <label for="nombre">Escriba su usuario</label>
            <input type="text" name="nombre" maxlength="20" /><br />
            <label for="contra">Escriba la contraseña</label>
            <input type="password" name="contra"maxlength="20" />
            <br />
            <input type="submit" value="enviar"/><br />
        </form>
<?php }//cierre del if
        else {//Se han pasado datos, se procesan
          $nombre=$_POST["nombre"];
          $contra=$_POST["contra"];
          if($nombre!="Jorge" || $contra!="123456") {
            echo "No es válido ese usuario y contraseña<br>";
            echo "<a href='form1.php'>Volver al formulario</a>";
          }
          else  echo "Entrada correcta";
        }
  }  ?>
    </body> </html>
```

3.7 REDIRIGIR HACIA OTRA PÁGINA

Es muy habitual al hacer aplicaciones web que ante una determinada condición, deseemos cargar otra página diferente a la que estamos. El método más sencillo con PHP es utilizar la función **header**.

Esta función es muy poderosa, su finalidad es enviar datos de cabecera de paquetes http sin formato. Gracias a ella podemos utilizar la cabecera **Location** del protocolo http que provocará que se cargue otra página. Por ejemplo, podemos lanzar inmediatamente la página del buscador Google con el código:

```
header("Location:http://www.google.es");
```

3.8 PRÁCTICAS RESUELTAS

RECOMENDACIÓN:

Para muchas prácticas se necesita utilizar la función mt_rand, la cual recibe dos números y retorna, como resultado, un número aleatorio entre el primero y el segundo. Por ejemplo, `mt_rand(10,20)` devuelve un número aleatorio entre 10 y 20 (ambos incluidos).

Práctica 3.1: Imagen aleatoria

- Crea una página PHP que muestre de forma aleatoria dos imágenes. Es decir, se muestra una u otra de forma aleatoria e impredecible.

SOLUCIÓN: PRÁCTICA 3.1

Para realizar la práctica suponemos ya instalado el entorno de trabajo y, además, que hemos descargado las dos imágenes en un directorio. Supondremos también que las imágenes se llaman *imagen1.jpg* e *imagen2.jpg*. El código de la página sería:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Imagen aleatoria</title>
</head>
<body>
<?php
$numero=mt_rand(0,1);
if($numero==0){
   echo "<img src='imagen1.jpg'>";
}
else{
   echo "<img src='imagen2.jpg'>";
}
?>
</body>
</html>
```

Práctica 3.2: Cálculo de salario

- Lee el nombre, los apellidos, el salario (número con decimales) y la edad de una persona (un número) en un formulario. Recoge los datos y con ellos calcula un nuevo salario para esa persona en base a esta situación:
  - Si el salario es mayor de 2000 euros, no cambiará

- Si el salario está entre 1000 y 2000:
  - Si además la edad es mayor de 45 años, se sube un 3%
  - Si la edad es menor de 45 o igual, se sube un 10%
- Si el salario es menor de 1000
  - Los menores de 30 años cobrarán, a partir de ahora, exactamente 1100 euros
  - De 30 a 45 años, sube un 3%
  - A los mayores de 45 años, sube un 15%

SOLUCIÓN: PRÁCTICA 3.2

En este caso necesitamos crear dos documentos. El primero es el que realiza el formulario de lectura de datos (a este archivo le llamamos ***form-practica2.php***):

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Formulario</title>
</head>
<body>
   <form action="practica2.php">
      <label for="nombre">Nombre</label>
      <input type="text" id="nombre" name="nombre"><br>
      <label for="apellidos">Apellidos</label>
      <input type="text" id="apellidos" name="apellidos"><br>
      <label for="edad">Edad</label>
      <input type="number" id="edad" name="edad"><br>
      <label for="salario">Salario</label>
      <input type="number" id="salario" name="salario"><br>
      <button>Enviar</button>
   </form>
</body>
</html>
```

El formulario envía la información al archivo **practica2.php** cuyo código es:

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Mostrar cálculo</title>
```

```
    <style>
    .error{
      color:red;
    }
    </style>
</head><body>
<?php
if(isset($_GET["nombre"]) && isset($_GET["apellidos"])
   && isset($_GET["edad"]) && isset($_GET["salario"])){
   $nombre=$_GET["nombre"];
   $apellidos=$_GET["apellidos"];
   $edad=$_GET["edad"];
   $salario=$_GET["salario"];
   if($salario<1000){
      if($edad<30) $salario=1100;
      elseif($edad<=45) $salario*=1.3;
      else $salario*=1.15;
   }
   elseif($salario<=2000){
      if($edad>45) $salario*=1.03;
      else $salario*=1.1;
   }
   echo "$nombre $apellidos, tu salario será de $salario &euro;";
}
else{
   echo "<p class='error'>Faltan datos</p>";
   echo "<a href='form-practica2.html'>Volver</a>";
}
?>
</body>
</html>
```

Práctica 3.3: Saber si hay número

- Crea un formulario que lea un número, después un mensaje nos indicará si era realmente o no un número y, si es un número, si tenía decimales.

SOLUCIÓN: PRÁCTICA 3.3

Como en la práctica anterior, primero creamos un documento HTML con un formulario que sirva para leer el número. Le llamamos *form-practica3.html.*

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Decimales</title>
</head>
<body>
<form action="practica3.php">
    <label for="n">Escriba un número</label>
    <input type="number" step="0.001" name="n" id="n"/><br/>
    <button>Enviar</button>
</form>
</body>
</html>
```

A continuación creamos el archivo PHP que procesará el número (se llama *practica3.php*). Hay que observar el uso de la función **is_numeric** para saber si realmente hemos recibido un número y de **header** para cambiar automáticamente de página.

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Comprobar número</title>
</head>
<body>
<?php
if(isset($_GET["n"])){//se ha recibido el número
    $n=$_GET["n"];//por comodidad usamos la variable $n
    if(is_numeric($n)){//$n contiene realmente un número
        $resto=$n-(int)$n;//quitamos la parte entera al número
        if($resto==0)
            echo "<h1>El número es entero</h1>";
        else
            echo "<h1>El número es decimal</h1>";
    }else{
        echo "<h1>No se ha recibido un número</h1>";
    }
    echo "<a href='form-practica3.html'>Volver</a>";
}else{
    //no hay número, volvemos al formulario
    header("location:form-practica3.html");
}
```

```
?>
</body>
</html>
```

Práctica 3.4: Fondo aleatorio

- Crea una página PHP que ponga de fondo un color aleatorio
- Para ello recuerda que en CSS el color de fondo se puede realizar mediante la función `rgb` a la que se le pasan tres números del 0 al 255, el primero es el nivel de rojo, el segundo el de verde y el tercero el de azul.

SOLUCIÓN: PRÁCTICA 3.4

El código que soluciona este problema es:

```
<!doctype html>
<html lang="es">
<head>
    <?php
    //cálculo aleatorio de cada nivel de color
    $rojo=mt_rand(0,255);
    $verde=mt_rand(0,255);
    $azul=mt_rand(0,255);
    ?>
    <meta charset="UTF-8">
    <title>Color aleatorio</title>
    <style>
        body{
           background-color:
                  <?="rgb($rojo,$verde,$azul);"?>
        }
    </style>
</head>
<body>
</body>
</html>
```

Práctica 3.5: Asteriscos

- Crea un formulario en el que se pida un número entero positivo. Después, haz que la página escriba tantos asteriscos (en la misma página) como el número que se haya escrito. Si se escribe 5, se mostrarán 5 asteriscos.

SOLUCIÓN: PRÁCTICA 3.5

Para solucionar esta práctica, el código podría ser el siguiente:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Asteriscos</title>
    <style>
        .error{
            color:red;
        }
    </style>
</head>
<body>
<form action="practica5.php">
    <label for="n">Escriba el número de asteriscos</label>
    <input type="number" name="n" id="n" min="1"/><br/>
    <button>Enviar</button>
</form>
<?php
//comprobamos si la página recibe correctamente el
//número de asteriscos
if(isset($_GET["n"])){
    $n=$_GET["n"];
    if(is_numeric($n) && $n>=1){
        //los datos son correctos, escribimos los asteriscos
        echo "<p>";
        for($i=1;$i<=$n;$i++){
            echo "*";
        }
        echo "</p>";
    } else{
        echo "<p class='error'>Número incorrecto</p>";
    }
}
?>
</body>
</html>
```

Práctica 3.6: Creación de tabla

- Crea un formulario que pida dos números. Ambos tienen que valer 1 o más, de no ser así se indica el error.

- El resultado será una tabla (se mostrará en la misma página del formulario) con el tamaño indicado

SOLUCIÓN: PRÁCTICA 3.6

Como la tabla se muestra en la misma página que en la que está el formulario, el formulario envía los datos a sí mismo. Suponemos que la página se llama ***practica5.php***, el código podría ser el siguiente:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Tabla</title>
    <style>
        .error{
            color:red;
        }
        td{
            border:1px solid black;
        }
    </style>
</head>
<body>
<form action="practica6.php">
    <label for="columnas">Escriba el número de columnas</label>
    <input type="number" name="columnas" id="columnas" min="1"/><br/>
    <label for="filas">Escriba el número de filas</label>
    <input type="number" name="filas" id="filas" min="1"/><br/>
    <button>Enviar</button>
</form>
<?php
//comprobamos si la página recibe los parámetros para
//dibujar la tabla
if(isset($_GET["filas"]) && isset($_GET["columnas"])){
    $filas=$_GET["filas"];
    $columnas=$_GET["columnas"];
    if(is_numeric($filas) && is_numeric($columnas) &&
        $filas>=1 && $columnas>=1){
        //los datos son correctos, dibujamos la tabla
        echo "<table>";
        for($i=1;$i<=$filas;$i++){
            echo "<tr>";
```

```
            for($j=1;$j<=$columnas;$j++) {
                //en cada celda escribimos un espacio en blanco
                //para asegurar que se muestre
                echo "<td> </td>";
            }
            echo "</tr>";
        }
        echo "</table>";
    }
    else{
        echo "<p class='error'>".
             "Número de filas y/o columnas incorrecto</p>";
    }
}
?>
</body>
</html>
```

3.9 PRÁCTICAS PROPUESTAS

Práctica 3.7:

- Crea una página PHP que muestre los números del 1 al 1000 pero de forma que aparezcan en 5 columnas y se lean de izquierda a derecha. Ejemplo de resultado (se muestran las primeras y las últimas filas):

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
.....				
986	987	988	989	990
991	992	993	994	995
996	997	998	999	1000

Práctica 3.8:

- Crea un formulario que pida al usuario un número.
- Después, en otra página, recoge ese número y muestra la suma de todos los números pares anteriores a él.
- Por ejemplo, si el usuario escribe el número 9 saldría por pantalla el número 20, resultado de sumar 2 + 4 + 6 + 8

- Mejorar el resultado para que la página que muestra la suma, después muestre un enlace con el que regresar al formulario de modo que, al hacer clic en él, el cuadro de entrada del número muestre el último número introducido

Práctica 3.9:

- Crear una página PHP que muestre por pantalla todo el código ASCII en una tabla de 16 columnas.
- Como pista, la función `chr`, recibe un número y muestra el código ASCII equivalente. Así `chr(65)` muestra el carácter *A*.

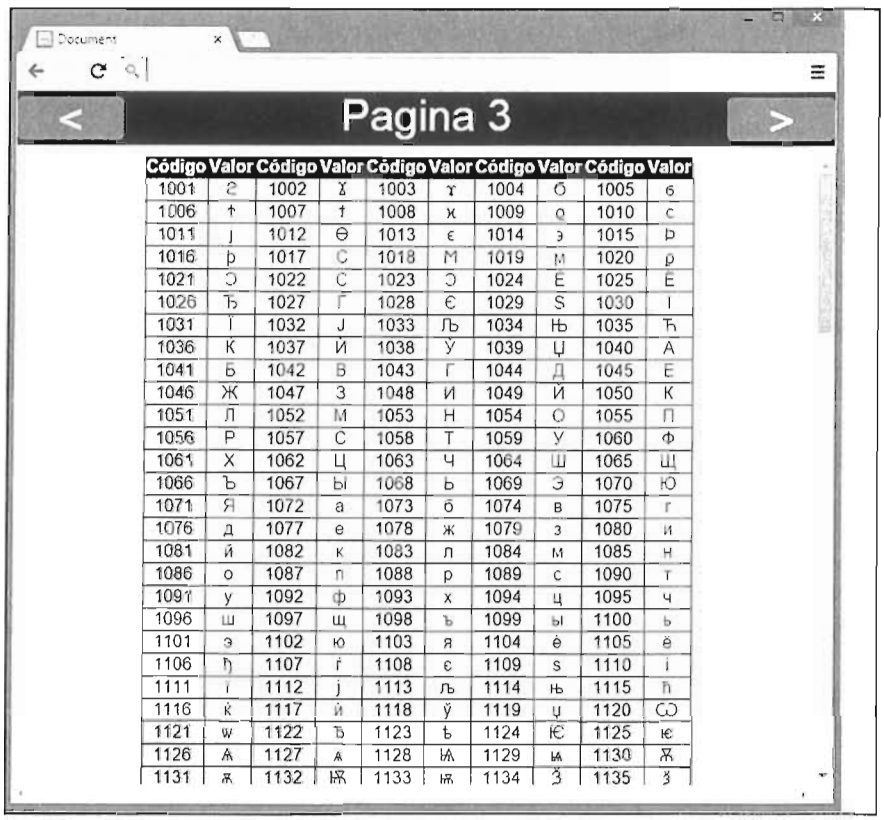
Código	Valor	Código	Valor	Código	Valor	Código	Valor	Código	Valor	Código	Valor	Código	Valor	Código	Valor
1		2		3		4		5		6		7		8	
9		10		11		12		13		14		15		16	
17		18		19		20		21		22		23		24	
25		26		27		28		29		30		31		32	
33	!	34	"	35	#	36	$	37	%	38	&	39	'	40	(
41	)	42	*	43	+	44	,	45	-	46	.	47	/	48	0
49	1	50	2	51	3	52	4	53	5	54	6	55	7	56	8
57	9	58	:	59	;	60	<	61	=	62	>	63	?	64	@
65	A	66	B	67	C	68	D	69	E	70	F	71	G	72	H
73	I	74	J	75	K	76	L	77	M	78	N	79	O	80	P
81	Q	82	R	83	S	84	T	85	U	86	V	87	W	88	X
89	Y	90	Z	91	[	92	\	93	]	94	^	95	_	96	`
97	a	98	b	99	c	100	d	101	e	102	f	103	g	104	h
105	i	106	j	107	k	108	l	109	m	110	n	111	o	112	p
113	q	114	r	115	s	116	t	117	u	118	v	119	w	120	x
121	y	122	z	123	{	124	|	125	}	126	~	127			

Práctica 3.10:

- Mejora la práctica anterior para mostrar los caracteres de la tabla Unicode del 0 al 50000
- En este caso es mejor usar esta idea: en HTML si escribimos *&#* seguido de un número y del símbolo de punto y coma (*;*), nos muestra el carácter Unicode correspondiente a ese número. Así el código *ñ* en una página web, muestra el carácter *ñ*.

Práctica 3.11:

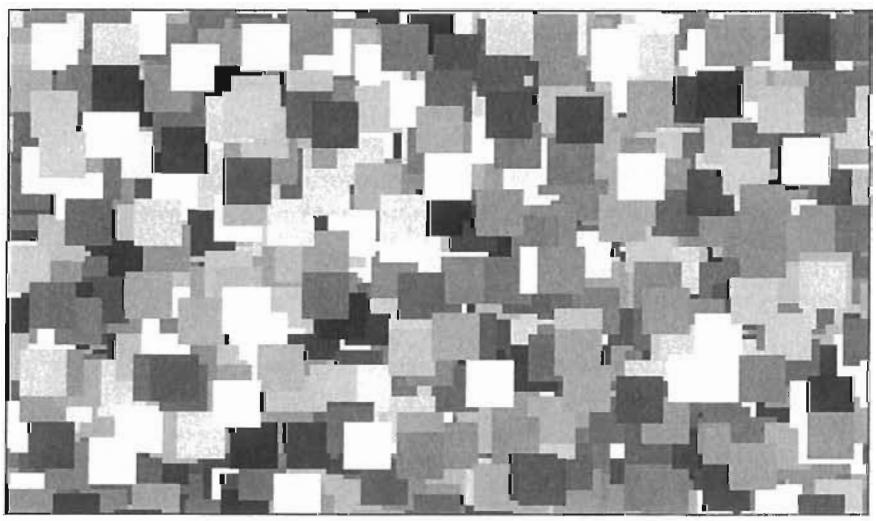
- Como última mejora (muy avanzada), haz que la tabla anterior se pueda mostrar paginada. De modo que en cada pantalla se muestren 500 códigos y unos enlaces permitan pasar a la siguiente página.
- Ejemplo de salida (si hemos pulsado dos veces al enlace de avance de página):



Código	Valor	Código	Valor	Código	Valor	Código	Valor	Código	Valor
1001	Ғ	1002	ҳ	1003	ҳ	1004	Ө	1005	ө
1006	ҧ	1007	ҭ	1008	ҳ	1009	ҩ	1010	ҫ
1011	ӈ	1012	Ө	1013	є	1014	ҽ	1015	Ҏ
1016	ҏ	1017	Ҁ	1018	Ҍ	1019	ҍ	1020	ҏ
1021	Ɔ	1022	Ҁ	1023	Ɔ	1024	Ё	1025	Ё
1026	Ҍ	1027	Ґ	1028	Є	1029	S	1030	І
1031	Ї	1032	J	1033	Љ	1034	Њ	1035	Ћ
1036	К	1037	Й	1038	Ў	1039	Џ	1040	А
1041	Б	1042	В	1043	Г	1044	Д	1045	Е
1046	Ж	1047	З	1048	И	1049	Й	1050	К
1051	Л	1052	М	1053	Н	1054	О	1055	П
1056	Р	1057	С	1058	Т	1059	У	1060	Ф
1061	Х	1062	Ц	1063	Ч	1064	Ш	1065	Щ
1066	Ъ	1067	Ы	1068	Ь	1069	Э	1070	Ю
1071	Я	1072	а	1073	б	1074	в	1075	г
1076	д	1077	е	1078	ж	1079	з	1080	и
1081	й	1082	к	1083	л	1084	м	1085	н
1086	о	1087	п	1088	р	1089	с	1090	т
1091	у	1092	ф	1093	х	1094	ц	1095	ч
1096	ш	1097	щ	1098	ъ	1099	ы	1100	ь
1101	э	1102	ю	1103	я	1104	ѐ	1105	ё
1106	ђ	1107	ѓ	1108	є	1109	ѕ	1110	і
1111	ї	1112	ј	1113	љ	1114	њ	1115	ћ
1116	ќ	1117	ѝ	1118	ў	1119	џ	1120	Ѡ
1121	ѡ	1122	Ѣ	1123	ѣ	1124	Ѥ	1125	ѥ
1126	Ѧ	1127	ѧ	1128	Ѩ	1129	ѩ	1130	Ѫ
1131	ѫ	1132	Ѭ	1133	ѭ	1134	Ѯ	1135	ѯ

Práctica 3.12:

- Crea una página que muestre 2000 cuadrados de colores aleatorios que se coloquen también de forma aleatoria por la pantalla. Cada cuadrado deberá medir 50x50 píxeles
- Ejemplo de resultado:



3.10 RESUMEN DE LA UNIDAD

- Hay diversos lenguajes de scripts, cuyo código se interpreta en el lado del servidor. PHP es el más popular aporta una sintaxis muy poco estricta, similitud con los lenguajes C y Perl y una enorme librería de trabajo.
- PHP sigue siendo el lenguaje de scripts más utilizado y sus ventajas son numerosas, lo que le ha otorgado una enorme influencia entre los desarrolladores de aplicaciones web.
- Hay numerosas herramientas para escribir código PHP. Hay otros editores de texto multipropósito (**Sublime Text, Coda, Notepad++, Atom...**), entornos de desarrollo (**Aptana, PDT, Netbeans, PHP Storm...**) y herramientas de diseño visual (como **Dreamweaver**).
- Los archivos PHP deben tener extensión .php y su código estar entre las etiquetas **<?php** y **?>**.
- Las bases de la escritura en PHP son sencillas con reglas muy habituales entre los lenguajes de programación.
- Las variables de PHP deben empezar con el carácter $, se las debe asignar un valor antes de poderse utilizar y admiten almacenar resultados de expresiones complejas,
- Los tipos de datos básicos en PHP son enteros, decimales de coma flotante y texto (strings). PHP puede convertir los datos de un tipo a otro automáticamente, pero, a veces, hay que hacerlo a mano mediante los operadores de casting u otro tipo de componente.
- La asignación de valores PHP admite el uso del operador de preincremento y postincremento, así las acciones de operar y asignar a la vez.
- PHP dispone de numerosos operadores para trabajar: aritméticos, lógicos, concatenación, de bit, etc.
- PHP dispone de las clásicas estructuras de control if, while, switch y for, Lo que permite crear todo tipo de estructuras y bucles.
- Los datos de un formulario se pueden enviar a un documento PHP tanto por GET como por POST (e incluso por otros comandos http como PUT o DELETE). Los arrays $_GET o $_POST o $_REQUEST almacenan los datos enviados.
- La función header permite enviar paquetes con cabeceras http El uso más común es redireccionar el flujo hacia otra página mediante el comando **Location.**

3.11 TEST DE REPASO

¿Qué expresión nos permite saber si una determinada página PHP está recibiendo datos vía GET de un control de formulario llamado *apellido1*?

a) `if($_GET["apellido1"]) ...`
b) `if(isset($_GET["apellido1"])) ...`
c) `if(is_def($_GET["apellido1"])) ...`
d) `if(is_defined( $_GET["apellido1"] )) ...`

2. ¿Cuál es la tecnología del lado del servidor más utilizada actualmente?

a) PHP
b) ASP
c) JSP
d) JavaScript

¿El código PHP entre qué etiquetas se debe escribir?

a) Entre *<?* y *?>*
b) Entre *<?php* y *?>*
c) Entre *<%* y *%>*
d) Entre *<php>* y *</php>*

¿Cuáles de los siguientes son comentarios válidos en PHP?

a) `/*comentario */`
b) `//comentario`
c) `--comentario`
d) `#comentario`
e) `'comentario`

¿Qué nombre tiene el array de PHP que En PHP ¿Cómo se crearía una constante llamada TOTAL a la que le queremos asignar el valor 250?

a) `const $TAM=250;`
b) `const TAM=250;`
c) `define(TAM,250);`
d) `define($TAM,250);`

¿Cuáles de las siguientes proposiciones son ciertas?

a) Cualquier bucle se puede escribir con la instrucción **while**
b) Hay bucles que no se pueden escribir con la instrucción **while**
c) El cuerpo del bucle **while** al menos se ejecuta una vez
d) No siempre se ejecuta el cuerpo de un bucle **while**

7. ¿Se puede escribir código HTML dentro de PHP?

a) No, nunca
b) Sí, pero solo etiquetas HTML, no se puede incrustar en ellas código CSS
c) Sí y también código CSS
d) Se puede escribir código de cualquier lenguaje que se pueda incluir en una página web
Sí, pero solo código que iría dentro de la sección **body**.

¿Qué mostraría por pantalla este código?

```
echo "Hola a \todos";
```

Hola a \todos
Hola a odos
Hola a
odos
Hola a
todos

¿Qué atributo HTML de los controles de formulario es el que define el índice que tendrá ese parámetro dentro de los arrays $_GET o $_POST?

id
name
parameter
placeholder

10.- `define("TAM",250);` permite leer tanto parámetros enviados vía GET como vía POST?

a) $_FORM
b) $_RESPONSE
c) $_REQUEST
d) $_GET_POST

11.- El texto literal en PHP ¿Cómo se debe escribir?

a) Dentro comillas dobles
b) Dentro de comillas simples
c) Dentro de comillas dobles o simples
d) Dentro de llaves
e) Dentro de corchetes

12.- Dado este código:

```
$x=2;
$y=1;
$z=3;
$exp=($x>$y && $z>$y)||($x==$z);
```

¿Qué valor toma la variable $exp?

a) Verdadero (*true*)
b) Falso (*false*)
c) Ninguno porque la expresión tiene un error
d) 3

13.- ¿Qué muestra por pantalla este código?

```
$v=10;
while($v>0){
    echo $v--." ";
}
```

a) 10 9 8 7 6 5 4 3 2 1
b) 9 8 7 6 5 4 3 2 1 0
c) 9 8 7 6 5 4 3 2 1
d) Nada

14.- PHP es...

a) Un lenguaje de creación de aplicaciones web del lado del cliente
b) Un lenguaje de creación de aplicaciones web del lado del servidor
c) Un lenguaje de creación de aplicaciones web mixto
d) Un lenguaje de creación de aplicaciones web del lado de la base de datos

15.- Observa este código:

```
$x=9;
$y=&$x;
$y=7;
```

¿Cuánto vale $x al final de ese código?

a) 9
b) 7
c) indefinido
d) "y"

16.- ¿Cuál de estas afirmaciones es cierta?

a) El valor 1 significa verdadero y el valor 0 falso. El resto de números no son verdaderos ni falsos
b) Cualquier valor distinto de cero se toma como verdadero, el cero es el único valor falso
c) Los valores positivos son verdaderos y los negativos falsos. El cero es indefinido
d) En PHP los números no son ni verdaderos ni falsos.

17.- ¿Cuáles de las siguientes proposiciones son ciertas?

a) El método de paso de parámetros de formulario POST, no permite que nadie que esté pinchando la línea pueda obtener los datos que se están enviando
b) Es posible obtener los datos que se envían vía POST, ya que se encuentran dentro del paquete http
c) Es posible obtener los datos que se envían vía POST, ya que son visibles en la barra de direcciones del navegador
d) Es posible obtener los datos que se envían vía POST con herramientas especiales. Pero las contraseñas no, ya que viajan cifradas.

UNIDAD 4

NOCIONES AVANZADAS SOBRE EL LENGUAJE PHP

OBJETIVOS

- Reconocer la sintaxis de creación y uso de funciones personales
- Crear aplicaciones modulares utilizando funciones
- Asimilar las estructuras de datos que aporta el lenguaje PHP
- Crear aplicaciones web que requieran la manipulación de arrays
- Crear aplicaciones que requieran manipular y validar textos
- Reconocer las funciones de cifrado que aporta PHP para aumentar la seguridad de las aplicaciones
- Identificar la sintaxis y funciones de uso con expresiones regulares
- Reconocer los elementos de PHP que permitan el manejo de fechas

CONTENIDOS

4.1 FUNCIONES

4.1.1 INTRODUCCIÓN. PROGRAMACIÓN MODULAR

Cuando necesitamos escribir muchas líneas de código para crear una determinada aplicación, rápidamente se percibe el problema de tener que organizar el problema. Sin más elementos en el lenguaje, que los vistos en la unidad anterior, el problema se puede volver tan complejo que puede ser inabordable.

Para mitigar este problema apareció, ya hace bastantes años, la **programación modular.** En ella el código se divide en módulos de tamaño manejable. Cada módulo realiza una función muy concreta y así, el programador se concentra en esa tarea y evita la complejidad de tener que resolver el problema completo. Es decir, la aplicación se divide en módulos, cada uno de los cuales se programa de forma independiente.

En definitiva, se trata de concentrar los esfuerzos en resolver problemas sencillos y una vez resueltos, ensamblar todos los módulos de forma apropiada para resolver el problema completo. Esto no es más que modelar el famoso paradigma ***divide y vencerás*** en el mundo de la programación de aplicaciones.

En PHP la programación modular se implementa mediante funciones. Una función es simplemente código que recibe unos datos y, tras operar con ellos, devuelve un resultado. En realidad, no siempre se devuelve un resultado, a veces la función simplemente realiza una determinada tarea. A ese último tipo de funciones otros lenguajes las llaman **procedimientos**.

A los datos que necesita la función para realizar su tarea se los llama **parámetros**. Una función no siempre requiere parámetros.

Cuando una función ha sido totalmente probada y validada, se puede utilizar las veces que haga falta sin necesidad de tener que volver a programarla; incluso, si ha sido bien programada, se puede utilizar en otras aplicaciones que escribamos. Así un conjunto de funciones útiles que ha creado un programador, y que se utilizan en diferentes aplicaciones, forman lo que se conoce como **librería**; un archivo que solo contiene funciones.

En PHP las funciones tienen esta forma de trabajar:

[1] Las funciones poseen un nombre o **identificador** que cumple las reglas indicadas para los demás identificadores que conocemos (como los de las variables). Pero, a diferencia de las variables, no utilizan el signo **$**. Se aconseja que el nombre de las funciones se escriba en minúsculas. El identificador de una función siempre va seguido de paréntesis.

[2] Al definir las funciones, entre paréntesis se indican los parámetros que necesita la función para hacer su labor.

[3] Tras la declaración de los parámetros se escribe el código de la función, en el que se implementa lo que la función debe ser capaz de realizar.

[4] Las funciones pueden devolver un valor, resultado del trabajo de la misma, mediante el comando **return**.

PHP incorpora muchas funciones ya creadas para trabajar (como la propia **print** o la muy utilizada **echo**). Por ejemplo, este código:

```
echo mt_rand(1,10);
```

Escribe un número aleatorio entre uno y diez. La función **mt_rand** (mt_rand sería su identificador) recibe como parámetros dos números y con ellos consigue devolver un número aleatorio entre ambos. Al ser una función perteneciente al propio núcleo de PHP, no podemos ver su código, pero si utilizarla como si fuera una función que hubiéramos creado nosotros.

4.1.2 DECLARACIÓN Y USO DE FUNCIONES PERSONALES

Para crear nuestras propias funciones hay que seguir esta sintaxis:

```
function nombreDeLaFunción(listaDeParámetros){
   código de la función
}
```

Entre el código de la función se puede encontrar la palabra **return** que, como se ha comentado anteriormente, sirve para devolver un resultado. Ejemplo de función:

```
function doble($valor){
   return 2*$valor;
}
```

Las funciones se pueden declarar en cualquier parte de una página PHP, pero lo aconsejable es declarar las funciones en la cabecera de la página web, o incluso, por encima del inicio del código HTML, para facilitar su mantenimiento.

Una vez definida una función, para utilizarla, simplemente hay que invocarla pasando los parámetros que requiere, por ejemplo:

```
$x=9.75;
echo doble(8)."<br />"; //escribe 16
echo doble($x);         //escribe 19.5
```

Como se ha dicho, no siempre las funciones devuelven valores, por ejemplo:

```
function negrita($texto){
    echo "<strong>".$texto."</strong>";
}
```

La función no tiene instrucción **return**, puesto que no devuelve valores, sino que lo que hace es escribir el texto en negrita.

Ejemplo de uso de la función anterior:

```
negrita("Hola"); //escribe <strong>Hola</strong>
```

4.1.3 ALCANCE DE LAS VARIABLES

Las variables definidas en una función, al finalizar la función se eliminan. Es decir, su ámbito es local a la función. Ejemplo:

```
function f1(){
   $h=9;
}
echo $h; //error: Variable no definida
```

La instrucción ***echo*** del ejemplo anterior, provoca un fallo de variable no definida, porque PHP no reconoce a la variable ***$h***, la única ***$h*** del código se crea en la función y solo se puede usar en ella; hacer referencia a ***$h*** fuera dela función no tiene sentido, ya que tras el cierre de la llave en la que se definió, la variable muere. La variable es, en definitiva, local a la función. Otro ejemplo:

```
$h=5;
function f1(){
   $h=9;
}
echo $h; //escribe 5
```

En este código, se clarifica que una variable no existe fuera de la función. Para PHP, en el ejemplo anterior, hay dos variables llamadas ***$h***: una es global y la otra es local a la función ***f1***. Cuando se cierra la función, la variable ***$h*** que se ha definido dentro de la función desaparece, y por ello, en la última línea el valor que se escribe es 5, correspondiente a la variable ***$h*** global.

4.1.4 PASO DE PARÁMETROS POR REFERENCIA

Por defecto las funciones reciben los parámetros por valor, esto significa que los parámetros de la función usan copias de los valores:

```
function f1($x){
  $x=9;
}
$x=7;
f1($x);
echo $x; //escribe 7
```

Este es un ejemplo de código confuso. La función f1 se ha definido de modo que requiere un parámetro al que ha llamado ***$x*** y que dentro de la función toma el valor 9. Por otro lado, fuera de la función se ha definido una variable que también se llama **$x**, a la que se asigna el valor 7. Cuando se invoca a la función con el código `f1($x)`, lo que hace la función es tomar una copia

de ese valor. El cambio que se realiza dentro de la función no afecta a la variable exterior a ella. Por ello, lo que escribe es 7.

A veces, sí se desea que las funciones cambien el valor de las variables que se envían como parámetro. El ejemplo más claro es el de la clásica función ***swap***. La función swap sirve para intercambiar los valores de dos variables.

Se desea que está instrucción: `swap($x,$y)`, sirva para intercambiar los valores de *x* e *y*. Para ello, se programa de esta forma:

```
function swap($x,$y){
    $aux=$x;
    $x=$y;
    $y=$aux;
}
$valor1=12;
$valor2=17;
swap($valor1,$valor2);
echo $valor1." ".$valor2;
```

En apariencia swap funciona, pero realmente no modifica los valores de las variables ***$valor1*** y ***$valor2***, puesto que los parámetros (***$x*** y ***$y***) reciben una copia de los valores, tras invocar a ***swap***, ***$x*** e ***$y*** siguen valiendo lo mismo. La función ***echo*** escribirá *12 17*.

Si deseamos modificar los valores originales, basta indicar el signo *&* antes del nombre del parámetro que deseamos modificar (véase Apartado 3.4.6 *"Referencias &"*, en la página 131). Así:

```
function swap(&$x,&$y){
    $aux=$x;
    $x=$y;
    $y=$aux;
}
$valor1=12;
$valor2=17;
swap($valor1,$valor2);
echo $valor1." ".$valor2;
```

Escribe *17 12* ya que ahora sí que las variables originales han sido modificadas.

Cuando una función utiliza paso por referencia, entonces al invocarla debe recibir nombres de variables y no expresiones o valores literales. Es decir, este código:

```
swap(9,8); //error: Se requiere una variable
```

Produce un error, porque la función ***swap*** espera variables y no expresiones.

4.1.5 PARÁMETROS PREDEFINIDOS

Se puede indicar un valor por defecto a los parámetros a fin de permitir que dicho parámetro sea opcional; de modo que, si se pasa el parámetro se toma el valor que se pasa y si no, se toma el valor por defecto. Ejemplo:

```
function potencia($base,$exponente=2){
    $valor=1;
    for($i=1;$i<=$exponente;$i++){
        $valor*=$base;
    }
    return $valor;
}
echo potencia(2,3); //Escribe 8, es decir 2^3
echo potencia(4);    //Escribe 16, es decir 4^2
```

La segunda vez que se invoca a la función, el parámetro ***$exponente*** toma el valor por defecto **2**.

4.1.6 VARIABLES GLOBALES

Las variables globales permiten ser utilizadas en todo el código, sin que las funciones utilicen variables que se han definido fuera de la función. Ejemplo:

```
function encadenar($número,$carácterRelleno){

    global $texto;
    for($i=1;$i<=$número;$i++){
        $texto.=$carácterRelleno;
    }
}
$texto="Hola";
encadenar(12,"d");
echo $texto."<br />"; //Escribe: Holaddddddddddd
encadenar(9,"+-");
echo $texto."<br />"; //Escribe: Holaddddddddddd+-+-+-+-+-+-+-+-+-
```

La función ***encadenar*** utiliza la variable global ***$texto*** definida fuera de la función, gracias al uso de la palabra **global**.

Esta técnica, en principio, no es recomendable ya que la función ya no es independiente del código que se ha escrito fuera de ella. Las funciones deberían ser lo más independientes posibles. No es lógico que debamos conocer el cuerpo de la función (el código interior a ella) para poder utilizarla o que, para utilizar la función, se nos obligue a definir variables globales.

En resumen, es una posibilidad que ofrece PHP, pero que es poco recomendable porque con ella se adquieren malas mañas al programar. Sin embargo hay librerías que requieren el uso de variables globales, y esa es una razón por la que conviene conocer esta técnica.

4.1.7 VARIABLES ESTÁTICAS

Se trata de variables locales a la función (solo se podrán utilizar dentro de la función), pero que son capaces de recordar su valor entre cada llamada a la función. Ejemplo:

```
function estatica(){
    static $cuenta=0;
    $cuenta++;
    echo "Esta es la llamada número $cuenta<br />";
}

for($i=1;$i<=10;$i++){
    estatica();
}
```

La salida del código anterior es:

```
Esta es la llamada número 1
Esta es la llamada número 2
Esta es la llamada número 3
Esta es la llamada número 4
Esta es la llamada número 5
Esta es la llamada número 6
Esta es la llamada número 7
Esta es la llamada número 8
Esta es la llamada número 9
Esta es la llamada número 10
```

El código de la función escribe el valor de la variable ***$cuenta***. Esa variable debería eliminarse al abandonar la función, pero sobrevive y se puede seguir utilizando. Eso solo es posible gracias a la línea en la que se define la variable como estática (`static $cuenta=0`) que solo se ejecuta en la primera llamada. Si en esa línea eliminamos la palabra **static**, entonces la variable se crea en cada llamada y las diez líneas anteriores, escribirían lo mismo (***Esta es la llamada número 1***)

4.1.8 RECURSIVIDAD

La recursividad es una técnica de creación de funciones, pensada para solventar problemas complejos. Se basa en que, dentro del código de la función, ésta se invoca a sí misma.

Es una técnica peligrosa, ya que se pueden generar fácilmente llamadas infinitas: la función se llama a sí misma, tras la llamada se vuelve a llamar a sí misma, y así sucesivamente en un bucle sin fin. Hay que ser muy cauteloso con esta técnica y, en todo momento, pensar en la condición que debe poner fin a las llamadas.

Como se ha comentado, la idea es solucionar problemas complejos de forma sencilla. El planteamiento general es que en cada llamada a la función, ésta resuelva parte del problema y se invoque a sí misma de nuevo para resolver la parte que queda, y así sucesivamente. En cada lla-

mada el problema debe ser cada vez más sencillo, hasta llegar a una llamada en la que la función devuelve un único valor. Tras esa llamada los resultados se encadenan hasta conseguir devolver el resultado final.

Es fundamental tener muy clara la condición que hace que la función debe deje de llamarse a sí misma; lo que implica calibrar muy bien cuándo acabar. Un bucle infinito, utilizando recursividad, puede causar graves inestabilidades en el ordenador que interpreta el código. En este caso además, el ordenador que ejecuta el código es el servidor web, y un error de este tipo puede provocar que no se pueda acceder a ninguna de las páginas que aloja.

El ejemplo clásico de función recursiva es la que resuelve el problema del factorial. El factorial es una operación matemática que devuelve la multiplicación de todos los números desde el número 1 hasta llegar al número del que se calcula el factorial. Así el factorial de 5 es 1·2·3·4·5, es decir 120. La función PHP de cálculo de factorial se puede escribir así:

```
function factorial($n){
   if($n<=1) return 1; //condición de salida
   else return $n*factorial($n-1);
}
```

La última instrucción (*return $n*factorial($n-1)*) es la que realmente aplica la recursividad. La idea (por otro lado más humana) es considerar que el factorial de nueve es, nueve multiplicado por el factorial de ocho; a su vez el de ocho es ocho por el factorial de siete y así sucesivamente hasta llegar al factorial de uno, que devuelve uno.

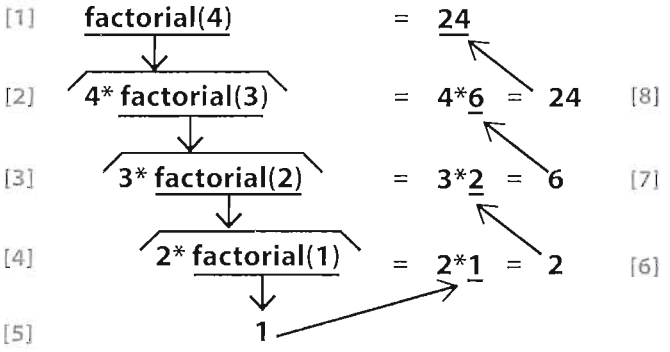


Figura 4.7: Ciclo de llamadas a la función recursiva factorial

Para la instrucción *factorial(4);* usando el ejemplo anterior, la ejecución del programa generaría los siguientes pasos:

[1] Se llama a la función factorial usando como parámetro el número *4* que será copiado en la variable-parámetro *$n*

[2] Como *$n>1*, entonces, el resultado será el número *4* multiplicado por el resultado de la llamada *factorial(3)*

[3] La llamada anterior hace que el nuevo ***$n*** (variable distinta de la anterior) tome el valor *3*, por lo que esta llamada devolverá *3* multiplicado por el resultado de la llamada ***factorial(2)***

[4] La llamada anterior devuelve *2* multiplicado por el resultado de la llamada ***factorial(1)***

[5] Por primera vez el resultado de la llamada recursiva devuelve un resultado directo, el número *1*

[6] Gracias a ese resultado, la llamada ***factorial(2)*** da como resultado *2*1*, es decir *2*

[7] Eso hace que la llamada ***factorial(3)*** devuelva *3*2*, es decir *6*

[8] Por lo que la llamada ***factorial(4)*** devuelve *4*6*, es decir *24* Y ese es ya el resultado final

4.1.8.1 ¿RECURSIVIDAD O ITERACIÓN?

Hay otra versión de la función factorial resuelta mediante un bucle **for** (solución iterativa), en lugar de utilizar la recursividad. En concreto es:

```
function factorial2($n){
    $res=1;
    for($i=1;$i<=$n;$i++){
        $res*=$i;
    }
    return $res;
}
```

La cuestión es ¿cuál es mejor? ¿La solución recursiva o la iterativa? Ambas utilizan sentencias repetitivas hasta llegar a una determinada condición. Por lo que ambas pueden provocar bucles infinitos si se programan mal. En el caso de la iteración es un contador el que permite determinar el final. Lo que hace la recursividad es ir simplificando el problema hasta generar una llamada a la función que devuelva un único valor.

Pero por la manera de implementar funciones en los ordenadores, es más costosa la recursividad, ya que requiere realizar muchas llamadas a funciones, y cada llamada genera una copia del código de la función. Esto sobrecarga la zona de memoria de las computadoras conocida como montículo o ***heap***. Es decir, **es más rápida y menos costosa, en términos de memoria, la solución iterativa de un problema recursivo.**

¿Por qué utilizar entonces la recursividad? Lo cierto es que si podemos resolver un problema mediante soluciones iterativas, no deberíamos utilizar recursividad. La recursividad se utiliza solo si:

- No encontramos la solución iterativa a un problema
- El código es muchísimo más claro en su versión recursiva y no implica un gran coste de procesamiento al ordenador

4.1.9 ÁMBITO DE LAS FUNCIONES

Las funciones en PHP siempre son globales. Es decir, una función se puede utilizar en cualquier parte del código PHP. Por supuesto, esto prohíbe poner el mismo nombre a dos funciones.

4.2 INCLUSIÓN DE FICHEROS

Dentro del código PHP se puede hacer uso de las instrucciones **include** y/o **require** para incluir el código de otro archivo. El archivo puede ser de cualquier tipo: **html**, **php** u otras extensiones. Esto permite la creación de archivos que contengan librerías de funciones o código reutilizable en múltiples páginas.

Tanto **include** como **require** lo que hacen es simplemente copiar y pegar el código del archivo tal cual. La diferencia es que si el archivo no existe (o no se encuentra), **include** seguirá ejecutando el código (aunque normalmente mostrará una advertencia del error), mientras que **require** generará un error grave y parará la ejecución del código.

En la inclusión de archivos PHP, se entiende que el código que se incluye es un archivo normal, en el que el código PHP se encuentra entre las etiquetas <?**php** y ?>, mientras que el código fuera de esas etiquetas se entiende que es HTML.

Ejemplo de uso:

```
include("funciones.php");
```

4.3 ARRAYS

4.3.1 INTRODUCCIÓN A LOS ARRAYS

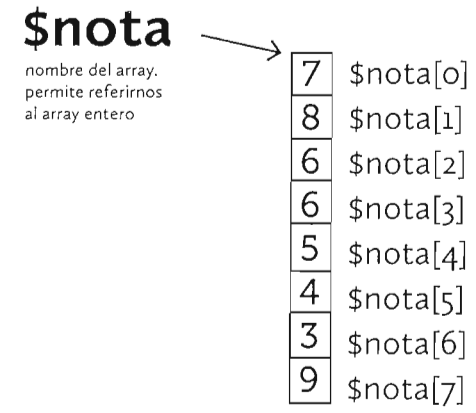


Figura 4.8: Ejemplo de array para almacenar notas académicas

Los tipos de datos que conocemos hasta ahora no permiten manejar grandes cantidades de datos a la vez. Por ejemplo, imaginemos que deseamos almacenar las notas de una clase de 25 alumnos, no habrá más remedio, con lo que sabemos hasta ahora, que declarar 25 variables.

Eso es tremendamente pesado de programar. Utilizar esos datos significaría estar continuamente manejando las 25 variables. Por supuesto, si necesitamos almacenar 2000 notas, el problema se hace inmanejable.

Por ello, en casi todos los lenguajes, se pueden agrupar una serie de variables del mismo tipo en una misma estructura que comúnmente se conoce como

array[1]. Esa estructura permite referirnos a todos los elementos, pero también nos permite acceder individualmente a cada elemento.

Los arrays son una colección de datos a la que se asigna un identificador (por ejemplo ***nota***). Es una variable, solo que tiene capacidad para almacenar una serie de valores. Para acceder a un dato individual de la colección hay que utilizar su posición dentro del array. La posición es un número entero que se llama **índice**. Así, para acceder a la quinta nota, se utilizaría la expresión ***$nota[4]***.

Hay que tener en cuenta que en los arrays el primer elemento tiene como índice el número cero. El segundo el uno y así sucesivamente; es decir, ***nota[4]*** en realidad es el quinto elemento del array.

Esta definición de arrays es la común en todos los lenguajes clásicos. En el caso de PHP, los arrays son elementos más complejos, puesto que admiten datos de distinto tipo. La realidad es que los arrays de PHP son, en realidad, elementos que asocian un valor y una clave (**arrays asociativos**) como se explica más adelante.

4.3.2 ARRAYS ESCALARES

Los arrays escalares cumplen a rajatabla la definición indicada anteriormente: son un conjunto de valores a los que se accede a través de un índice, que es un número entero en el que el primer elemento tiene índice cero. Así, por ejemplo, las notas se podrían almacenar de esta forma:

```
$nota[0]=7;
$nota[1]=8;
$nota[2]=6;
$nota[3]=6;
$nota[4]=5;
$nota[5]=4;
...
```

Acceder a una nota cualquiera, requiere conocer su índice. Se observa en el código anterior que no hace falta declarar el array (siguiendo el estilo habitual de PHP), simplemente se utiliza. Es válido incluso este código:

```
$valor[0] =18;
$valor[1]="Hola";
$valor[2]=true;
$valor[3]=3.4;
```

Es decir, los datos pueden ser heterogéneos, de distinto tipo. También podemos asignar valores a un array sin indicar índice:

1 Otros nombres que se utilizan habitualmente para referirse a los arrays son: listas, matrices, arreglos,... Parece claro que **array** es el más aceptado en la actualidad, los otros términos son más ambiguos.

```
$valor[] =18;
$valor[]="Hola";
$valor []=true;
$valor[]=3.4;
```

En este caso, los índices se van asignando automáticamente (empezando por el cero).

4.3.2.1 BUCLE FOR DE RECORRIDO DE ARRAYS

La gracia de los arrays en todos los lenguajes es la facilidad de rellenar o utilizar su contenido mediante bucles **for**. Imaginemos que necesitamos almacenar 1000 números aleatorios del 1 al 10; evidentemente sería terrible tener que escribir 1000 líneas de código para realizar esta labor:

```
$valor[0]=mt_rand(1,10);
$valor[1]=mt_rand(1,10);
$valor[2]=mt_rand(1,10);
...//¡¡¡mucho trabajo!!!!
```

Lo lógico es escribir este código:

```
for($i=0;$i<1000;$i++)
    $valor[$i]=rand(1,10);
```

La variable **$i** sirve para recorrer cada elemento del array. Va cambiando de 0 a 999 y así en dos líneas de código, se rellena de forma cómoda el array.

4.3.2.2 ASIGNACIÓN DE VALORES CON LA FUNCIÓN ARRAY()

Los arrays pueden definirse de forma cómoda mediante la función **array()**. Dentro de esa función se colocan los valores que tendrá el array; y así en una sola línea podemos asignar todos los valores del array. Ejemplo:

```
$numero=array(17, 34, 45, 2, 9, -5, 7);
```

Esa línea asigna valores a cada elemento del array empezando por el cero. Es decir, es equivalente a:

```
$numero[0]=17;
$numero[1]=34;
$numero[2]=45;
$numero[3]=2;
$numero[4]=9;
$numero[5]=-5;
$numero[6]=7;
```

Esta función tiene posibilidad de elegir índices concretos. Eso se hace colocando el índice y después el símbolo => antes del valor.

Ejemplo:

```
$numero=array(17,  34, 3=>45, 2, 7=>9, 10=>-5, 7);
```

Sería equivalente a:

```
$numero[0]=17;
$numero[1]=34;
$numero[3]=45;

$numero[4]=2;
$numero[7]=9;
$numero[10]=-5;
$numero[11]=7;
```

Los índices que no se indican quedarán sin definir

4.3.2.3 FUNCIÓN COUNT

La función **count** que posee PHP permite saber el tamaño de un array. Es útil para usar un array del que previamente no sabemos su tamaño.

Por ejemplo, si deseamos recorrer un array que ya tiene valores para, por ejemplo, mostrar cada elemento en pantalla, el código sería:

```
for($i=0;$i<count($valor);$i++){
    echo $valor[$i]."<br />";
}
```

4.3.2.4 ELEMENTOS VACÍOS

En este código:

```
$nota[0]=5;
$nota[1]=9;
$nota[3]=5;
$nota[4]=6;
$nota[5]=7;
```

Deliberadamente se ha dejado sin rellenar el elemento ***nota[2]***. De modo que la cuestión es ¿qué sacaría por pantalla la instrucción `echo $nota[2]`? Normalmente un servidor Apache en el que se ejecute este código mostraría un mensaje parecido a este:

```
Notice: Undefined offset: 2 in E:\xampp\htdocs\arrays\arrayPrb1.php
on line 18
```

Undefined offset se puede traducir como ***desplazamiento indefinido*** y además indica que dicho desplazamiento ocurre por el número **2**. Es decir, lo que Apache indica es que al moverse el ele-

mento *2* del array se encuentra con que es una posición indefinida. Dicho de otra forma: que es un índice de array que no existe.

La función **count** devuelve el tamaño del array sin tener en cuenta los elementos vacíos; es decir, solo cuenta los elementos definidos. Por todo ello el código anterior:

```
for($i=0;$i<count($valor);$i++){
    echo $valor[$i]."<br />";
}
```

Puede fallar si el array ***$valor*** tiene elementos vacíos. Además los últimos elementos no saldrían porque el contador no llega a ellos al contar elementos y no los índices más altos.

Para evitar eso, podemos utilizar la función **isset** que devuelve verdadero cuando a la variable que se le pasa como parámetro se le ha asignado ya un valor. El bucle anterior quedaría entonces:

```
$tope=count($nota);
for($i=0;$i<$tope;$i++){
    if(isset($nota[$i]))
     echo $nota[$i]."<br />";
    else
     $tope++;
}
```

El código es más enrevesado. Ahora solo se escribe cada elemento del array si la función **isset** nos asegura que dicho elemento tiene valor; la variable ***$tope*** va incrementando su valor a fin de alcanzar al último elemento real del array.

4.3.3 ARRAYS ASOCIATIVOS

En este caso el índice es un texto que se comporta como una clave que permite acceder al dato. Los arrays así construidos forman estructuras de datos de tipo clave/valor, como ocurre con los arrays **$_GET** y **$_POST**. Ejemplo:

```
$nota["Antonio"]=5;
$nota["Luis"]=9;
$nota["Ana"]=8;
$nota["Eloy"]=5;
$nota["Gabriela"]=6;
$nota["Berta"]=7;
```

La ventaja de estos arrays es que son más legibles, la desventaja es que no permiten su manejo mediante los bucles clásicos. Al simular datos de tipo ***clave/valor*** se relacionan muy bien con las bases de datos. Estos arrays también permiten el uso de la función **array()**:

```
$nota=array("Antonio"=>5,"Luis"=>9,"Ana"=>8,"Eloy"=>5,
   "Gabriela"=>6,"Berta"=>7);
```

La función **array()** incluso permite mezclar valores escalares y asociativos (aunque no es muy recomendable), por ejemplo:

```
$nota=array("Antonio"=>5,  4,  7, 5=>6, "Luis"=>9, 2,"Berta"=>7);
```

El ejemplo anterior sería equivalente a:

```
$nota["Antonio"]=5;
$nota[0]=4;
$nota[1]=7;
$nota[5]=6;
$nota["Luis"]=9;
$nota[6]=2;
$nota["Berta"]=7;
```

4.3.4 BUCLE FOREACH

Recorrer arrays asociativos o incluso escalares con elementos si inicializar, resulta muy complejo (y pesado) con los bucles **while** o **for**. Por ello PHP dispone de un bucle muy potente que permite recorrer todos los elementos de un array sin importar si hay elementos indefinidos, ya que es capaz de ignorarles; lo que le hace idóneo para recorrer arrays asociativos.

Se trata del bucle **foreach** que tiene esta sintaxis:

```
foreach(array as índice=>valor ){
    sentencias
}
```

array es el nombre del array que se va a recorrer; ***índice*** es la variable que recogerá el **índice** de cada elemento a medida que se recorra el array; y ***valor*** es la variable que irá recogiendo el valor de cada elemento del array. Ejemplo:

```
$nota=array("Antonio"=>5,  4,  7, 5=>6, "Luis"=>9, 2,"Berta"=>7);
foreach ($nota as $i=>$v){
  echo "La nota de $i es $v\n";
}
/* Escribe:
   La nota de Antonio es 5
   La nota de 0 es 4
   La nota de 5 es 6
   La nota de Luis es 9
   La nota de 6 es 2
   La nota de Berta es 7
*/
```

4.3.5 ARRAYS MULTIDIMENSIONALES

Un array multidimensional sirve para representar datos agrupados en una estructura que utiliza dos índices para representar cada elemento del array. Ejemplos:

```
$nota[0][0]=9;
$nota[0][1]=7;
$nota[1][0]=9;
...
```

Por supuesto es posible que el array sea asociativo:

```
$poblacion["Alemania"]["Berlin"]=4000000;
$poblacion["Alemania"]["Hanover"]=1200000;
$poblacion["Francia"]["Paris"]=7400000;
$poblacion["Francia"]["Lyon"]=2300000;
$poblacion["España"]["Palencia"]=80000;
```

Se pueden utilizar incluso más de dos dimensiones:

```
$poblacion["España"]["Castilla y León"]["Palencia"]=80000;
$poblacion["España"]["Castilla y León"]["Valladolid"]=370000;
$poblacion["España"]["Asturias"]["Oviedo"]=115000;
$poblacion["Alemania"]["Brandemburgo"]["Berlin"]=4000000;
```

De esa forma podemos considerar a los arrays como una especie de almacenes de datos que se parecen mucho a las tablas de las bases de datos relacionales; ese detalle facilita mucho la comunicación entre PHP y las bases de datos. Otra forma posible es:

```
$alemania=array("Berlin", "Hannover");
$francia=array("París", "Lyon");
$espania=array("Palencia");
$poblacion=array("alemania"=>$alemania,
        "francia"=>$francia,"espania"=>$espania);
```

4.3.5.1 CONSTRUCTOR **ARRAY()** EN ARRAYS MULTIDIMENSIONALES

La función **array()** está pensada para arrays de una dimensión. Pero es posible utilizarla en arrays multidimensionales considerando que un array de dos dimensiones es un array de arrays. De esa forma:

```
$poblacion=array(
  "España"=>array(
          "Palencia"=>80000,
          "Valladolid"=>350000,
          "Oviedo"=>120000),
```

```
    "Francia"=>array(
            "Paris"=>7000000,
            "Lyon"=>2100000)
);
```

El código anterior no tiene porque ocupar tantas líneas, pero es más legible así.

Ese código es equivalente a:

```
$poblacion["España"]["Palencia"]=80000;
$poblacion["España"]["Valladolid"]=350000;
$poblacion["España"]["Oviedo"]=120000;
$poblacion["Francia"]["Paris"]=7000000;
$poblacion["Francia"]["Lyon"]=2100000;
```

4.3.5.2 RECORRIDO MEDIANTE FOREACH DE ARRAYS MULTIDIMENSIONALES

Nuevamente se basa la idea en considerar que un array multidimensional es un array de arrays. Así en el que caso de que con el array anterior hiciéramos este recorrido:

```
$poblacion=array(
  "España"=>array(
          "Palencia"=>80000,
          "Valladolid"=>350000,
          "Oviedo"=>120000
  ),
    "Francia"=>array(
          "Paris"=>7000000,
          "Lyon"=>2100000
    )
  );
foreach($poblacion as $i=$valor){
  echo "El índice $i vale $valor<br />";
}
```

El resultado es:

```
El índice España vale Array
El índice Francia vale Array
```

Lo que nos indica el resultado es que el valor de cada elemento es un array, con lo cual el código que realmente nos permite examinar cada valor sería:

```
$poblacion=array(
   "España"=>array(
            "Palencia"=>80000,
            "Valladolid"=>350000,
            "Oviedo"=>120000
   ),
     "Francia"=>array(
              "Paris"=>7000000,
              "Lyon"=>2100000
     )
   );
foreach ($poblacion as $pais => $ciudades) {
    foreach ($ciudades as $ciudad => $valor) {
        echo "<p>País: $pais, Ciudad: $ciudad,".
         " población = $valor</p>";
    }

  }
```

Que obtiene el resultado:

```
País: España, Ciudad: Palencia, población = 80000
País: España, Ciudad: Valladolid, población = 350000
País: España, Ciudad: Oviedo, población = 120000
País: Francia, Ciudad: Paris, población = 7000000
País: Francia, Ciudad: Lyon, población = 2100000
```

4.3.6 INSPECCIÓN DE ARRAYS MEDIANTE FUNCIONES DE RECORRIDO

Se trata de funciones basadas en el manejo de punteros de los lenguajes tradicionales. Utilizan un puntero virtual, que sería un objeto que señala a uno de los elementos del array y que al moverle permite acceder al resto de elementos.

Esta técnica se basa en el manejo de las siguientes funciones:

FUNCIÓN	USO
current(array)	Devuelve el valor del elemento al que actualmente señala el puntero. Si no hay ningún elemento, devuelve **false**
key(array)	Devuelve la clave a la que señala el puntero
next(array)	Mueve el puntero al siguiente elemento del array y devuelve su valor. Si el elemento actual es el último, **next** devuelve **false**

FUNCIÓN	USO
prev(array)	Mueve el puntero al elemento anterior del array y devuelve su valor. Si el elemento actual es el primero, **prev** devuelve false
reset(array)	Coloca el puntero en el primer elemento del array y devuelve su valor. Si no hay ningún elemento en el array, devuelve **false**
end(array)	Coloca el puntero en el último elemento y devuelve su valor

Ejemplo de recorrido de un array mediante estas funciones:

```
$capital=array(
   "Castilla y León"=>"Valladolid",
   "Asturias"=>"Oviedo",
   "Aragón"=>"Zaragoza"
);
while(current($capital)){
      echo "<strong>".current($capital)."</strong><br />";
      next($capital);
}
```

Para recorrer el array al revés:

```
$capital=array(
   "Castilla y León"=>"Valladolid",
   "Asturias"=>"Oviedo",
   "Aragón"=>"Zaragoza"
);
end($capital)
while(current($capital)){
      echo "<strong>".current($capital)."</strong><br />";
      prev($capital);
}
```

4.3.7 FUNCIONES Y ARRAYS

Las funciones pueden recibir y devolver arrays al igual que cualquier otro tipo de variables. A diferencia de la mayoría de lenguajes estructurados, en PHP los arrays se pasan por valor a las funciones. Es decir, cuando se pasa como parámetro un array a una función, el parámetro genera una copia del mismo. Así, las modificaciones al array que se hagan dentro de la función, no afectan al array original. Ejemplo:

```
function prueba($a){
   $a[0]=18;
}
```

```
$array=array(1,2,3,4,5,6,7);
prueba($array);
echo $array[0]; //Escribe el valor: 1
```

La función ***prueba*** modifica el elemento con índice cero en el array para darle el valor *18*. Sin embargo, al ejecutar el código PHP anterior, veremos el número *1* en pantalla. La modificación de la función prueba se ha hecho con el array ***$a*** que es una copia en realidad del original ***$array***.

Si deseáramos que realmente la función modifique el array, necesitamos pasar el array por referencia, como ya se ha visto en PHP basta con indicar el símbolo & delante del parámetro (o parámetros) que deseamos pasar por referencia:

```
function prueba(&$a){
   $a[0]=18;
}
$a=array(1,2,3,4,5,6,7);
prueba($a);
echo $a[0];
```

Ahora sí escribe *18*.

4.3.8 USO DE ARRAYS EN FORMULARIOS

Una de las virtudes de un array, es su capacidad de recoger en una sola variable, valores procedentes de diferentes controles en un formulario. La manera de hacerlo es muy sencilla, basta con indicar un nombre de array como atributo **name** en los elementos del formulario; para indicar que ese nombre es de un array, se añaden al nombre la apertura y cierre de corchetes [y]. Ejemplo:

```
<!DOCTYPE html>
<html lang="es-ES">
<head>
        <meta charset="UTF-8">
        <title>Array en formularios</title>
</head>
<body>
   <form action="formArrayget.php" method="GET">
     Elige estas opciones:<br />
     <input type="checkbox" name="opciones[]" value="menor"
         id="menor" />
     <label for="menor"> Menor de edad </label><br />
     <label for="minus"> Minusvalía</label><br />
```

```
        <input type="checkbox" name="opciones[]" value="numerosa"
           id="numerosa" />
        <label for="numerosa">Familia numerosa</label><br />
        <input type="checkbox" name="opciones[]" value="minima"
           id="minima" />
        <label for="minima"> Renta mínima</label><br />
        <input type="submit" values="Enviar" />
    </form>
</body>
</html>
```

La página resultante es (suponiendo que hemos marcado después algunas opciones):

Elige estas opciones:
Menor de edad
Minusvalía
Familia numerosa
Renta mínima
Enviar

En esta página, todos los checkbox están asociados a un array llamado ***opciones***. Dicho array contendrá un valor por cada elemento al que le hayamos hecho clic. Los índices se indican de forma escalar; es decir, el primer valor (en el orden de escritura de la página web) será el cero, luego el uno...

De modo que si el código de la página que recoge el array (*formArrayget.php*) es:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="UTF-8">
      <title>Impresión de opciones</title>
  </head>
  <body>
        <?php
            print_r($_GET["opciones"]);
        ?>
  </body>
</html>
```

La salida de dicha página tras recoger los valores del formulario anteriormente comentados es:

```
Array ( [0] => menor [1] => numerosa )
```

Indicando que el array ***opciones*** (presente en el array de recogida del formulario **$_GET**) tiene dos elementos (con índices 0 y 1) con las valores ***menor*** y ***numerosa*** (correspondientes a los valores de los controles checkbox del formulario).

Es posible utilizar índices concretos en el array al recoger valores.

Ejemplo:

```
<input type="checkbox"  name="compra['periodico']"
                  value="1.2" id="periodico"/>
<label for="periodico">Comprar periódico </label><br />
<input type="checkbox"  name="compra['revista']"
                  value="2.5" id="revista" />
<label for="revista">Comprar revista</label><br />
<input type="checkbox"  name="compra['libro']"
                  value="15" id="libro" />
<label for="libro">Comprar libro </libro><br />
```

De esa forma, tanto el índice del array, como el valor nos proporciona información. Si esos checkbox les ponemos en un formulario y pulsamos todos, se recibirá un array llamado compra con los índices ***periodico, revista y libro*** y los valores *1.2*, *2.5* y *15* respectivamente.

4.3.9 ANEXO: FUNCIONES DE USO CON ARRAYS

4.3.9.1 FUNCIONES BÁSICAS

FUNCIÓN	SIGNIFICADO
count(array)	Devuelve el número de elementos del array
print_r(array)	Escribe el contenido del array (tanto valores como índices)
is_array(array)	Devuelve verdadero si el parámetro que recibe es un array

4.3.9.2 ORDENACIÓN DE ARRAYS

FUNCIÓN	SIGNIFICADO
sort(array, [flags])	Ordena el array indicado. Si no se indica el segundo parámetro, ordena de forma normal sin convertir los tipos de datos El segundo parámetro permite establecer la forma de ordenación pudiendo indicar estas posibilidades: ■ **SORT_REGULAR.** Ordenación normal (utilizada por defecto), se adapta según el valor sea numérico o String. Solo funciona bien caracteres pertenecientes al código ASCII original, por lo que no funciona con símbolos como la eñe o las vocales con tilde ■ **SORT_NUMERIC**. Ordena entendiendo que el array contiene valores numéricos ■ **SORT_STRING**. Ordena entendiendo que el array contiene valores String (texto). Ordena según el código ASCII

FUNCIÓN	SIGNIFICADO
sort *(continuación)*	■ SORT_LOCALE_STRING. Ordena basándose en la configuración regional establecida con la función **setlocale**. De modo que `setlocale(LC_ALL,"es_ES")` establecería como configuración regional, español de España (en Windows se usa el código `esp_ESP`) En cualquier caso no sirve para arreglar el problema de la ordenación de caracteres fuera del ASCII ■ SORT_NATURAL. Ordena el array considerando sus valores como lo haría un ser humano (es decir el texto *"texto2"* iría delante de *"texto10"* porque entendería el diez como un número superior al dos, y no ordenaría como si fueran textos alfabéticos. El funcionamiento es el mismo que la función **natsort** ■ SORT_FLAG_CASE Permite combinar (usando el operador **OR** (|) a nivel de bits) con **SORT_STRING** o **SORT_NATURAL** para ordenar cadenas de forma insensible a mayúsculas/minúsculas. Por ejemplo: `sort($array,SORT_STRING|SORT_NATURAL)` Al ordenar mediante la función **sort**, los índices desaparecen, el array final es un array escalar (el primer índice será el cero, luego el uno, etc.)
usort(array, funcionUsuario)	Ordena el array utilizando una función de usuario. Dicha función se debe crear de forma que reciba dos parámetros y devuelva un número mayor de cero cuando el primer parámetro sea mayor que cero, cero cuando sean iguales y un número menor que cero cuando el segundo parámetro sea mayor que el primero **usort** recibe el nombre de la función entre comillas. Ejemplo: `usort($array,"funcion1")` Se supone que la ***funcion1*** ha sido creada previamente y cumple los requisitos indicados anteriormente. Esto permite crear ordenaciones personales con los arrays. Las claves se pierden (al igual que ocurre con la función **sort**) y solo se ordenan los valores, resultando un array escalar
uasort(array, funcionUsuario)	Igual que la anterior, pero se respeta la relación de las claves con sus valores
uksort(array, funcionUsuario)	Igual que la anterior, pero lo que ordena son las claves en lugar de los valores
rsort(array, [flags])	Funciona igual que **sort**, pero ordena en orden descendente
asort(array, [flags])	Idéntica a **sort**, pero los índices no se desprecian y se respetan. Es decir, se ordenan los valores y se asegura que cada valor sigue siendo accesible a través de su índice original

FUNCIÓN	SIGNIFICADO
arsort(array, [flags])	Funciona igual que asort, pero ordena en orden descendente
ksort(array, [flags])	Idéntica a **sort**, pero se ordenan las claves en lugar de los valores. La relación de la clave junto con su valor correspondiente se respeta
krsort(array, [flags])	Como la anterior pero el orden es descendente
natsort(array)	Ordena el array de modo que se ordena en la forma en la que lo haría un ser humano. Es decir el texto ***imagen2*** iría antes que ***imagen10***. Se respeta la relación entre clave y valor, por lo que no se pierden las claves.
natcasesort(array)	Igual que la anterior, pero no distingue entre mayúsculas y minúsculas.
array_multisort(array1, arg1 ó array2,...)	Permite ordenar varios arrays al mismo tiempo, indicando cada array y (opcionalmente) la forma de ordenar. La forma de ordenar se indica mediante las siguientes palabras: **SORT_ASC** (orden ascendente), **SORT_DESC** (orden descendente), **SORT_REGULAR** (orden normal), **SORT_NUMERIC** (ordenación numérica), **SORT_STRING** (ordenación para texto). Ejemplo: `array_multisort( $a,SORT_DES, SORT_NUMERIC, $b,SORT_STRING )` En el ejemplo, el primer array (***$a***) se ordena en descendente y de forma numérica, mientras que el segundo (***$b***) se ordena de forma textual

4.3.9.3 BÚSQUEDAS Y FILTROS EN LOS ARRAYS

FUNCIÓN	SIGNIFICADO
array_search(valorBusq, array [,estricto])	Busca el valor indicado en el array y devuelve la clave del valor en el array o **false** si no lo encuentra. El tercer parámetro (***estricto***) es opcional y en caso de valer verdadero (por defecto vale falso), solo encuentra el valor si además de ser igual, es del mismo tipo
in_array(valorBusq, array [,estricto])	Busca el valor indicado en el array, de forma estricta o no (según indique su tercer parámetro) y devuelve **true** si lo encuentra o false en caso contrario
array_key_exists(array, clave)	Devuelve verdadero si en el array existe la clave indicada

FUNCIÓN	SIGNIFICADO
preg_prep(expresionRegular, array[,flag])	Devuelve un array que contiene los valores del array que cumplen la expresión regular. Más adelante, en esta misma unidad, se explica el funcionamiento las expresiones regulares Si se usa, como tercer parámetro, la constante **PREG_GREP_INVERT**, el array resultado contiene los elementos que no cumplen la expresión regular.
array_filter(array,función)	El segundo parámetro es el nombre de una función existente, indicada entre comillas. Dicha función se creará de modo que tenga un solo parámetro y devuelva verdadero o falso en base a una condición que se valorará en el parámetro. **array_filter** aplica dicha función a cada elemento del array y devuelve un nuevo array que contiene los elementos del primer array que, enviados como parámetro a la función, hacen que ésta devuelva falso. Es decir, elimina todos los elementos que no cumplan la condición establecida por la función.

4.3.9.4 FUNCIONES PARA MANEJO ALEATORIO

FUNCIÓN	SIGNIFICADO
shuffle(array)	Reordena de forma aleatoria el array
array_rand(array)	Devuelve un índice aleatorio del array

4.3.9.5 FUNCIONES DE MANIPULACIÓN DEL CONTENIDO DEL ARRAY

FUNCIÓN	SIGNIFICADO
array_keys(array)	Devuelve un array escalar que contiene todas las claves del array
array_values(array)	Devuelve un array escalar que contiene todos los valores del array. Los índices del array original se eliminan
array_flip(array)	Intercambia las claves por los valores en el array. Es decir: las claves pasan a ser valores y los valores las claves de dichos valores
array_combine(array1, array2)	Toma los valores de ambos arrays y devuelve un nuevo array donde las claves son los valores del primer array y los valores son los del segundo array Ambos arrays deben contener el mismo número de valores (de otro modo ocurrirá un error). Las claves de ambos arrays se ignoran en el resultado final.

FUNCIÓN	SIGNIFICADO
array_fill(inicio,n,valor)	Devuelve un array que contiene ***n*** elementos, todos ellos con el valor indicado. El parámetro ***inicio*** indica cuál será el primer índice del array (es numérico); los siguientes índices serán los números consecutivos a ese inicial
array_fill_keys(claves,valor)	Devuelve un array a partir de otro, indicando mediante el parámetro ***claves***, cuyos valores se considerarán las claves del nuevo. Todos los elementos del nuevo array valdrán el valor indicado
array_unique(array, [flags])	Elimina los valores duplicados en el array. El segundo parámetro sirve para indicar la forma de comparar los valores y utiliza las mismas posibilidades que las indicadas en la función **sort**
array_count_values(array)	Genera un nuevo array en el que los índices son los valores del array que recibe como parámetro y los valores son el número de veces que cada valor antiguo aparece en el array original
array_sum(array)	Suma todos los valores del array y retorna el resultado
array_product(array)	Multiplica, entre sí, todos los valores del array y retorna el resultado
array_shift(array)	Retira el primer elemento del array, desplazando a todos los demás elementos en el mismo. Es decir, no deja el hueco del eliminado. Los índices funcionan de forma escalar, es decir comienzan a numerarse desde el número cero. La función devuelve el elemento eliminado
array_unshift(array, valor1 [,valor2,...])	Coloca al principio del array los valores indicados y devuelve el tamaño del nuevo array
array_pop(array)	Retira del array su último elemento y lo devuelve como resultado
array_push(array, valor1 [, valor2,...])	Coloca los valores indicados (al menos uno) al final del array original. Combinado con **array_pop** permite simular pilas

array_map(función, array)	Indica una función existente (pero entrecomillada) y devuelve un array, resultado de aplicar la función indicada a cada elemento del array. La función se debe definir usando un solo parámetro, ese parámetro representa a cada elemento del array
	
	`function doble($v){`
	`  return $v*2;`
	`}`
	
	`$a=array(1,2,3,4,5);`
	`$b=array_map("doble",$a)`
	`print_r($b);`
	`//Escribiría: 2,4,6,8,10`

FUNCIÓN	SIGNIFICADO
array_walk_recursive(array, función)	Indica una función existente (pero entrecomillada) y la aplica a cada elemento del array. Al definir la función que se usa, se utilizará un solo parámetro que representa a cada elemento del array. En el caso de que los valores del array sean otros arrays, se ejecutará la función para dichos arrays

array_pad(array,n,valor)	Devuelve un nuevo array copia del primero, al cual se le añaden elementos con el valor indicado, hasta alcanzar el tamaño marcado por el parámetro ***n***. Si ***n*** es un número positivo el rellenado de valores se hace hacia la derecha y si no, hacia la izquierda. Ejemplo:
	
	`$a=array("a","b","c");`
	`$b=array_pad($a, 5, "x")`
	
	El array b será: ***a, b, c, x, x***

array_walk(array,
función)
	Indica una función existente (pero entrecomillada) y la aplica a cada elemento del array. La función al definirla usará un solo parámetro que representa a cada elemento del array. Dicho elemento se define por referencia al crear la función (es decir utiliza el operador &) de ese modo la función realmente podrá modificar el valor de cada elemento.
	
	Ejemplo:
	
	`function doble(&$x){`
	`  $x*=2;`
	`}`
	`$a=array(1,2,3,4,5);`
	`array_walk($a,"doble");`
	`print_r($a);`
	`//Escribiría: 2,4,6,8,10`

array_slice(array, posInicio [,posFin])	Devuelve una porción del array, que se indica como primer parámetro, de modo que se toman los elementos desde la posición inicial indicada por el parámetro ***posInicio***, hasta la posición final indicada por el parámetro ***posFin***. Si el parámetro ***posFin*** se omite, entonces se toma desde la posición inicial hasta el último elemento del array Las posiciones indicadas se toman de forma escalar, es decir el primer elemento tiene la posición cero, el segundo uno,... El array original no cambia, pero en el resultante los índices nuevos serán escalares; es decir, comienzan a numerar los índices desde el cero

FUNCIÓN	SIGNIFICADO
array_splice(array,posInicio [,tamaño [,arraySubst]])	Elimina los elementos del array, usado como primer parámetro de la función, desde la posición marcada mediante el parámetro ***posInicio*** hasta el final. Este parámetro puede ser negativo y entonces la posición (***posInicio***) desde la que eliminar se cuenta desde el final A la hora de indicar la ***posInicio***, se maneja el array como si fuera escalar. Es decir, el primer elemento es el cero, el segundo es el uno, etc. Indicando un **tamaño**, solo se cambian el número de elementos indicados por ese parámetro. El parámetro **arraySubst** indica una array que servirá para sustituir los elementos eliminados por los que están contenidos en el array Devuelve un array con los elementos eliminados
array_chunk(array ,tamaño [,respetarClaves])	Devuelve un array multidimensional, resultado de dividir el array que se pasa como parámetro en trozos del tamaño indicado. El parámetro opcional **respetarClaves**, si vale **true** (por defecto es **false**) respeta las claves originales; de otro modo, las claves se pierden y cada array se renumera de forma escalar.

4.3.9.6 CONVERSIÓN DE ARRAYS A PARTIR DE VARIABLES COMUNES Y VICEVERSA

FUNCIÓN	SIGNIFICADO

list(listaDeVariables)=array	Permite dar a una lista de variables valores procedentes de un array. Ejemplo:
	
	`$a=array(`
	`  979797979,`
	`  "Manuel",`
	`  1250.45`
	`);`
	`list($tlfno,$nombre,`
	`  $salario)=$a;`
	`//$telefono vale 979797979, $nombre vale`
	`//Manuel y salario vale 1250.45`

FUNCIÓN	SIGNIFICADO

compact(lista de variables)	Se le pasa una lista de variables. La lista es en realidad una lista de textos (strings) que contienen el nombre de las variables (sin el signo $). La función devuelve un array con esos valores y usa como índices del array, el nombre de las variables. Ejemplo:
	`$a="Hola";`
	`$b=176;`
	`$c=true;`
	`$array=compact("a","b","c");`
	`print_r($array);`
	
	`/*Sale:`
	`Array`
	`(`
	`[a] => Hola`
	`[b] => 176`
	`[c] => 1`
	`)  */`

4.3.9.8 COMBINACIÓN DE ARRAYS

FUNCIÓN	SIGNIFICADO
array_intersect(array1,array2[,...])	Genera un nuevo array, intersección de los indicados. En el array resultante solo aparecen los valores duplicados en todos los arrays. Se mantienen los índices, pero toma los primeros índices que aparezcan en la lista de arrays
array_intersect_key(array1,array2,...)	Genera un nuevo array en el que aparecen las claves presentes en todos los arrays indicados como parámetros
array_diff(array1,array2,...)	Genera un nuevo array en el que aparecen los valores del primer array que no están en el segundo (array de diferencia). La relación entre clave y valor se mantiene
array_diff_key(array1,array2,...)	Genera un nuevo array en el que aparecen las claves del primer array que no están en el segundo. La relación entre clave y valor se mantiene

FUNCIÓN	SIGNIFICADO

array_merge(array1,array2,...)	Une los dos arrays que se le pasan. Los valores del segundo se colocan a continuación de los del primero. Si el array es asociativo, en caso de que haya claves repetidas, solo se queda con las del segundo array
	`$a=array(`
	`"uno"=>"Pedro",`
	`"dos"=>"Antonio",`
	`"tres"=>"Santiago"`
	`);`
	`$b=array(`
	`"dos"=>"Sara",`
	`"tres"=>"Antonio",`
	`"cuatro"=>"Santiago"`
	`);`
	`$c=array_merge($a,$b);`
	`print_r($c);`
	`/* obtiene:`
	`Array`
	`(`
	`[uno] => Pedro`
	`[dos] => Sara`
	`[tres] => Antonio`
	`[cuatro] => Santiago`
	`)`
	Si el array es escalar se renumeran de nuevo todas las claves y sí aparecen las claves con índices repetidos

array_merge_recursive(a1,a2,...)	Igual que el anterior, solo que ahora si hay índices repetidos, en el array resultante, cada valor se convertirá en un array que contiene todos los valores de ese índice

4.4 STRINGS

4.4.1 INTRODUCCIÓN

La palabra String en inglés significa ***cadena***. Sin embargo, en realidad, por String se entiende una serie de caracteres, lo que normalmente llamamos simplemente **texto**. En toda aplicación informática los strings son el tipo de dato fundamental. Tanto es así que en realidad ya hemos necesitado usar strings en los apartados anteriores. Sin embargo ahora vamos a desglosar todo el funcionamiento de los strings que permite PHP.

4.4.2 ASIGNACIÓN DE STRINGS

Ya se ha explicado en la unidad anterior (Apartado 3.4.3 *"Asignación de valores"*, en la página 126). Para asignar valores a un **string** basta con utilizar el operador =. El texto literal se debe entrecomillar, ya sea entre comillas dobles o simples.

```
$string1="Soy un texto";
$string2='Yo también';
```

En el caso de los caracteres especiales, éstos solo se pueden indicar dentro de comillas dobles.

4.4.3 CONCATENACIÓN DE TEXTOS

Como se ha comentado en apartados anteriores, el operador de concatenación de textos es el símbolo del punto (.). Ejemplos:

```
$texto1="Hola";
$texto2="a todos y todas";
$texto3=$texto1." ".$texto2;
echo $texto3;
```

Por pantalla saldría ***Hola a todos y todas***. Como se observa en el ejemplo se pueden concatenar tanto variables de tipo strings como textos literales (se encadena el contenido de la variable ***$texto*** se añade un espacio en blanco y se concatena con el contenido de la variable ***texto2***).

Podemos usar el operador de concatenación y asignación:

```
$texto1="Hola ";
$texto1.="a todos y todas";
echo $texto1;
```

El resultado es como en el texto anterior ya que el símbolo .= permite añadir al final de la variable el String que se indique.

4.4.4 USO DE VARIABLES EN STRINGS. USO DE LLAVES

Una de las grandes capacidades PHP es la de poder traducir contenidos de variables por su valor dentro de un texto. Ejemplo:

```
$libre="free";
echo "En inglés libre se dice $libre";
      //muestra: En inglés libre se dice free
```

Sin embargo, este código daría error:

```
echo "<br />Me gusta patinar en $librestyle";
```

Ya que no existe la variable ***$librestyle***, y no hay razón para que haya que considerar ***$libre*** y luego ***style*** como textos separados. Sí funciona mediante:

```
echo "<br />Me gusta patinar en {$libre}style";
//Sale: Me gusta patinar en freestyle
```

Las llaves son además imprescindibles para utilizar con arrays; aunque, en cualquier caso el operador de concatenación de texto puede ayudarnos de forma más clara, con expresiones más complejas.

```
echo "El doble de x es".x*2."<br />";
```

4.4.5 MANEJO DE STRINGS COMO ARRAYS DE CARACTERES

Se puede usar un string como si fuera un array donde cada elemento lo forma cada carácter del string. Esto nos permite dar más versatilidad a los arrays Ejemplo:

```
$string1="Este es el texto de prueba";
$string1[6]="X";
echo $string1; //Escribe: Este eX el texto de prueba
```

Aunque realmente no se considera un array; de hecho, no se admite el uso de **foreach** para recorrer strings, ni tampoco funcionarían ninguna de las funciones de arrays.

4.4.6 CADENAS HEREDOC

Permiten asignar valores de texto extensos, sin usar comillas. A través de delimitadores. Ejemplo:

```
$nombre="Jorge";
$texto=<<<fin
   Mi querida amiga <br />
   escribo estas líneas esperando que me leas. <br />
   Firmado: $nombre <br />
fin;
   echo $texto;
```

En el ejemplo se ha remarcado el texto que se almacena en la variable ***$texto***. El texto a asignar es el que sigue al símbolo de inserción de documento (<<<) y al **marcador de texto**, que es un grupo de caracteres concreto (en este caso se ha usado la palabra ***fin*** como marcador de texto). El marcador se indica inmediatamente después del símbolo <<< y vuelve a aparecer en la **primera columna** tras el último carácter que se almacenará en la variable.

Es decir, en una página web, el texto aparecería como:

```
Mi querida amiga
escribo estas líneas esperando que me leas.
Firmado: Jorge
```

4.4.7 CADENAS NOWDOC

Funciona igual que el anterior, solo que entiende que el texto es totalmente literal, por lo que no se interpretan los nombres de variable ni los códigos de escape. Para diferenciar las dos notaciones, en ésta el marcador de línea va entre comillas simples; pero solo en la declaración, no en el cierre. Ejemplo:

```
   $nombre="Jorge";
   $texto=<<<'fin'
   Mi querida amiga <br />
   escribo estas líneas esperando que me leas. <br />
   Firmado: $nombre <br />
fin;
   echo $texto;
```

Resultado:

```
Mi querida amiga
escribo estas líneas esperando que me leas.
Firmado: $nombre
```

No se ha interpretado la variable ***$nombre***, se ha entendido que es un texto normal. Esa es la diferencia con los textos **heredoc**.

4.4.8 ANEXO: FUNCIONES ESTÁNDAR DE USO CON STRINGS

PHP tiene una enorme batería de funciones para trabajar con strings. Eso nos va a facilitar enormemente realizar tareas difíciles, como validar datos de formularios, o realizar acciones avanzadas con los textos que introduzca el usuario.

4.4.8.1 FUNCIONES BÁSICAS

FUNCIÓN	SIGNIFICADO
strlen(string)	Devuelve el tamaño del String

4.4.8.2 MAYÚSCULAS Y MINÚSCULAS

FUNCIÓN	SIGNIFICADO
strtolower(texto)	Convierte el texto a minúsculas
strtoupper(texto)	Convierte el texto a mayúsculas
lcfirst(texto)	Retorna un string resultado de poner en minúsculas el primer carácter del texto (suponiendo que sea una letra).
ucfirst(texto)	Retorna un string resultado de poner en mayúsculas el primer carácter del texto (suponiendo que sea una letra).

4.4.8.3 FUNCIONES DE COMPARACIÓN

FUNCIÓN	SIGNIFICADO
strcmp(texto1, texto2)	Compara los dos textos (strings) y devuelve cero si son iguales, uno si el primero es mayor en orden alfabético (usando el código ASCII) y -1 si es mayor el segundo string
strccasemp(texto1, texto2)	Igual que la anterior pero no tiene en cuenta las mayúsculas (solo en inglés)
strnatcmp(texto1, texto2)	Igual que las anteriores, pero la comparación que hace tiene en cuenta la forma natural humana de ordenar. Así por ejemplo, el texto ***imagen2*** sería considerado menor que ***imagen11*** (en orden alfabético es mayor)
strcasenatcmp(texto1, texto2)	Igual que la anterior, pero sin considerar mayúsculas ni minúsculas
levenshtein(texto1, texto2)	Devuelve un número entero conocido como distancia **Levenshtein** que simboliza el número de modificaciones al primer texto que nos permitirían conseguir el segundo texto. De modo que si esa distancia es pequeña, los dos textos son parecidos
metaphone(texto [,fonemas])	Devuelve un código que indica la forma de pronunciar una palabra (usando el inglés), de modo que dos palabras con pronunciación similar tendrían el mismo código El parámetro ***fonemas*** indica el número máximo de fonemas a tener en cuenta (menos, más palabras parecidas habrá)
similar_text(texto1, texto2 [,porcentaje])	Devuelve el número de caracteres parecidos entre el ***texto1*** y el ***texto2***. En realidad lo interesante es usar el tercer parámetro ***porcentaje***. Este parámetro se pasa por referencia, por lo que tiene que ser una variable. Dicha variable recibe un porcentaje de similitud entre ambas cadenas de texto

4.4.8.4 FUNCIONES DE BÚSQUEDA Y REEMPLAZO

FUNCIÓN	SIGNIFICADO

strpos(texto, textoBusq)	Devuelve la posición del segundo texto dentro del primero (empieza a contar la posición desde el número cero). Si el texto buscado no se encuentra, devuelve ***false***. Ejemplo:
	`$a="Esta es la comunidad de ". "Castilla y León";`
	`echo strpos($a," Castilla");`
	Escribiría *24*, ya que ***Castilla*** empieza a aparecer en la posición 24 del String ***$a***

stripos(texto, textoBusq)	Igual que la anterior pero no tiene en cuenta mayúsculas ni minúsculas. Solo es válida con textos que no usen caracteres fuera del alfabeto inglés

strpbrk(texto, listaCars)	***listaCars*** es un string que contiene todos los caracteres que deseamos buscar en el texto de modo que busca cualquiera de esos caracteres dentro del texto
	La función si encuentra cualquiera de esos caracteres, devuelve el resto desde la posición del primer carácter que encuentre. Ejemplo:
	`echo strpbrk( "este año no voy a la montaña","ñyn");`
	Escribe (puesto que encuentra primero a la ***eñe***):
	`ño no voy a la montaña`

str_replace(textoBuscado, textoReemplazo, texto [,veces])	Localiza todas las apariciones del ***textoBuscado*** en el texto que se pasa como tercer parámetro y las cambia por el texto indicado en ***textoReemplazo*** El cuarto parámetro (opcional), ***veces***, se envía por referencia y almacenará el número de reemplazos realizados
str_ireplace(textoBuscado, textoReemplazo, texto [,veces])	Idéntica a la anterior, pero no distingue entre mayúsculas y minúsculas
substr_replace(texto, textoReemplazo, posInicial [,posFinal])	Coloca en el texto original, el texto indicado como ***textoReemplazo***, de modo que sustituya a todos los caracteres desde la posición indicada como ***posInicial***, hasta el final o hasta el número indicado con el parámetro ***posFinal***

FUNCIÓN	SIGNIFICADO
strtr(texto, origen, destino)	Cambia byte a byte (usando formato ASCII) cada carácter del texto reemplazando el primer carácter del string origen por el primer carácter del string destino, el segundo por el segundo y así sucesivamente
strtr(texto, arrayTraducción)	Hace reemplazos múltiples de caracteres en el texto y devuelve el texto resultante de realizar esos reemplazos El array que se usa como segundo parámetro, contiene datos de forma que cada índice se buscará en el texto y se reemplazará por su valor.

strtr
(continuación)
	Ejemplo:
	
	`$array=array( "a"=>"á","e"=>"é","i"=>"í",`
	`"o"=>"ó");`
	`echo strtr("este texto tiene", $array)`
	
	Escribe:
	
	`ésté téxtó tíéné`

strip_tags(texto [,noRetirables])	Retira del texto todas las etiquetas de tipo HTML o PHP que haya contenidas, excepto las que se indiquen en el parámetro ***noRetirables***, que es un string que contendrá las etiquetas que no queremos retirar seguidas. Ejemplo: `strip_tags($texto,"<p><strong")` Elimina del texto todas las etiquetas que haya excepto ***p*** y ***strong***
stripslash(texto)	Elimina en el texto todos los caracteres ***backslash*** (\). Es muy útil para texto procedente de lenguajes de programación

4.4.8.5 FUNCIONES DE EXTRACCIÓN DE TEXTO

FUNCIÓN	SIGNIFICADO
substr(texto, posInicial [,tamaño])	Extrae del string que se pasa como primer parámetro, el texto que va desde la posición indicada (empezando a contar por cero) hasta el final. O bien, extrae desde dicha posición el número de caracteres indicados por el parámetro ***tamaño***

FUNCIÓN	SIGNIFICADO

strstr(texto, textoBusq)	Busca un string dentro de otro y devuelve los caracteres desde su primera aparición hasta el final. En la respuesta incluye el texto buscado. Ejemplo:
	`$a="Esta es la comunidad de Castilla León";`
	`echo strstr($a," Castilla");`
	`//escribe: Castilla León`

stristr(texto, textoBusq)	Igual que la anterior pero no tiene en cuenta mayúsculas ni minúsculas.

strtok(string [,token])	Permite extraer un texto en base a una serie de caracteres, de modo que primero coge el primer texto dentro del string original (primer token) y en las siguientes llamadas irá devolviendo el resto de tokens (en las siguientes llamadas no se usa el primer parámetro) hasta que tras devolver el último retorna el valor falso.
	Ejemplo:
	`$s="este es el texto, le vamos a partir";`
	`$tok=strtok($s," ,");`
	`while($tok!=false){`
	`  echo $tok."\n"; $tok=strtok(" ,");`
	`}`
	Escribe:
	`este`
	`es`
	`el`
	`texto`
	`le`
	`vamos`
	`a`
	`partir`

str_repeat(texto, veces)	Devuelve un string que contiene el texto indicado repetido las ***veces*** que se indiquen en el segundo parámetro.
str_shuffle(texto)	Devuelve una copia del texto donde cada carácter se ha movido aleatoriamente. Es decir, forma un anagrama.
strrev(texto)	Devuelve un string que contiene el texto original al revés.
str_split(texto [,tamaño])	Devuelve un array en el que en cada elemento hay un trozo del texto original. El texto se trocea por tamaño de caracteres; si no se indica ***tamaño*** se divide carácter a carácter; si, por ejemplo, ***tamaño*** vale *3*, se divide de tres en tres caracteres.

<table>
<tr><th>FUNCIÓN</th><th>SIGNIFICADO</th></tr>
<tr><td>explode(
delimitador,
texto
[,límite]
)</td><td>Devuelve un array donde cada elemento del mismo es una subcadena que procede de separar el texto indicado, en base a un delimitador. El parámetro límite indica el máximo número de cadenas a extraer. Ejemplo:

<pre>$s="soy una frase, con unas,".
     "cuantas, comas";
$a=explode(",",$s);
print_r($a);</pre>
Escribirá:

<pre>Array()
   ([0]=>soy una frase
   [1]=>con unas
   [2]=>cuantas
   [3]=>comas
)</pre></td></tr>
<tr><td>implode(array)</td><td>Devuelve el string que resulta de concatenar todos los elementos del array</td></tr>
<tr><td>str_word_count(
texto
[,formato
[,listaCaracteres]]
)</td><td>Sin indicar segundo ni tercer parámetro, devuelve el número de palabras encontradas en el texto

El parámetro formato puede tomar los valores:

<table><tr><td>1</td><td>La función devuelve el número de palabras encontradas</td></tr><tr><td>2</td><td>Devuelve un array escalar con todas las palabras encontradas</td></tr><tr><td>3</td><td>Devuelve un array asociativo donde cada valor es cada palabra encontrada y su índice la posición en el texto original.</td></tr></table>
listaCaracteres es un string que contiene caracteres que en esta función se deben de considerar como parte de una palabra y no como caracteres separadores de palabras (como la coma o el punto). Es muy útil para texto escrito en cualquier lengua distinta del inglés y así considerar, por ejemplo, a la eñe como parte de una palabra</td></tr>
</table>

4.4.8.6 FUNCIONES DE LIMPIEZA DE TEXTO

FUNCIÓN	SIGNIFICADO
ltrim(texto, [caracteres])	Elimina los caracteres indicados que estén al inicio del texto. Si no se indican caracteres, se eliminan los espacios en blanco

FUNCIÓN	SIGNIFICADO
rtrim(texto, [caracteres])	Igual que la anterior, pero elimina los caracteres al final del texto
trim(texto, [caracteres])	Sin usar el segundo parámetro, elimina del texto los espacios en blanco del principio y el final. Elimina también saltos de línea, retornos de carro, tabuladores, nulos y caracteres de salto vertical El segundo parámetro permite indicar una lista de caracteres entrecomillada que serán los que se eliminen si se encuentran al final y al principio del texto (en lugar de eliminar los espacios)

4.4.8.7 OBTENER CÓDIGOS ASCII

FUNCIÓN	SIGNIFICADO
chr(códigoASCII)	Devuelve el carácter correspondiente al código ASCII indicado
ord(carácter)	Inversa a la anterior. Devuelve el código ASCII del carácter indicado

4.4.8.8 FUNCIONES DE FORMATO DE DATOS

FUNCIÓN	SIGNIFICADO
money_format(formato, número)	Devuelve un string resultante de aplicar al número que se recibe como segundo parámetro el formato de moneda indicado mediante el primer parámetro. Solo funciona en entornos compatibles con la función **strfmon** de lenguaje C (Windows no lo es) El parámetro formato es, a su vez, un string que contiene símbolos especiales para dar formato. Esa expresión está precedida por el símbolo % al que le pueden seguir uno o más de estos otros símbolos

i	Formatea el número usando los símbolos y separadores pertinentes según lo especificado por la función **setlocale** basándose en el sistema monetario internacional. Por ejemplo si antes hemos usado ***setlocale(LC_ALL,"es-ES")***, el formato del número será el correspondiente al formato monetario español
n	Idéntica al anterior pero usa el sistema nacional de moneda, según la configuración regional especificada por **setlocale**
=f	Se especifica un carácter de relleno para el número (por defecto se usa el espacio en blanco)
^	Deshabilita el separador de miles

<table>
<tr><th>FUNCIÓN</th><th>SIGNIFICADO</th></tr>
<tr><td>money_format
(continuación)</td><td>
<table>
<tr><td>+</td><td>Hace que siempre aparezca el signo del número</td></tr>
<tr><td>(</td><td>Números negativos entre paréntesis</td></tr>
<tr><td>!</td><td>Quita el símbolo de la moneda</td></tr>
<tr><td>#n</td><td>Indica la anchura de la parte entera del número</td></tr>
<tr><td>.p</td><td>Número de decimales a mostrar</td></tr>
<tr><td>-</td><td>El número se alinea a la izquierda y no a la derecha</td></tr>
</table>
</td></tr>
<tr><td>number_format(
número
[,decimales
[,caracterDecimal
[,caracterDeMiles]]]
)</td><td>Devuelve un string que contiene al número que se indica formateado de modo que aparece el separador de miles

El segundo parámetro (opcional) permite indicar cuántos decimales vamos a utilizar para mostrar el número. Los otros dos parámetros se utilizan para indicar cuál es el carácter de separador de decimales y el de miles. Ejemplo:

<code>echo number_format(1234567.892,2,",",".");
// Resultado: 1.234.567,89</code></td></tr>
</table>

4.5 CIFRADO

PHP incorpora numerosas funciones para hacer labores de cifrado. En el caso más habitual, lo que se pretende al cifrar es esconder una contraseña u otros textos que deseamos ocultar a alguien que quiera espiar la comunicación entre nuestro servidor PHP y el cliente.

La forma de solucionar el problema es utilizar funciones que convierten la contraseña en un texto ***hash*** obtenido a través de un algoritmo de cifrado. Lógicamente la idea es que el hash sea indescifrable; es decir, que a través del hash no podamos obtener el texto original.

4.5.1 ALGORITMOS DE CIFRADO

La vulnerabilidad de los hash tiene que ver con los algoritmos de cifrado. Los más populares son *MD5*, *SHA1* y *SHA256* para los que PHP dispone de varias funciones. Pero, lo cierto es que no son recomendables actualmente porque se pueden descifrar utilizando algoritmos de fuerza bruta aprovechando la gran potencia de las máquinas actuales.

En su lugar se recomienda el uso de **Blowfish**. Por fortuna, Blowfish es actualmente el algoritmo predeterminado por PHP en sus funciones de cifrado para contraseñas.

Además es muy recomendable utilizar una sal (del inglés ***salt***) que es un pequeño dato añadido que complica la obtención del texto original. Sin ese dato, se puede obtener el texto original a partir del hash utilizando las llamadas ***tablas rainbow*** en la que se almacenan textos originales y sus hash correspondientes y luego se cruzan con el hash que deseamos descifrar. Como los usuarios utilizan claves que, fácilmente, otros usuarios han utilizado, su hash puede estar disponible

en esas tablas. La sal dificulta enormemente esta técnica al añadir un segundo dato que, además, puede ser aleatorio.

Hay que tener en cuenta que para que este funcionamiento recomendable de PHP sea posible, debemos tener instalada una versión de PHP al menos igual o superior a la 5.5.

4.5.2 FUNCIÓN PASSWORD_HASH

Se trata de la función recomendada actualmente para el cifrado. Su sintaxis es:

password_hash(texto, algoritmo [,opciones])

Se explican a continuación los parámetros de esta función.

- *texto*. Es el texto a cifrar, normalmente una contraseña.
- *algoritmo*. Es un número entero que indica el algoritmo a utilizar. Se utilizan las constantes:
  - PASSWORD_DEFAULT. Utiliza el algoritmo **bcrypt** (disponible desde la versión 5.5 de PHP). De modo que utilizará automáticamente el algoritmo de cifrado más robusto que tenga disponible nuestra instalación de PHP.
  - PASSWORD_BCRYPT. Utiliza el algoritmo Blowfish. Eso asegura la compatibilidad con la función **crypt** (utilizada tradicionalmente para estos menesteres).
- *opciones*. Se trata de un array asociativo que permite utilizar estas claves:
  - **salt**. Permite indicar una sal manual al obtener el hash. Sin usar esta opción, la función genera una sal aleatoria.
  - **cost**. Coste. Un valor entre 4 y 32 que indica el número de iteraciones (en base 2) que realizará el algoritmo. Por defecto se usa 10. Poner más coste hace que se tarde más en generar la clave (a cambio de más seguridad).

La función retorna el texto hash resultante de aplicar las opciones. Ejemplo:

```
echo password_hash("prueba",PASSWORD_DEFAULT,
    array("salt"=>"123456789ABCDEFGHIJKLM"));
```

La función escribe:
`$2y$10$123456789ABCDEFGHIJKL.YsIiXtCFs/v7Qo2eLjFKjWUtnfZRpd.`

En el propio texto están especificadas las características del algoritmo utilizado:

- `$2y$`. Este texto inicial indica el algoritmo utilizado. En este caso es el código del algoritmo **Blowfish**.
- `10$`. Coste empleado. En este caso 10.
- `123456789ABCDEFGHIJKL.` Sal

- `YsIiXtCFs/v7Qo2eLjFKjWUtnfZRpd.` Texto hash

La idea es utilizar esta función para cifrar inicialmente la contraseña y así almacenarla. Cuando un usuario utilice una contraseña para verificarse, la cifraremos de la misma forma y compararemos el resultado con el que tenemos almacenado a través de **password_verify**.

4.5.3 OTRAS FUNCIONES DE CIFRADO

FUNCIÓN	RESULTADO
password_get_info(hash)	Recibe un texto hash y nos devuelve un array asociativo con información sobre el texto. En ese array tendremos los siguientes índices: ■ **algo**. Devuelve un entero relacionado con la constante predefinida que representa el algoritmo utilizado como PASSWORD_DEFAULT por ejemplo ■ **algoName**. Nombre del algoritmo, por ejemplo **bcrypt** ■ **options**. Array de opciones utilizado al cifrar
password_verify(texto,hash)	Devuelve verdadero si el hash es correcto para ese texto. Es la función principal de verificación de una contraseña

Ejemplo de uso de *password_verify*:

```
if(password_verify("prueba",
    "$2y$10$123456789ABCDEFGHIJKL.YsIiXtCFs/v7Qo2eLjFKjWUtnfZRpd.")){
    echo "Contraseña correcta!";
}
```

4.6 EXPRESIONES REGULARES

Se trata de uno de los elementos más interesantes a utilizar por parte de los programadores. Se utilizan, sobre todo, para establecer un patrón que permita validar el contenido correcto de un texto. Ese patrón permite búsquedas avanzadas, criterios avanzados de verificación de claves o códigos, etc.

PHP permite el uso de dos tipos de expresiones regulares:

- POSIX (de *Portable Operating System Interface*) 1003.2 correspondientes a un estándar aceptado por el organismo IEEE (*The Institute of Electrical and Electronics Engineers*) muy influyente en normas electrónicas y que se basa en la sintaxis del sistema **Unix**. Las funciones compatibles con este formato comienzan por la palabra **ereg**.
- PCRE (*Perl Compatible Regular Expressions*). Incorporado a PHP desde la versión 4.2, procedente del lenguaje **Perl**, lenguaje precisamente famoso por su uso de las expresiones regulares. Este es el formato que más se usa, de hecho, se recomienda no utilizar

el anterior. Las funciones que utilizan este formato de expresión regulares comienzan por la palabra **preg**. Por otro lado, este formato es compatible con **Unicode**, por lo que también es más recomendable para lenguas que usan símbolos fuera del ASCII original como el castellano.

4.6.1 FORMATO DE LAS EXPRESIONES REGULARES PCRE

Las expresiones regulares utilizan símbolos especiales para indicar el patrón correspondiente. Las expresiones regulares de tipo Perl deben ir delimitadas por algún carácter. De manera clásica se suele delimitar con la barra de dividir (/), pero vale cualquier delimitador (símbolo de sostenido, menor y mayor, etc). Ejemplo:

```
$expReg="/^[A-Z]{8}[0-9]$/"; //8 letras y un número
```

En el ejemplo, la variable ***$expReg*** encierra una expresión regular. Como se observa al ser un texto, va entrecomillado; dentro de las comillas hay otro delimitador, la barra de dividir, para marcar la expresión regular.

Los símbolos y expresiones que se pueden utilizar dentro de la expresión regular son:

SÍMBOLO	SIGNIFICADO
c	Un carácter cualquiera: por ejemplo ***a***, ***H***, ***ñ***, ***etc.*** Dentro de una expresión regular, obligaría a que aparezca ese carácter en la posición en la que se indica
cde	Siendo *c*, *d*, y *e* caracteres, indica que esos caracteres deben aparecer en ese orden en la expresión. Por ejemplo, ***ola*** obligaría a que el texto que se está comprobando tenga dentro ese mismo texto
(x)	Permite indicar un subpatrón dentro del paréntesis. Ayuda a formar expresiones regulares complejas
.	Cualquier carácter. El punto indica que en esa posición puede ir cualquier carácter. Por ejemplo ***amig.*** validaría como textos válidos a ***amigo*** y a ***amiga***
^x	Comenzar por. Indica el String debe empezar por la expresión *x*. Por ejemplo *^H* obliga a que el texto empiece por la letra *H*
x$	Finalizar por. Indica que el String debe terminar con la expresión *x*
x+	La expresión a la izquierda de este símbolo se puede repetir una o más veces
*x**	La expresión a la izquierda de este símbolo se puede repetir cero o más veces
x?	La expresión a la izquierda de este símbolo se puede repetir cero o una veces
x{n}	Significa que la expresión *x* debe aparecer *n* veces, siendo *n* un número entero positivo
x{n,}	Significa que la expresión *x* debe aparecer *n* o más veces
x{m,n}	Significa que la expresión *x* debe aparecer de *m* a *n* veces

SÍMBOLO	SIGNIFICADO
x|*y*	La barra indica que las expresiones *x* e *y* son opcionales, se puede cumplir una u otra
c-*d*	Cumplen esa expresión los caracteres que, en orden ASCII, vayan del carácter *c* al carácter *d*. Por ejemplo *a*-*z* representa todas las letras minúsculas presentes en el código ASCII
[*cde*]	Indica que es válido cualquiera de los caracteres *c*, *d* ó *e*
[*c*-*d*]	Indica que es válido cualquier carácter ASCII entre los caracteres c y d. Por ejemplo [*A*-*Z*] significa que valer cualquier letra mayúscula
[^*x*]	No es válido ninguno de los caracteres que cumplan la expresión *x*. Por ejemplo [^*dft*] indica que no son válidos las caracteres *d*, *f* ó *t*
\d	Dígito, vale cualquier dígito numérico
\D	Todo menos dígito
\s	Espacio en blanco
\S	Cualquier carácter salvo el espacio en blanco
\w	*Word*, carácter válido dentro de los que PHP considera para identificar variables. Es decir, letras, números o el guion bajo
\W	Todo lo que no sea un carácter de tipo *Word*
\n	Nueva línea
\t	Tabulador
\c	Permite representar el carácter *c* cuando este sea un carácter que de otra manera no sea representable (como [,], /, \,...). Por ejemplo \\ es la forma de representar la propia barra invertida
\x*ff*	Permite indicar un carácter **Unicode** mediante su código hexadecimal.
\p{*xx*}	Indica un carácter que cumpla la propiedad Unicode indicada con los códigos *xx*. Algunos códigos interesantes son:

CÓDIGO	SIGNIFICADO
L	Letra (incluye *Ll*, *Lm*, *Lo*, *Lt y Lu*)
Lu	Letra mayúscula
Ll	Letra minúscula
Lt	Letra tipo título (primera letra mayúscula, resto minúsculas)
Lm	Letra modificadora
Lo	Otra letra (que no sea Ll, Lu o Lm)
Lt	Letra de título
Lu	Letra mayúscula
C	Carácter de control
S	Símbolo (matemáticos, moneda, viñetas, bordes, etc.)

<table>
<tr><th>SÍMBOLO</th><th>SIGNIFICADO</th></tr>
<tr><td>(\p{xx})
(continuación)</td><td>
<table>
<tr><th>CÓDIGO</th><th>SIGNIFICADO</th></tr>
<tr><td>Sm</td><td>Símbolo matemático</td></tr>
<tr><td>Sc</td><td>Símbolo de moneda</td></tr>
<tr><td>So</td><td>Otros símbolos</td></tr>
<tr><td>N</td><td>Número</td></tr>
<tr><td>Nd</td><td>Número decimal</td></tr>
<tr><td>Nl</td><td>Número alfabético. Por ejemplo un número romano.</td></tr>
<tr><td>No</td><td>otros símbolos numéricos</td></tr>
<tr><td>P</td><td>Carácter de puntuación</td></tr>
<tr><td>Pd</td><td>Guión (sea corto o largo)</td></tr>
<tr><td>Ps</td><td>Apertura de paréntesis, corchete, llave o símbolo similar</td></tr>
<tr><td>Pd</td><td>Cierre de paréntesis, corchete, llave o símbolo similar</td></tr>
<tr><td>Pi</td><td>Comillas de inicio (no valen las rectas)</td></tr>
<tr><td>Pf</td><td>Comillas finales (no valen las rectas)</td></tr>
<tr><td>Pc</td><td>Carácter de conexión, como el guión bajo</td></tr>
<tr><td>Po</td><td>Cualquier otro carácter de puntuación</td></tr>
<tr><td>Z</td><td>Separador</td></tr>
<tr><td>Zl</td><td>Separador de línea</td></tr>
<tr><td>Zp</td><td>Separador de párrafo</td></tr>
<tr><td>Za</td><td>Separador de espacio</td></tr>
<tr><td>alfabeto</td><td>Permite indicar un código de alfabeto. Por ejemplo \p{Greek}, vale para cualquier carácter del alfabeto griego)</td></tr>
</table>
</td></tr>
<tr><td>\P{xx}</td><td>Carácter que no cumpla la propiedad xx (donde xx puede ser cualquiera de los códigos de la tabla anterior)</td></tr>
<tr><td>i</td><td>La letra i al final de los delimitadores permite ignorar mayúsculas y minúsculas.</td></tr>
</table>

4.6.2 PROBLEMAS CON UNICODE

Aunque, en teoría, hay muchas opciones para trabajar con texto Unicode, lo cierto es que PHP es un lenguaje que no está preparado para ello. De ahí que el modificador **\p** sea de obligado uso cuando estamos validando textos Unicode. Es el caso de los símbolos propios del español como las letras acentuadas, la eñe, etc. Observando este código:

```
$exp="/^[\p{L}]+$/";
echo preg_match($exp,"Español");
```

Debería escribir verdadero (en pantalla sale el valor 1), sin embargo, lo habitual en cualquier instalación de PHP, es que escriba falso (un cero o incluso nada). La expresión ***$exp*** debería vali-

dar cualquier string que contenga letras en cualquier alfabeto, pero no considera que el carácter ñ sea una letra. Realmente la expresión solo suele funcionar para letras del código ASCII, es decir, letras del alfabeto inglés.

Afortunadamente tenemos solución. Si escribimos el texto *(*UTF8)* justo tras el delimitador, indicaremos que la validación se debe hacer considerando que estamos utilizando el formato *Unicode UTF8*, realmente el habitual hoy en día.

Así el ejemplo anterior funcionaría si escribimos:

```
$exp="/(*UTF8)^[\p{L}]+$/";
echo preg_match($exp,"Español"); //ahora escribe verdadero
```

4.6.3 FUNCIONES DE EXPRESIONES REGULARES

FUNCIÓN	SIGNIFICADO
preg_match(patrón, string [, arrayResult] [, flag] [,despl])	Utiliza una expresión regular (parámetro ***patrón***) para validar que el string utilizado como segundo parámetro la cumple. Devuelve verdadero si se cumple la expresión Si se indica el parámetro **arrayResult,** que tiene que ser un array, entonces en él se almacena el texto dentro del string que cumpla ese patrón. De modo que el elemento cero del array será el texto completo que cumpla todo el patrón, el elemento 1 será el texto que cumpla el primer subpatrón que se haya definido (los subpatrones se indican entre paréntesis), el elemento 2, el segundo subpatrón, etc. El parámetro **flags** permite utilizar la constante **PREG_OFFSET_CAPTURE** para que el array anterior almacene además de cada texto que cumpla la condición, la posición en la que estaba ese texto dentro del array original (lo coloca en el array detrás de cada texto que cumpla la condición) **despl** permite indicar el índice dentro del String por el que se empezará a buscar el patrón (si no se indica este parámetro comenzamos a buscar por el principio
preg_match_all(patrón, string [, arrayResult][, flag] [,despl])	Igual que la anterior, solo que cuando encuentra el patrón, sigue buscando en el resto del texto para buscar la siguiente aparición. Esto la hace más útil para usar arrays de resultados

preg_split(
patrón, string
[, límite
[, flags]]
)
	Divide el string indicado según el patrón de expresión regular indicado y devuelve un array que contiene cada trozo obtenido del string. Ejemplo:
	`$s="texto a  dividir, palabra a palabra";`
	`print_r (preg_split("/[\s,]+/",$s));`

FUNCIÓN	SIGNIFICADO

preg_split *(continuación)*	Sale:
	`Array`
	`(`
	`[0] => texto`
	`[1] => a`
	`[2] => dividir`
	`[3] => palabra`
	`[4] => a`
	`[5] => palabra`
	`)`
	El parámetro opcional ***límite*** indica el máximo de trozos a obtener

preg_replace(patrón, reemplazo, string [, límite] [, cuenta])	Busca el patrón en el string indicado y cuando lo encuentra, devuelve un nuevo string resultado de sustituir el patrón encontrado por el texto de reemplazo especificado. Escribe: `Este es un documento PHP, y no de la isla de PHP` El patrón puede incluso ser un array con varios patrones: si es así, se sustituye cada patrón del array en el string por el texto de reemplazo indicado El parámetro opcional ***límite*** pone un tope al número de reemplazos efectuados. Por ejemplo si vale 1, solo se reemplaza la primera aparición del patrón. Por defecto vale -1 (es decir, no hay límite) La variable ***cuenta,*** si se utiliza, sirve para almacenar el número de reemplazos efectuados
preg_quote(texto [,delimitador])	Hace que en el string ***texto*** que se pasa como primer parámetro, todos los símbolos habituales de expresión regular: \ + * ? [^] $ ( ) { } = ! < > | : - pasen su forma ***escapada*** y así no se les considere como carácteres de significado especial. Lo que hace es añadir la barra invertida para provocar su secuencia de escape. Por ejemplo, el signo $, se convertiría en \$ En el segundo parámetro (opcional) se indica el carácter delimitador de la expresión regular (tradicionalmente la barra normal /) que será también convertido a su forma escapada (\/)
preg_grep(patrón,array [,flags])	Devuelve un nuevo array que contiene todos los elementos del array que se pasa como segundo parámetro, que cumplan la expresión regular indicada en el primer elemento

FUNCIÓN	SIGNIFICADO
preg_grep *(continuación)*	En el parámetro **flags** se puede indicar la constante **PREG_GREP_INVERT**, en cuyo caso se devolverán los valores que no cumplan la expresión regular

4.7 FUNCIONES DE FECHA

PHP proporciona numerosas funciones que nos facilitan el trabajo con fechas. En PHP se usa el formato de fecha y hora del sistema Unix, por lo que muchas funciones requieren las fechas en este formato. Hay otras muchas funciones que nos ayudan a crear fechas en ese formato a partir de un texto que representa una fecha, utilizando un formato más humano.

FUNCIÓN	SIGNIFICADO
time()	Devuelve la fecha y hora actual en el formato nativo (Unix) de PHP
strftime(formato[,fecha])	Devuelve un texto que representa la fecha actual (o la que se indique como segundo parámetro) en el formato regional establecido con la función **setlocale**. El formato puede incluir estos símbolos:

%a	Día de la semana (tres letras)
%A	Día de la semana (completo)
%d	El día del mes con dos dígitos (del 01 al 31)
%e	El día del mes (de 1 a 31)
%j	Día del año, 3 dígitos (del 001 al 366)
%u	Día de la semana (del 1 al 7)
%w	Día de la semana (del 0 al 6)
%W	Número de semana del año, comenzando con el primer Domingo como la primera semana
%b	Nombre del mes con tres letras
%B	Nombre del mes completo
%m	Número del mes en dos cifras (del 01 al 12)
%C	Número de siglo (por ejemplo 21)
%y	Año en dos cifras
%Y	Año en cuatro cifras
%H	Hora en formato 24 horas (del 00 al 23), con dos dígitos
%k	Hora en formato 24 horas (del 0 al 23), uno o dos dígitos
%I	Hora en formato de 12 horas (01 al 12), dos dígitos
%l	Hora en formato de 12 horas (01 al 12), uno o dos dígitos
%M	Minutos en dos cifras (01 al 59)

<table>
<tr><th>FUNCIÓN</th><th>SIGNIFICADO</th></tr>
<tr><td>strftime
(continuación)</td><td>
<table>
<tr><td>%p</td><td>'AM' o 'PM' en MAYÚSCULAS basados en la hora dada</td></tr>
<tr><td>%P</td><td>'am' o 'pm' en minúsculas basados en la hora dada</td></tr>
<tr><td>%S</td><td>Segundos en dos dígitos (del 00 al 59)</td></tr>
<tr><td>%X</td><td>Representación preferida de la hora basada en la configuración regional, sin la fecha</td></tr>
</table>
Ejemplo:
<pre>
setlocale(LC_ALL,"esp_ESP");
echo strftime("%d/%B/%Y",time());
//Sale por ejemplo: 1/Mayo/2015
</pre>
</td></tr>
<tr><td>mktime(
[hora [,minuto
[,segundo [,mes
[,día [,año [época]]]]]]]
)</td><td>Crea una fecha y hora (en formato UNIX) a partir de los parámetros indicados. Los que no se indiquen, se toman de la fecha y hora actuales

El último parámetro (época) indica con un 1 que estamos en horario de verano y con un -1 en el de invierno</td></tr>
<tr><td>strptime(fecha, formato)</td><td>Devuelve un array asociativo que contiene como valores los datos de la fecha indicada. El formato cumple los posibles valores del parámetro formato de la función strftime explicada anteriormente</td></tr>
<tr><td>local_time(
[fecha [,asociativo]]
)</td><td>Devuelve la fecha actual (o la que se indique como primer parámetro) en forma de array en el que cada elemento contiene cada parte de la fecha y hora

Se puede indicar una fecha concreta y un valor verdadero como segundo parámetro para indicar que deseamos un array asociativo en lugar de escalar</td></tr>
<tr><td>strtotime(texto)</td><td>Analiza el texto y devuelve la fecha correspondiente. Ejemplos de uso (extraídos de la página oficial de ayuda de PHP, php.net):
<pre>
<?php
echo strtotime("now"),"\n";
echo strtime("10 September 2000"), "\n";
echo strtotime("+1 day"), "\n";
echo strtotime("+1 week"), "\n";
echo strtotime("+1 week 2 ".
   "days 4 hours 2 seconds"),
   "\n";
echo strtotime("next Thursday"),"\n";
echo strtotime("last Monday"),"\n";
?>
</pre>
</td></tr>
</table>

FUNCIÓN	SIGNIFICADO
checkdate(mes, día, año)	Devuelve verdadero si la fecha indicada de esta forma es válida
date_default_timezone_get()	Devuelve la zona horaria en uso (por ejemplo ***Europe/Berlin***)
date_default_timezone_set(zona)	Establece la nueva zona horaria. El texto debe ser estándar, por ejemplo (***Europe/Berlin***)

4.8 PRÁCTICAS RESUELTAS

Práctica 4.1: Uso de arrays con formularios

- Crea una página PHP que permita elegir una serie de artículos de una tienda online mediante checkbox.
- Cada checkbox permite seleccionar un artículo, en el que se indica su precio.
- Tras pulsar el botón **Enviar** del formulario, se nos indicará el detalle de la compra, así como el total de lo que hemos comprado.

SOLUCIÓN: PRÁCTICA 4.1

Para solucionar esta práctica, creamos primero un formulario. En este formulario usamos como índice del array el nombre del artículo seleccionado y como valor el precio del mismo. Hemos optado por enviar los datos mediante la acción POST de http. El código podría ser este:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Tienda on-line</title>
</head>
<body>
<h1>Seleccione los artículos que desea comprar</h1>
<form action="practica1.php" method="post">
    <input type="checkbox" name="articulo['Bolígrafo rojo']"
                    value=".35" id="boliRojo"/>
    <label for="boliRojo">Bolígrafo Rojo (35 céntimos)</label><br/>
    <input type="checkbox" name="articulo['Bolígrafo azul']"
                    value=".35" id="boliAzul"/>
    <label for="boliAzul">Bolígrafo Azul (35 céntimos)</label><br/>
    <input type="checkbox" name="articulo['Lapicero grueso']"
                    value=".27" id="lapizGrueso"/>
    <label for="lapizGrueso">Lapicero grueso (27 céntimos)</label><br/>
    <input type="checkbox" name="articulo['Lapicero fino']"
                    value=".30" id="lapizFino"/>
    <label for="lapizFino">Lapicero fino (30 céntimos)</label><br/>
    <input type="checkbox" name="articulo['Goma de borrar']"
                    value=".35" id="goma"/>
    <label for="goma">Goma de borrar (35 céntimos)</label><br/>
    <button>Enviar</button>
</form>
</body>
</html>
```

Por lo tanto, este formulario envía un array llamado **articulo**. Cada elemento del array contiene como clave el nombre del artículo asociado al precio del mismo.

De esta forma, los cálculos son fáciles, basta recorrer el array (comprobando primero que realmente lo hemos recibido) e ir mostrando cada índice y valor, a la vez que los valores se acumulan en una variable (llamada ***$suma***) cuyo valor final mostraremos tras la lista . El código del archivo **practica1.php** encargado de mostrar lo que hemos comprado (además del precio total de la compra), sería:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Lista de compra</title>
</head>
<body>
<h1>Artículos comprados</h1>
<?php
if(isset($_POST["articulo"])){
    $articulo=$_POST["articulo"];
    if(is_array($articulo)){
        echo "<ul>";
        $suma=0;
        foreach($articulo as $nombre=>$valor){
            echo "<li>$nombre, precio: $valor &euro;</li>";
            $suma+=$valor;
        }
        echo "</ul>";
        echo "<p><strong>Total: $suma &euro;</strong></p>";
    }
    else{
        echo "<p>Los datos no se han enviado correctamente</p>";
    }
}
else{
    echo "<p>No se han comprado artículos</p>";
}
?>
</body>
</html>
```

Práctica 4.2: Clasificación de la liga de fútbol

- Crea un array que contenga todos los equipos de fútbol de primera división española, al final de la temporada 2015 y los puntos que consiguieron.

- La lista no hace falta que esté ordenada.
- A través de ese array, consigue que aparezca un formulario con un cuadro combinado que permita elegir el equipo y, tras hacerlo, se nos indiquen los puntos del equipo y su posición en la clasificación.

SOLUCIÓN: PRÁCTICA 4.2

Solucionaremos la práctica con un solo archivo. Se creará un array con la clasificación (no hace falta que esté ordenado) y después ordenaremos por los puntos. Cuando dos equipos tengan los mismos puntos nuestro programa no será fiel a la clasificación real; pero de otro modo tendríamos que hacer un array más complicado, no lo haremos en esta práctica.

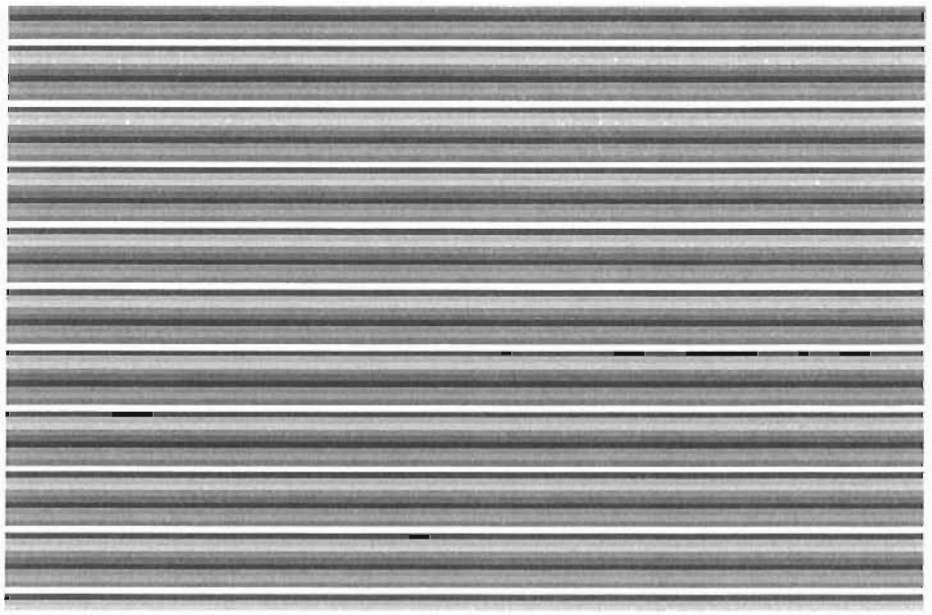
Tras esa ordenación, generamos un segundo array que contiene como valores, los nombres de los equipos ordenados según la clasificación que hemos hecho. Finalmente el array original le volveremos a ordenar, esta vez en orden alfabético que es como le mostraremos a través del cuadro combinado. El código es el siguiente (suponemos que el archivo se llama **practica2.php**):

```
<?php
$listaEquipos=array(
    "F.C. Barcelona"=>94,
    "Real Madrid"=>92,
    "Atlético Madrid"=>78,
    "Valencia"=>77,
    "Sevilla"=>76,
    "Villarreal"=>60,
    "Málaga"=>50,
    "Espanyol"=>49,
    "Athlétic Bilbao"=>55,
    "Celta"=>51,
    "Real Sociedad"=>46,
    "Rayo Vallecano"=>49,
    "Getafe"=>37,
    "Eibar"=>35,
    "Elche"=>41,
    "Deportivo"=>35,
    "Almería"=>29,
    "Levante"=>37,
    "Granada"=>35,
    "Córdoba"=>20
);
arsort($listaEquipos);
/*la lista de equipos se ordena por puntos, el array clasificacion tendrá la lista ordenada de equipos por puntos pero los puntos en sí */
$clasificacion=array_keys($listaEquipos);
?>
```

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Clasificación</title>
</head>
<body>
<form action="equipos.php" method="post">
    <label for="equipo">Elija el equipo</label>
    <select name="equipo" id="equipo">
    <?php
    //Ordenamos el array para que el cuadro combinado
    //muestre los equipos en orden alfabético
    ksort($listaEquipos);
    foreach($listaEquipos as $nombre=>$p){
        echo "<option value='$nombre'>$nombre</option>";
    }
    ?>
    </select><br/>
    <button>Comprobar</button>
    <?php
    if(isset($_POST["equipo"])){
        $equipo=$_POST["equipo"];
        /* comprobamos si el equipo existe, de ser así, tomamos los
            puntos y buscamos su posición en clasificación
            Podría no existir si alguien manipula los parámetros
        */
        if(isset($listaEquipos[$equipo])){
            $puntos=$listaEquipos[$equipo];
            $posicion=array_search($equipo,$clasificacion)+1;
            echo "<p>El $equipo tiene $puntos puntos, ahora mismo".
                " es el $posicion"."º en la clasificación </p>";
        }
        else{
            echo "<p>Equipo innexistente.".
                " Has intentado liarme</p>";
        }
    }
    ?>
</form>
</body>
</html>
```

Práctica 4.3: Bandas de color aleatorio

- Crea una página rellenada por 100 franjas horizontales de color aleatorio.
- Las franjas deben ocupar toda la página, la altura, por lo tanto, será calculada de forma que se reparta equitativamente entre cada franja
- Además realmente solo habrá 10 colores distintos. De modo que las 10 primeras franjas serán realmente aleatorias, pero luego se repiten en el mismo orden hasta rellenar la página
- Ejemplo de resultado:



SOLUCIÓN: PRÁCTICA 4.3

En principio el problema de crear 100 franjas aleatorias no tiene por qué requerir utilizar arrays, pero al repetirse los colores, entonces debemos almacenar los 10 primeros colores y lo más cómodo es utilizar el array. Código:

```
<?php
/*cálculo de las 10 primeras franjas el array contendrá 10 capas (div)
usando el atributo style con el color aleatorio de cada franja */
for($i=0;$i<10;$i++){
    $rojo=mt_rand(0,255);
    $verde=mt_rand(0,255);
    $azul=mt_rand(0,255);
    $arraycolor[$i]="background-color:rgb($rojo,$verde,$azul)";
}
?>
```

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Franjas aleatorias</title>
    <style>
        div{
            position:fixed;
            width:100%;
            height:1%;
        }
        body{
            margin:0;
            font-size:1px;
        }
    </style>
</head>
<body>
<?php
//escribimos cada franja 10 veces
$top=0;
for($i=1;$i<=10;$i++){
    foreach($arraycolor as $j=>$color){
        //Esta es la línea más compleja:
        //$color coloca el código de color de cada capa
        //$top% => va subiendo la coordenada vertical de cada capa
        //         de 1 en 1 en %
        echo "<div style='$color;top:$top%;'></div>";
        $top++;
    }
}
?>
</body>
</html>
```

Práctica 4.4: Función de dibujo de arrays

- Crea una función llamada **dibujarArray** que reciba un array y escriba el array usando una tabla HTML de dos columnas, en la primera aparecerán los índices (esta primera estará sombreada de gris) y en la segunda los valores.

- Ejemplo de resultado:

índice	valores
Palencia	80000
Valladolid	306000
Murcia	439000
Albacete	170000
Barcelona	1600000
A Coruña	250000

SOLUCIÓN: PRÁCTICA 4.4

Se expone, a continuación, el código de la función y el código de una página web con ejemplos de su uso (se utiliza el array de la tabla anterior)

```
<?php
function dibujar($array){
    echo "<table ".
      "style='border:1px solid black;border-collapse: collapse'>";
    echo "\t<tr style='background-color:black;".
         "color:white;border:1px solid black'>\n";
    echo "\t\t<th>Índices</th><th>Valores</th>\n";
    echo "\t</tr>\n";
    foreach($array as $i=>$v){
        echo "\t<tr style='border:1px solid black'>\n";
        echo "\t\t<td style='background-color:gray'>$i</td>";
        echo "\t\t<td>$v</td>\n";
        echo "\t</tr>\n";
    }
    echo "</table>";
}
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
<?php
    $localidades=array("Palencia"=>8000,"Valladolid"=>306000,
        "Murcia"=>439000,"Albacete"=>170000,"Barcelona"=>160000,
        "A Coruña"=>25000);
    dibujar($localidades);
```

```
    echo "<br>";
    $simbolos=array("Au"=>"Oro","Ag"=>"Plata",
                    "Hg"=>"Mercurio","H"=>"Hidrógeno");
    dibujar($simbolos);

?>
</body>
</html>
```

Práctica 4.5: Validación del NIF

- Crea una página web que pida los datos personales de una persona: (Nombre, Apellidos, Nombre de usuario, DNI o NIE, Teléfono). Después crea otra que valide los datos de esta forma:
  - Nos comunicará un error si en el nombre y los apellidos hay texto diferente de letras y/o espacios y guiones. Cualquier otro símbolo provocará un error.
  - Para el nombre de usuario solo se admite comenzar por letra (sea mayúscula o minúscula) y después seguirán números o más letras. Mínimos tiene que haber seis caracteres.
  - El DNI tiene que cumplir las reglas de los DNI españoles: 8 números y una letra. Pero se puede especificar también un NIE, en cuyo caso consta de una letra (solo puede ser X, Y o Z) y de siete números más una letra final.
  - Además la letra final del DNI cumple esta fórmula. Los números se dividen entre 23 y se toma el resto, el resto se sustituye por una letra siguiendo este patrón:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
T	R	W	A	G	M	Y	F	P	D	X	B	N	J	Z	S	Q	V	H	L	C	K	E

  - En el caso de los NIE, para calcular la letra final se hace lo mismo, pero sustituyendo la letra X inicial por cero, si es Y por uno y si es Z por 2.

SOLUCIÓN: PRÁCTICA 4.5

La mayoría de las validaciones a realizar se pueden hacer mediante expresiones regulares. El cálculo de la letra del DNI o NIE requiere otro tipo de acciones. Inicialmente aparecerá un formulario, el cual es un documento html (suponemos que se llama ***form-practica5.html***) los datos se envían a la página ***practica5.php***.

Archivo ***form-practica5.html***:

```
<!doctype html> <html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Datos personales</title>
</head>
```

```
<body>
   <form action="practica5.php" method="POST">
      <label for="nombre">Nombre</label>
      <input type="text" id="nombre" name="nombre"><br>
      <label for="apellido1">Primer apellido</label>
      <input type="text" id="apellido1" name="apellido1"><br>
      <label for="apellido2">Segundo apellido</label>
      <input type="text" id="apellido2" name="apellido2"><br>
      <label for="usuario">Nombre de usuario</label>
      <input type="text" id="usuario" name="usuario"><br>
      <label for="nif">Número de identificación (DNI o NIE)</label>
      <input type="text" id ="nif" name="nif"><br>
      <label for="telefono">Teléfono</label>
      <input type="text" id ="telefono" name="telefono"><br>
      <button type="submit">Validar</button>
   </form>
</body>
</html>
```

Código de la página *practica5.php*:

```
<!doctype html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Validar datos personales</title>
   <style>
   .error{
      color:red;
   }
   </style>
</head>
<body>
   <?php
      $todoBien=true;
      if(isset($_POST["nombre"]) && isset($_POST["apellido1"])
        && isset($_POST["apellido2"]) && isset($_POST["nif"])
        && isset($_POST["usuario"]) && isset($_POST["telefono"])){
          $nombre=$_POST["nombre"];
          $apellido1=$_POST["apellido1"];
          $apellido2=$_POST["apellido2"];
          $nif=$_POST["nif"];
          $usuario=$_POST["usuario"];
          $telefono=$_POST["telefono"];
```

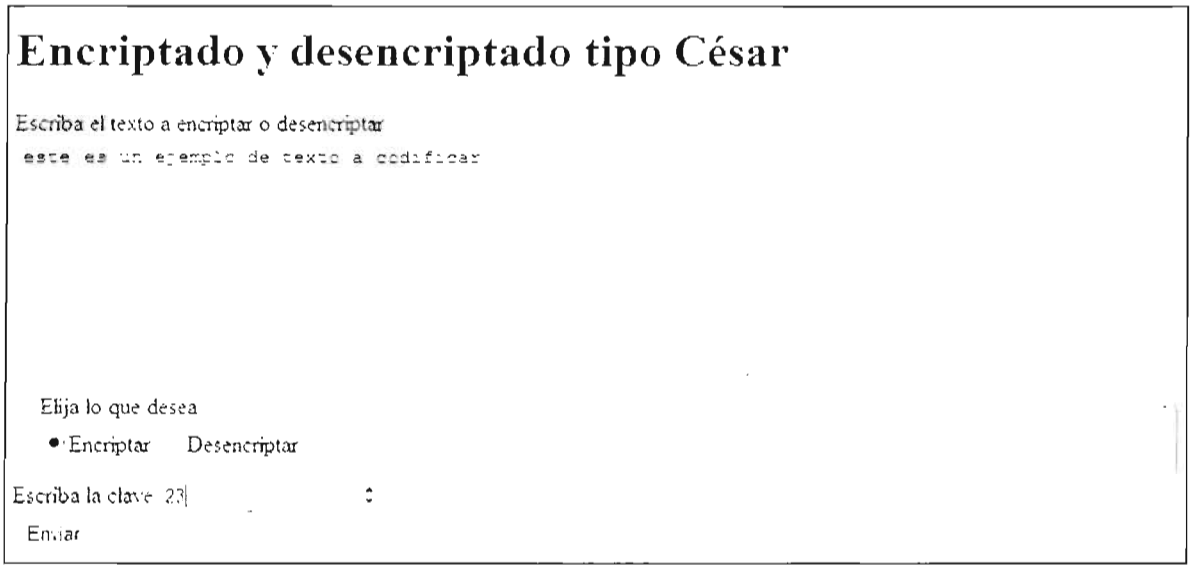
```
        //conjunto de caracteres alfabéticos del español
         $letrasValidas="A-Za-záéíóúüÁÉÍÓÚÜñÑ";
        if(preg_match("/^[$letrasValidas\- ]+$/",$nombre)==false){
          echo "<p class='error'>Nombre no válido</p>";
          $todoBien=false;
        }
        if(preg_match("/^[$letrasValidas\- ]+$/",$apellido1)==false){
          echo "<p class='error'>Primer apellido no válido</p>";
          $todoBien=false;
        }
        if(preg_match("/^[$letrasValidas\- ]+$/",$apellido2)==false){
          echo "<p class='error'>Segundo apellido no válido</p>";
          $todoBien=false;
        }
        if(preg_match("/^[$letrasValidas][{$letrasValidas}0-9]+$/",
                      $usuario)==false){
          echo "<p class='error'>Usuario no válido</p>";
          $todoBien=false;
        }
        if(preg_match("/^[0-9]{9}$/",$telefono)==false){
          echo "<p class='error'>Teléfono no válido</p>";
          $todoBien=false;
        }
        if(preg_match("/^[0-9XYZ][0-9]{7}[A-Z]$/",$nif)==false){
          echo "<p class='error'>NIF no válido</p>";
          $todoBien=false;
        }
        else{
          if($nif[0]=="X") $nif[0]="0";
          elseif($nif[0]=="Y") $nif[0]="1";
          elseif($nif[0]=="Z") $nif[0]="2";
          $numerosNIF=substr($nif, 0,8);
          $letras="TRWAGMYFPDXBNJZSQVHLCKE";
          $resto=$numerosNIF%23;
          if($nif[8]!=$letras[$resto]){
            echo "<p class='error'>Letra de NIF incorrecta</p>";
            $todoBien=false;
          }
        }
        if($todoBien==true){
          echo "<p>Datos correctos</p>";
        }
}
```

```
else{
        echo "<p class='error'>Faltan datos</p>";
      }
   ?>
   <a href="form-practica5.html">Volver al formulario</a>
</body>
</html>
```

4.9 PRÁCTICAS PROPUESTAS

Práctica 4.6: Encriptación tipo César

- Crea una página web con la apariencia que se muestra a continuación:



- Desde el formulario se invoca a una página que se encarga de encriptar el texto (o desencriptar, dependiendo de lo que elijamos). La encriptación se basa en sumar el número elegido (que debe de estar entre 1 y 99) a cada carácter del texto. El desencriptado se hace igual, pero restando. Hay que validar que el texto elegido tenga más de 10 caracteres (de otro modo se informa del error) y que la clave sea un número entre 1 y 99.
- **Nota:** Es muy interesante usar las funciones **ord** y **chr**.

Práctica 4.7: Palíndromos

- Crea un formulario en el que se pida un texto y después de enviar su contenido se nos indique si es un **palíndromo**. Ejemplos de palíndromos (no se usarán letras fuera de ASCII):
  - *dabale arroz a la zorra el abad*
  - *se verlas al reves*

- *adivina ya te opina, ya ni miles origina, ya ni cetro me domina, ya ni monarcas, a repaso ni mulato carreta, acaso nicotina, ya ni cita vecino, anima cocina, pedazo gallina, cedazo terso nos retoza de canilla goza, de panico camina, onice vaticina, ya ni tocino saca, a terracota luminosa pera, sacra nómina y animo de mortecina, ya ni giros elimina, ya ni poeta, ya ni vida*

Práctica 4.8: Palíndromos

- Crea una función PHP a la que se le pasen dos textos y nos digan si ambos son ***anagramas*** de la misma raíz. Un anagrama es un apalabra formada por los mismos caracteres que otra. Ejemplo: *ESPONJA – JAPONES*, son dos anagramas de la misma raíz.

Práctica 4.9: Validar emails

- Crea una función que reciba un array de strings (no hará falta validar la variable, supondremos que efectivamente es un array de strings). En dicho array se verificará que cada elemento es un string que contiene una dirección de email válida.
- La función devolverá verdadero si todos los elementos del array contienen direcciones de email válidas; devolverá falso de no ser así.
- Consideraremos un email válido si tenemos a la izquierda de la arroba, letras del código ASCII, guiones, números, el guion bajo o un punto (pero el punto solo puede ir entre medias de los anteriores). Tras la arroba solo permitiremos correos de los dominios ***gmail.com***, ***yahoo.es***, ***yahoo.com*** o ***hotmail.com***. Ejemplos:
  - *andres.perez18@yahoo.es* → válido
  - *eva@masn.es* →inválido
  - *pedro@er@es.es* →inválido
  - *rosa_garcia@googe.com* →inválido
  - *rosa_garcia@gmail.com* →válido
  - *pedro.diez.crespo@yahoo.com* →válido
  - *pedro..diez...@yahoo.es* →inválido, no puede haber dos puntos seguidos, delante y detrás del punto debe de haber letras, números o el guion bajo
  - *_._@yahoo.es* →válido

Práctica 4.10: Primitiva

- Crea una página PHP que muestre 100 combinaciones aleatorias de lotería primitiva; cada una de las cuales estará formada por seis números.
- Se debe de conseguir que en cada combinación no se repite ningún número.

Práctica 4.11: Validación avanzada al estilo JavaScript

- Crea un formulario que sirva para pedir los datos para un soporte técnico. El formulario debe de pedir y validar estos datos:
  - **Casilla de verificación** o botón de radio que permita elegir si la nacionalidad es española o no.
  - **Edad.** Tiene que ser de 1 a 150.
  - **Número de Identificación**. Basado en las reglas del NIF (que incluye los números de DNI, NIE y NIF especiales) que están disponibles en la URL: http://es.wikipedia.org/wiki/N%C3%BAmero_de_identificaci%C3%B3n_fiscal
  - **Nombre y apellidos**. Al menos debe de tener dos letras tanto el nombre como cada apellido. No se admiten números en el nombre ni en los apellidos.
  - **Correo electrónico**. Validaremos que tenga un símbolo de @ y además que pertenezca al dominio hotmail.com o bien gmail.com.
  - **Descripción** de la avería. Obligatoriamente hay que escribirla.
  - **Número de Modelo**. Consta de tres letras (ASCII) mayúsculas seguida de 4 números.
  - **Número de serie**. Empieza por las letras S o Z:
    - Si empieza por S. Entonces le siguen tres números y luego dos letras (ASCII) mayúsculas.
    - Si empieza por Z. Entonces le siguen 4 números, el último número solo puede ser 3, 5 o 7. Al final se indica una letra que puede ser B,C o D. Si la letra es la D le seguirá otro número del 1 al 3. Después (independientemente de si es B,C o D) hay dos letras (ASCII) más.
- Al validar si hay errores se vuelve a la página del formulario marcando de color rojo los elementos no válidos. Se debe conseguir que los datos introducidos no haya que volver a añadirles.
- Si los datos son correctos, simplemente se indica este hecho.

4.10 RESUMEN DE LA UNIDAD

- Las funciones son uno de los elementos que aporta PHP para escribir código modular que permita su reutilización.
- En PHP se pueden crear funciones con paso de variables por valor y por referencia, funciones recursivas, variables globales, variables estáticas, parámetros con valores por defecto, etc. En definitiva, todas las posibilidades de funciones que nos puede aportar un lenguaje avanzado.
- Las funciones se pueden almacenar en archivos que luego podemos incorporar con ayuda de la instrucción **include**.
- PHP dispone de arrays clásicos escalares y de arrays asociativos que permite crear estructuras de pares clave/valor. Se admite todo tipo de datos en los arrays.
- Disponemos de una estructura de control especial llamada **foreach** que facilita el recorrido de cada elemento de un array.
- PHP admite el uso de arrays multidimensionales.
- Además, el lenguaje proporciona una librería de funciones muy extensa para facilitar y potenciar el trabajo con arrays.
- Los Strings son las estructuras de datos que proporciona PHP para almacenar texto. Hay operadores especiales para trabajar con strings y, además, disponemos de una librería que facilita enormemente el trabajo con strings.
- PHP dispone de funciones para cifrar texto. En la versión 5.5 han mejorado notablemente sus prestaciones gracias al uso del algoritmo de cifrado **BlowFish**. El uso de estas funciones es obligatorio para tratar con textos en los que es imprescindible su encriptación.
- PHP dispone de funciones especiales para crear y operar con datos de tipo fecha. Además, aporta funciones que permiten personalizar la forma de mostrar fechas (y números).

4.11 TEST DE REPASO

¿Qué es una variable estática?

a) Una variable cuyo valor es siempre el mismo
b) Una variable cuyo valor persiste entre llamadas a la función en la que se creó
c) Una variable dentro de una función, cuyo valor está asociado al valor de una variable creada fuera de la función
d) Un parámetro al que se le asigna un valor por defecto

¿Qué tipo de función es la que se define con este código?

```
function f($n){
    if($n<=2) return $n;
    else return f($n-2)+f($n-1);
}
```

a) Estática
b) Referencial
c) Calcular
d) Recursiva

Usando el código de la pregunta anterior, si invocamos a esa función con el código `echo f(5)` ¿Qué sale por pantalla?

a) 15
b) 121
c) 5
d) 8
e) 12

En PHP los arrays...

a) Solo pueden ser unidimensionales
b) Solo pueden ser escalares
c) Son colecciones de datos del mismo tipo
d) Todas las respuestas anteriores son falsas

Observa este código:

```
$nota[0][0]=9;
$nota[0][1]=7;
$nota[1][0]=9;
```

¿Qué saca por pantalla el código `echo count($nota)`?

a) Sale un error al ser un array bidimensional
b) 1
c) 2
d) 3

¿Qué diferencia hay entre las funciones de ordenación de arrays sort y asort?

a) La primera ordena en ascendente y la segunda en descendente
b) Ninguna, son funciones sinónimas
c) La primera ordena los valores y la segunda los índices
d) La primera elimina los índices originales y la segunda los respeta

Si la variable $testo es un string, como podemos mostrar por pantalla su quinto carácter?

a) `$texto(5)`
b) `$texto(4)`
c) `$texto[5]`
d) `$texto(5)`

8. **¿Qué función permite obtener el tamaño de un string?**

a) `length`
b) `strlen`
c) `size`
d) `width`

9. **¿Cuáles de estos textos cumplen la siguiente expresión regular?**

```
/^[K-N]?[0-9]+[ASD]$/
```

a) kA
b) 78655144565SD
c) 78655144565S
d) L78655144565S
e) K78655144565S
f) K78655144565ASD
g) KS

UNIDAD 5

INTERCAMBIO DE INFORMACIÓN ENTRE PÁGINAS WEB CON PHP

OBJETIVOS

- Descubrir el problema de http como protocolo *sin estado*
- Asimilar los mecanismos de PHP para intercambios datos entre diferentes páginas
- Identificar los problemas de las cookies y las sesiones
- Enumerar los riesgos del uso de cookies y sesiones
- Asimilar el funcionamiento de las cookies u¡y sesiones
- Almacenar datos en cookies y archivos de sesión
- Leer datos de cookies y sesiones
- Borrar cookies y datos de sesión

CONTENIDOS

5.1 LIMITACIONES DEL PROTOCOLO HTTP

El protocolo que se utiliza para navegar por la web es **http**. Con lo cual la programación de aplicaciones web depende de este protocolo. http es un protocolo de petición (***request***) y respuesta (***response***); básicamente un cliente hace una petición de recurso y un servidor http responde a esa petición.

El problema está en que http es un **protocolo sin estado** (o sin memoria), cada transacción es independiente de la anterior.

Al principio, los desarrolladores de aplicaciones web no estaban preocupados por esta circunstancia ya que, esencialmente, los servidores web servían páginas estáticas que contenían información simple. Además esa información era la misma para cada usuario. Sin embargo en la actualidad, debido a que desde http se sirven todo tipo de servicios, necesitamos otorgar ***memoria*** a nuestros desarrollos.

El caso más evidente para entender por qué necesitamos memoria está en una aplicación web que nos pida usuario y contraseña para después mostrarnos nuestra página personal dentro de ese servicio. Esta portada puede ser nuestro buzón de mensajes, nuestra pared en una red social, nuestro carrito de la compra, etc. La cuestión es que esa zona personal debe recordar quiénes somos y para ello lo que hacemos en la pantalla de conexión (***login***) debe de generar un ***estado***.

Luego tenemos un problema. Si el protocolo http no tiene estado y necesitamos ese estado, la única manera es enviar la información necesaria para generar el estado dentro del propio paquete http. Como veremos a continuación hay numerosas técnicas para hacerlo, el problema estriba en que alguien pueda capturar el envío del paquete y hacerse con esa información confidencial.

5.2 FORMAS DE GENERAR UN ESTADO O SESIÓN

5.2.1 USO DE LA DIRECCIÓN IP

En PHP disponemos del array **$_SERVER** que proporciona mucha información interesante. Así, en ese array, ***$_SERVER["REMOTE_ADDR"]*** nos permite obtener la IP del cliente. En el caso, por ejemplo, de acceder a una zona de seguridad puede ser un método de autentificación. Pero no es un método seguro, ya que el usuario/a puede acceder desde servidores proxy o utilizando otras tecnologías que no nos permitan identificar con seguridad su dirección IP.

5.2.2 PASO DE PARÁMETROS MEDIANTE CADENA DE CONSULTA

Toda URL puede tener una cadena de consulta. Por ejemplo, si escribimos en nuestro navegador la url: https://www.google.es/#q=PHP abriremos la página de Google en España y auto-

máticamente se nos mostrarán páginas que, según Google, se relacionan con PHP. Usado con habilidad es un método interesante para pasar datos de una página a otra.

En realidad, esta forma de trabajar se relaciona con el método GET de los formularios. Así podemos enviar parámetros usando direcciones URL como por ejemplo la siguiente: http://miservidor.com/prueba.php?id=2309&precio=24 de modo que a la página ***prueba.php*** le pasamos los parámetros ***id*** (con valor ***2309***) y precio (con valor ***24***). Es un método muy utilizado, pero tiene la desventaja de que se lee perfectamente en la barra de direcciones lo que estamos enviando, por lo que no es válido para enviar información confidencial (a no ser que la cifremos).

ACTIVIDAD 5.1:

- Crea una página capaz de recoger un nombre mediante un parámetro GET de cadena de consulta. Así si la página se llama **consulta.php,** la podremos invocar con el texto: ***consulta.php?nombre=jorge.***
- Consigue que la página escriba ***Hola*** seguida del nombre que le mandes en la cadena de consulta.

5.2.3 PASO DE PARÁMETROS MEDIANTE MÉTODO POST

Los formularios también permiten pasar información entre páginas. Si ese paso se hace por GET, tenemos la desventaja de que los datos se ven en la barra de dirección, al igual que hemos visto en el apartado anterior. Si usáramos POST, en lugar de GET, para pasar información entre páginas, los datos no se ven en la barra de dirección, sino que viajan en la cabecera del paquete http (véase Figura 3.6 en la página 150).

Muchas veces se usan controles de formulario ocultos para pasar información adicional sin que el usuario la perciba. El truco para ello es utilizar controles de formulario con el código HTML: `<input type="hidden"`, de modo que el usuario no los vea y cuando los datos visibles se envíen, también se estarán enviando los datos ocultos.

ACTIVIDAD 5.2:

- Prueba a crear un formulario que tenga un control visible y un control oculto. En el control visible pide por ejemplo el nombre al usuario. En el oculto almacena el texto "***No me ves***".
- Haz una página que recoja estos datos vía POST y comprueba que recibes ambos valores mostrándolos por pantalla.

5.2.4 COOKIES

Se explica en detalle en esta misma unidad. La idea es que en el ordenador del usuario se almacena la información que necesitamos en forma de archivo de texto.

En realidad una cookie es un par nombre/valor (al igual que los parámetros GET o POST), de modo que a un determinado nombre o clave le asignamos un valor (que siempre es texto).

Lo malo es que el usuario puede bloquear las cookies o borrarlas y eso está fuera de nuestro control por lo que es una técnica que tiene también sus riesgos. Además hay técnicas para intentar obtener las cookies del usuario con la finalidad de conseguir información confidencial. Por este último tema, las cookies son motivo de controversia, de hecho, **estamos legalmente obligados a avisar al usuario de que estamos grabando cookies en su ordenador**.

5.2.5 SESIONES

En realidad todas las técnicas anteriores sirven para generar información de sesión. Entendiendo por sesión la actividad que tiene un usuario en su navegador desde que lo abre hasta que lo cierra.

Sin embargo cuando en PHP se habla de sesiones, se denomina así a la técnica, que se explica en detalle más adelante, por la que se almacenan datos de usuario, al estilo de las cookies, pero en el servidor. La técnica se basa en que al usuario se le asigna un **identificador de sesión,** por el cual relacionaremos su archivo de sesión en el servidor.

Ese número de sesión es el cuello de botella de la seguridad en esta técnica, obtenerlo por parte de terceros es una actividad habitual de aquellos que intentan un **robo de sesión (o *hijacking*)** sobre un usuario, para hacerse con su información confidencial.

5.2.6 BASES DE DATOS

El uso de esta técnica se explica en detalle en la siguiente unidad. Es el método habitual de almacenaje permanente de información. Se trata de que desde PHP accedemos a un sistema gestor de bases de datos encargado de almacenar la información que necesitamos.

Lo bueno es que toda la gestión y mantenimiento de los datos lo realiza un software especializado en ello, descargando a PHP de esa responsabilidad. Lo malo es que para datos de sesión, se convierte en una técnica difícil de mantener, por lo que en la práctica se combinan las sesiones y las bases de datos.

5.3 USO DE COOKIES DESDE PHP

5.3.1 FUNCIONAMIENTO DE LAS COOKIES

Son indudablemente la opción más habitual para almacenar información del usuario. Lo malo es lo ya comentado: el cliente de un sitio web puede prohibir o borrar las cookies almacenadas, con lo que no tenemos la seguridad de que esta técnica funcione en todos los clientes.

Las cookies son archivos de texto que almacenan información sobre el usuario y que se guardan en el propio ordenador del usuario. La información almacenada asocia un valor (de tipo string) a un nombre. Al igual que el paso por GET o POST, el uso de cookies es otra técnica de almacenaje de información mediante pares nombre/valor.

Figura 5.1: Esquema de funcionamiento de las cookies

Las cookies se pasan en la cabecera de la petición http por lo que su información viaja en el mismo paquete. En el momento que el servidor decide grabar los datos de la cookie, esta viaja con el paquete en todas las peticiones y respuestas de esa sesión.

La idea conceptual es la que refleja la Figura 5.1; en cuanto, desde una página PHP, grabamos datos mediante cookies, nuestras peticiones y respuestas llevan anexas los datos de las cookies que hemos almacenado y, por lo tanto, podremos utilizar en cualquier momento dichos datos.

Figura 5.2: Extracto de la cabecera http de un paquete que contiene cookies, visible a través de las herramientas de inspección de Google Chrome

5.3.2 ALMACENAMIENTO DE COOKIES DESDE PHP. SETCOOKIE

La función **setcookie** es la encargada de añadir (o borrar) una nueva cookie. Sintaxis:

```
setcookie(nombre [,valor [,expiración [,ruta [,dominio [,segura
                                                    [soloHttp]]]]] )
```

Los parámetros son:

- *nombre*. Es el nombre que le damos a la cookie. Después podremos consultarle para poder saber su valor.
- *valor*. Valor que le damos a la cookie. Si no indicamos valor alguno, entonces estaremos borrando la cookie.
- *expiración*. Por defecto toma el valor cero, que significa que la cookie caduca en cuanto se cierre el navegador con el que se creó. Otra posibilidad es indicar una fecha en formato **Unix**. Por ejemplo, `time()+600` indicaría que la cookie expira dentro de 10 minutos
- *dominio*. Si no se indica nada toma como dominio en el que se aplica la cookie el dominio actual, pero se puede indicar otro.
- *segura*. Por defecto vale **false**, si indicamos true; entonces indica que la cookie solo se almacena si estamos comunicándonos mediante un protocolo seguro.
- *soloHttp*. Disponible desde la versión 5.2 de PHP. Es un valor booleano que indica que la cookie solo está disponible mediante el protocolo **http** y no, por ejemplo, desde **JavaScript**. La idea es paliar, en cierta forma, problemas de seguridad al hacer que la cookie no se pueda recoger en un ataque de tipo *Cross Side Scripting* (*XSS*).

Ejemplos:

```
setcookie("visitas",1);
//graba la cookie visitas con valor 1 y que caduca con esta sesión
setcookie("usuario","andrei",time()+60*60*24)
//graba la cookie con nombre usuario y valor andrei que
//caducará al día siguiente
```

5.3.3 ACCEDER A LOS DATOS DE LAS COOKIES

El array global **$_COOKIE** permite acceder a las cookies almacenadas. Para acceder al valor de una cookie concreta, basta indicar el nombre de la cookie como índice de este array. Por ejemplo, *$_COOKIE["visitas"]* devolvería el valor de la cookie con nombre ***visitas*** (suponiendo que esté definida dicha cookie). Ejemplo:

```
if(isset($_COOKIE["visitas"])){
   echo "esta es tu visita número{$_COOKIE["visitas"]}";
}
```

5.3.3.1 BORRAR COOKIES

Cuando se vuelve a utilizar la función **setcookie** indicando el nombre de una cookie ya existente y un nuevo valor, el nuevo valor sobrescribe el anterior valor que tuviera dicha cookie. Bajo la misma idea, si usamos **setcookie** indicando un nombre de cookie existente y el valor cero o **false**, entonces, se eliminará la cookie con ese nombre.

Ejemplo:

```
setcookie("visitas",false);//elimina la cookie de nombre visitas
```

Podemos también eliminar cookies más antiguas indicando una fecha de caducidad anterior a la actual, por ejemplo:

```
setcookie("visitas",false,time()-60*60*24*7);
/* elimina la cookie de nombre visitas con fecha de caducidad de
hace al menos una semana */
```

5.3.3.2 ALMACENAR DATOS BINARIOS EN COOKIES

En las cookies no se pueden almacenar datos binarios. Las cookies solo admiten valores de tipo string. Es decir, no es válido el código:

```
setcookie("notas",array(9,7,6,5));  //inválido: datos binarios
```

Por ello, todos los programadores para almacenar datos binarios, como los arrays, en las cookies, crean mecanismos para convertirles en texto. En general al proceso de convertir datos binarios en formato de texto, se lo conoce bajo el término de **serializar**; traducción excesivamente literal del inglés ***serialize***.

Podemos idear nuestro propio mecanismo de *serialización*, pero en PHP hay una función que nos ayuda a automatizar este proceso. Se trata de la función **serialize.** A esta función se le pasa un dato binario y se convierte a un formato de texto equivalente. De ese modo, si queremos almacenar en una cookie llamada ***notas*** un array llamado ***$array1***, la instrucción sería:

```
setcookie("notas",serialize($array1));
```

El problema ahora es que cuando lo queramos leer, tenemos el problema contrario: es decir, convertir el texto en el binario correspondiente. Ese proceso es incluso más difícil; pero PHP lo facilita mediante la función contraria: **unserialize**. Esta función realiza el proceso inverso a ***serialize***. Para extraer el array del ejemplo anterior escribiríamos este código:

```
$array1=unserialize($_COOKIE["notas"]));
```

5.4 USO DE SESIONES EN PHP

5.4.1 VENTAJAS Y DESVENTAJAS

PHP permite automatizar la gestión de sesiones. Las sesiones, comparadas con las cookies ofrecen estas ventajas:

- Pueden funcionar aunque el usuario/a desactive la posibilidad de cookies. Aunque hay que tener en cuenta que para ello debemos pasar el identificador de sesión mediante GET; es decir, el identificador de la sesión será visible en la barra de direcciones del

navegador. Además, por defecto PHP usa cookies para pasar el identificador de sesión, si no lo queremos pasar por cookies, debemos modificar varios parámetros del archivo de configuración de PHP.

- Almacenan más información que una cookie y, además, son capaces de almacenar datos binarios.
- Son más seguras puesto que los datos que se almacenan de la sesión lo hacen en el servidor y no en el cliente. En principio, la seguridad de los servidores es mayor que la de los ordenadores de los usuarios

Desventajas:

- Son susceptibles al ataque conocido como **secuestro de sesión** (***session hijacking***), mediante el cual un usuario se puede hacer con el identificador de sesión de otro usuario y hacerse pasar por él. El paso de ese identificador es el cuello de botella del sistema de sesiones.
- El identificador de sesión se almacena normalmente mediante cookies, por lo que si no nos damos cuenta, podríamos tener la misma dependencia con el usuario que en el caso de las cookies. Si el usuario las desactiva, no podremos utilizar sesiones.

5.4.2 FUNCIONAMIENTO

El manejo de sesiones se basa en asignar un identificador a cada sesión. PHP dispone de la posibilidad de calcular dicho número de sesión para que sea único en cada sesión de usuario. Tras lo cual podemos identificar de formas unívoca a cada sesión. Ese identificador es el que se relacionará con los datos del usuario.

Los pasos son los siguientes:

[1] Cuando deseamos generar una nueva sesión, se invoca a la función **session_start()**. Esa función genera un nuevo identificador de sesión (SESSION_ID).

[2] Ese identificador de sesión se almacena en una cookie o bien se envía como parámetro GET dentro de cada petición del cliente. Que se utilice uno u otro método dependerá de la configuración de PHP.

[3] Asociado a ese identificador de sesión, se crea un archivo en el servidor en el que se almacenarán físicamente los datos de la sesión.

[4] Si queremos grabar un dato de sesión, debemos utilizar el array **$_SESSION** al que le indicaremos el nombre y el valor (que puede ser binario) a almacenar.

[5] En una página en la que deseemos recoger datos almacenados de la sesión actual, debemos invocar primero a **session_start()** y luego, mediante el array $_SESSION, podremos leer, modificar o añadir datos de la sesión.

[6] La sesión finaliza automáticamente cuando el usuario cierra el navegador.

Veamos el proceso con un ejemplo. Este código crea una página que inicia una sesión y graba en ella un parámetro llamado ***nombre*** asociado al valor ***Jorge***:

```
<?php
    session_start();
    $_SESSION["nombre"]="Jorge";
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <p>Se ha iniciado la sesión</p>
    <a href="pagina2.php">Ir a la página 2</a>
</body>
</html>
```

En este código se crea una nueva sesión y se asigna en ella el nuevo valor ***Jorge*** asociado al identificador ***nombre***. Además, se muestra el texto ***Se ha iniciado la sesión*** y un enlace a la página ***pagina2.php***.

Supongamos que hacemos clic en el enlace y que el código de ***pagina2.php*** es este:

```
<?php
    session_start();
?>
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
<?= "Hola ".$_SESSION["nombre"] ?>
</body>
</html>
```

Se mostraría en pantalla el texto ***Hola Jorge***, resultado de acceder correctamente a los datos de la sesión.

Lo fundamental a recordar es el uso de la función **session_start** cada vez que utilicemos sesiones en PHP. Esta función gestiona el paso del identificador de sesión, además, retorna verdadero si la sesión se pudo realizar. Otro detalle es conseguir que esa instrucción sea la primera en el

código PHP, la razón es que es una instrucción que maneja cabeceras del protocolo http, y por ello, no tiene sentido colocarla en el cuerpo con el riesgo de que no se ejecute debidamente.

En realidad las sesiones son más sencillas de manejar que las cookies. Pero queda pendiente la cuestión sobre cómo se pasa el identificador de sesión de petición en petición. Las dos posibilidades son:

- **Mediante una cookie**. En este caso en cuanto se genera la sesión mediante la función **session_start**, se crea una cookie cuyo nombre es el nombre de la sesión y cuyo valor es el identificador de la sesión. En PHP es la opción por defecto. Como se ha comentado en el apartado dedicado a las cookies, éstas se envían en la cabecera del paquete http, por lo que se pueden interceptar en el caso de que un tercero tenga acceso a los datos que estamos enviando mediante un analizador de paquetes o ***sniffer***. La solución más inmediata es que la comunicación esté cifrada de alguna manera, por ejemplo, con ayuda del protocolo https.

  Utilizando el código anterior, el proceso de funcionamiento de la sesión sería el siguiente:

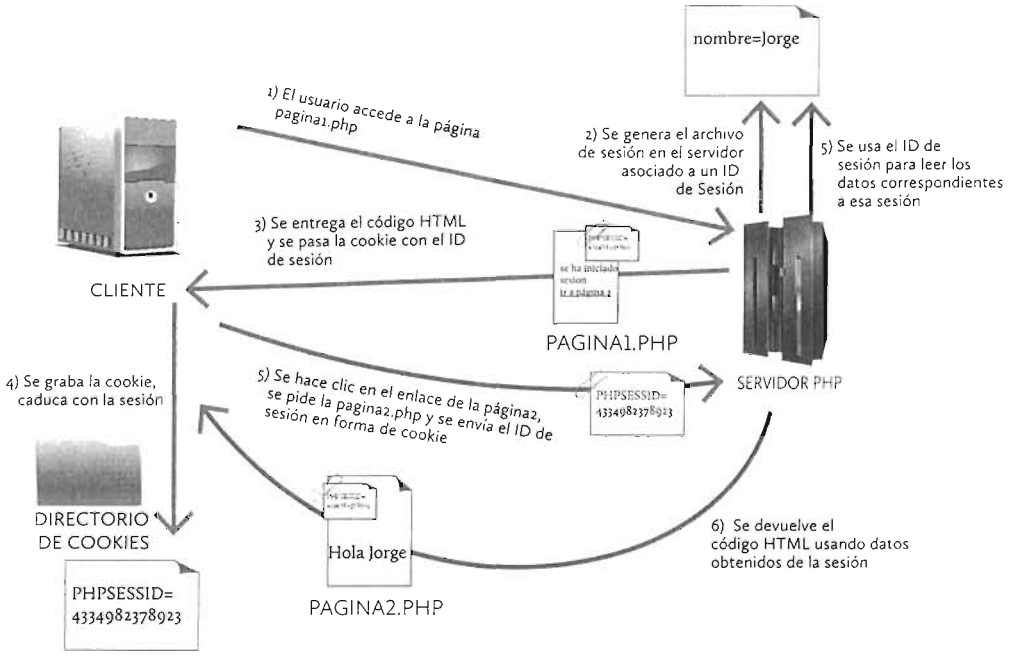


Figura 5.3: Funcionamiento de una sesión gestionada a través de una cookie

Para que esta opción sea la que funcione (si no fuera así) hay que modificar el archivo de configuración php.ini y colocar estos valores:

```
session.use_cookies 1
session.use_trans_sid 0
```

- **Mediante GET**. En este caso se pasa el identificador de la sesión acompañando a cada petición añadiendo una cadena de consulta GET a la URL. Es decir, que si nuestra sesión

tiene el nombre habitual de ***PHPSESSID***, con el identificador de sesión ***8FR45237A89*** y estamos en la URL ***http://prueba.com/pog1.php,*** la sesión provoca una URL parecida a esta:

```
http://prueba.com/pog1.php?PHPSESSID=8FR45237A89
```

Para que sea esta la opción que está funcionando, deberemos modificar el archivo php.ini y hacer estos cambios:

```
session.use_cookies 0
session.use_trans_id 1
```

El esquema de funcionamiento de sesiones con identificador en la URL sería el siguiente:

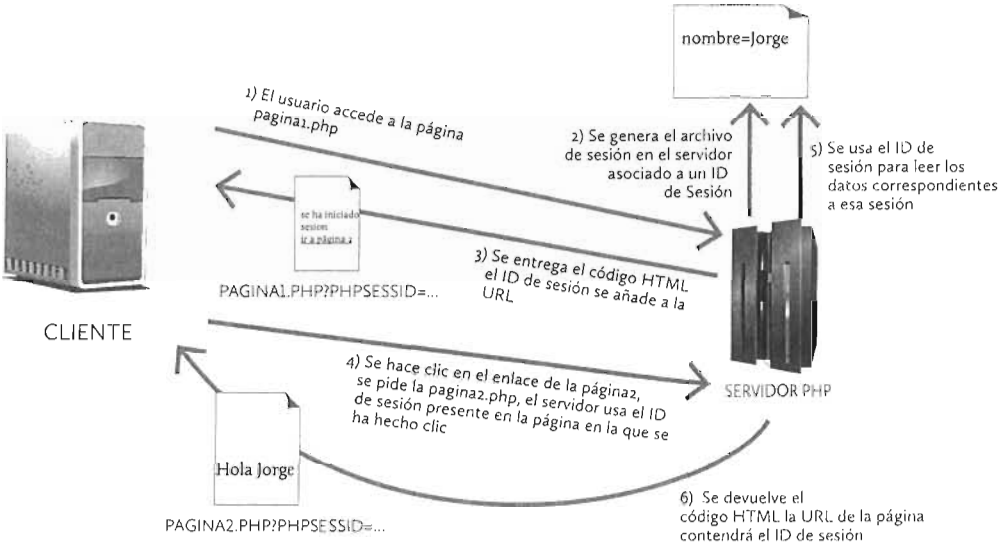


Figura 5.4: Funcionamiento de una sesión gestionada a través de una cookie

Normalmente, la sesión caduca en cuanto el usuario abandona el navegador. Se puede hacer (sobre todo si el identificador es una cookie) que dure más, pero no es lo recomendable. Las sesiones deben caducar cuando el usuario cierra la sesión, de otro modo hay más posibilidad de inseguridad. Es la forma habitual de usar sesiones.

Si necesitamos que los datos de la sesión perduren, lo lógico es usar cookies o bases de datos que son las técnicas de almacenamiento de información que están pensadas para esa finalidad.

5.4.3 INICIO DE SESIÓN

Una sesión comienza en la primera página PHP que invoque a la función **session_start**. Esta función no tiene argumentos, simplemente inicia la sesión si no ha sido iniciada, o bien permite el acceso a los datos de la sesión si ésta había sido iniciada anteriormente. Iniciar la sesión implica:

[1] Crear un identificador aleatorio de sesión; salvo que a través de la función **session_id** (se explica más adelante) seamos nosotros los que generemos un identificador.

[2] Crear en el servidor un archivo asociado a ese identificador, en el que se almacenarán los datos de la sesión.

[3] Crear la variable superglobal **$_SESSION** que es un array (al estilo de **$_GET**, **$_POST** o **$_COOKIE**) en el que se almacenarán los datos de la sesión.

5.4.4 USO DE LA SESIÓN PREVIAMENTE INICIADA

Si la sesión ya había sido iniciada cuando se invocó a **sessión_start**, entonces, no se crea una nueva sesión sino que:

- Se sigue utilizando la sesión anterior
- Se permite el uso del array **$_SESSION** que contendrá los datos de la sesión, los cuales pueden ser modificados, leídos e incluso podremos añadir más datos.

RECOMENDACIÓN

- En resumen, siempre que deseemos trabajar con sesiones debemos utilizar la función **session_start**.
- Como esta función manipula cabeceras http, debemos intentar que sea el primer código de la página PHP, antes incluso que la etiqueta HTML de cabecera, **!DOCTYPE**.

5.4.5 OBTENER DATOS DE LA SESIÓN

Dos funciones permiten obtener (y cambiar) los datos de la sesión:

- **session_id**. Si la invocamos sin parámetros, devuelve el identificador de sesión actual. Admite que le pasemos un identificador de sesión nosotros, con lo que ese se convierte en el identificador de la sesión actual. Poner números de sesión propios es peligroso, en cuanto a que si no tenemos un buen algoritmo de cálculo del identificador podríamos tener dos sesiones con el mismo identificador y; por lo tanto, indistinguibles y, lo que es peor, podríamos provocar que una sesión vea los datos de otra.
- **session_name**. Si la invocamos sin parámetros, devuelve el nombre de sesión actual (por defecto la sesión se llama **PHPSESSID**). Admite que le pasemos un nombre de sesión nosotros. Esto es útil para que no sea tan claro que estamos usando sesiones PHP ya que el nombre de sesión por defecto: **PHPSESSID**, es conocido por todos los programadores PHP, lo que facilita el robo de la sesión.

  Otra opción para cambiar el nombre de los identificadores de sesión es utilizar la directiva **session.name** dentro del archivo de configuración de PHP (normalmente **php.ini**). A esta directiva se le puede asignar otro nombre para las sesiones.

5.4.6 USAR VARIABLES DE SESIÓN

Una variable de sesión permite asociar una clave a un valor de la sesión actual.

Para almacenar un dato de sesión basta usar esta sintaxis:

```
$_SESSION["clave"]=valor
```

Para recoger un valor almacenado se utiliza el mismo array indicando la clave o nombre del valor que deseamos recoger. Por ejemplo:

```
$variable=$_SESSION["clave"]
```

Como hemos comentado, es posible incluso almacenar datos binarios dentro de las variables de sesión, lo que convierte a las sesiones un método de trabajo más versátil que las cookies.

5.4.7 BORRAR DATOS DE LA SESIÓN

Si deseamos eliminar un dato de la sesión basta con usar la función PHP de eliminación de variables **unset**. De modo que si deseamos eliminar los datos de la sesión asociados al nombre o clave ***visitas***, haremos lo siguiente:

```
unset($_SESSION["visitas"])
```

Para eliminar todos los datos, hay que hacer esta secuencia:

```
unset($_SESSION);
$_SESSION=array();//imprescindible, sino la sesión queda
inutilizable
```

5.4.8 ELIMINAR LA SESIÓN ENTERA

Eliminar la sesión de forma absoluta, consiste en:

```
session_start(); //si no la hubiéramos invocado ya
unset($_SESSION); //elimina el array, sino se queda gastando memoria
session_destroy();//función de eliminación en sí de la sesión
```

5.5 PRÁCTICAS RESUELTAS

Práctica 5.1: Grabado de preferencias de usuario con cookies

- Crea una página web (***practica1-index.php***) en la que se pregunten datos del usuario, concretamente:
  - Nombre y apellidos
  - Color deseado para el fondo
  - Color deseado para la letra
  - Elegir una de entre tres tipos de letra (pueden ser, por ejemplo, tres tipos accesibles en la página de Google fonts)
- Tras aceptar los datos, una segunda página (***practica1-saludo.php***) saluda al usuario, por ejemplo, con el mensaje ***Hola Ana Fernández***, utilizando el tipo de letra y colores elegidos.
- La siguiente vez que accedamos a la página inicial (***practica1-index.php***), ya no nos preguntará las preferencias porque se dará cuenta que tiene los datos en cookies y nos llevará a la segunda página.
- Si intentamos acceder directamente a la página del saludo y no hemos grabado las preferencias, esta página nos llevará a la página con el formulario.
- En la página del saludo, un enlace nos permitirá borrar las preferencias.

SOLUCIÓN: PRÁCTICA 5.1

Para facilitar el trabajo vamos a crear un archivo llamado ***practica1-comprobar.php***. En este archivo solo vamos a escribir el código de dos funciones: una función llamada ***hayCookie*** que va a devolver verdadero si existen las cookies con las preferencias de usuario y otra casi igual llamada ***hayGet*** que devuelve verdadero si están llegando parámetros vía GET con las preferencias de usuario, correctamente. El código de ese archivo es:

```
<?php
function hayCookie(){
    return isset($_COOKIE["nombre"]) && isset($_COOKIE["apellidos"])
        && isset($_COOKIE["fondo"]) && isset($_COOKIE["frente"])
        && isset($_COOKIE["letra"]);
}
function hayGet(){
    return isset($_GET["nombre"]) && isset($_GET["apellidos"])
        && isset($_GET["fondo"]) && isset($_GET["frente"])
        && isset($_GET["letra"]);
}
?>
```

Código de la página inicial (formulario), ***practica1-index.php***. Simplemente se comprueba si las preferencias no están ya guardadas. Si lo están, directamente se redirige la página a la del saludo, si no se mostrará el formulario.

```
<?php
/********practica1.index.php*******************/
include "practica1-comprobar.php"; //carga las funciones de comprobación
if(hayCookie()){
  //si tenemos preferencias ya guardadas, directamente vamos al saludo
    header("location:practica1-saludo.php");
}
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Formulario de recogida de preferencias</title>
</head>
<body>
<form action="practica1-saludo.php">
    <label for="nombre">Nombre</label>
    <input type="text" name="nombre" id="nombre"/><br/>
    <label for="apellidos">Apellidos</label>
    <input type="text" name="apellidos" id="apellidos"/><br/>
    <label for="fondo">Color de fondo</label>
    <input type="color" name="fondo" id="fondo"/><br/>
    <label for="frente">Color de letra</label>
    <input type="color" name="frente" id="frente"/><br/>
    <label for="letra">Tipo de letra</label>
    <select name="letra" id="letra">
        <option value="'Shadows Into Light', cursive">
            Shadows Into Light
        </option>
        <option value="'Slabo 27px', serif">Slabo 27px</option>
        <option value="'Roboto', sans-serif">Roboto</option>
    </select><br/>
    <button>Enviar</button>
</form>
</body>
</html>
```

A continuación se expone el código de la página ***practica1-saludo.php***. Básicamente comprueba si tenemos cookies con los valores de preferencia de usuario y, si no es así, comprueba si están llegando las preferencias vía GET (lo cual significaría que nos los está enviando el formulario anterior). Si no llegan datos correctos, entonces cargamos de nuevo la página del formulario.

```
<?php
/**********************practica1-saludo.php*********************/
include "practica1-comprobar.php";//carga las funciones de comprobación
$hayPreferencias=
                true;//nos dice si tenemos preferencias de usuario
$array=null; //accede al array, sea GET o POST con las preferencias
if(hayCookie()==false){
    if(hayGet()==false){
        $hayPreferencias=false;
    }
    else{
        $array=$_GET;
    }
}
else{
    $array=$_COOKIE;
}
if($hayPreferencias==false)
    //si no tenemos preferencias, volvemos al formulario
    header("location:practica1-index.php");
else{
    //Recorremos el array con los datos de usuario
    //y creamos una variable con cada nombre de preferencia
    foreach($array as $indice=>$valor){
        $$indice=$valor;
    }
    //grabar cookies con las preferencias de usuario
    setcookie("nombre",$nombre,time()*60*60*24*30);
    setcookie("apellidos",$apellidos,time()*60*60*24*30);
    setcookie("fondo",$fondo,time()*60*60*24*30);
    setcookie("frente",$frente,time()*60*60*24*30);
    setcookie("letra",$letra,time()*60*60*24*30);
}
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Saludo</title>
<!--Carga de las letras desde Google fonts -->
<link
href='http://fonts.googleapis.com/css?family=Shadows+Into+Light|Slabo+27px|Roboto'
rel='stylesheet' type='text/css'>
```

```
<style>
        body{
            background-color:<?=fondo?>;
            color:<?=$frente?>;
            font-family:<?=$letra?>;
        }
        a{
            background-color:white;
        }
    </style>
</head>
<body>
<h1>Hola <?="$nombre $apellidos"?></h1>

<p><a href="practica1-borrar.php">Volver a cambiar las preferencias
</a></p>
</body>
</html>
```

Finalmente, el código del archivo ***practica1-borrar.php*** es el encargado de borrar las cookies (y por lo tanto las preferencias de usuario) y pedir las preferencias de usuario en el formulario. Este código se invoca cuando se hace clic en el enlace ***Volver a cambiar las preferencias*** en la página anterior.

```
<?php
//borra las cookies y nos devuelve al formulario
setcookie("nombre",false);
setcookie("apellidos",false);
setcookie("fondo",false);
setcookie("frente",false);
setcookie("letra",false);
header("location:practica1-index.php");
?>
```

Práctica 5.2: Grabado de preferencias mediante sesiones

- Se trata de hacer la misma página que en la práctica anterior, pero ahora utilizando sesiones.

SOLUCIÓN: PRÁCTICA 5.2

Básicamente el funcionamiento es el mismo. La única precaución al utilizar sesiones es invocar a **session_start** antes de utilizar el array $_SESSION.

Código del archivo *practica2-comprobar.php*:

```
<?php
function hayCookie(){
    return isset($_COOKIE["nombre"]) && isset($_COOKIE["apellidos"])
        && isset($_COOKIE["fondo"]) && isset($_COOKIE["frente"])
        && isset($_COOKIE["letra"]);
}
function hayGet(){
    return isset($_GET["nombre"]) && isset($_GET["apellidos"])
        && isset($_GET["fondo"]) && isset($_GET["frente"])
        && isset($_GET["letra"]);
}
?>
```

Código de la página del formulario, *practica2-index.php*:

```
<?php
include "practica2-comprobar.php";
if(haySesion()){
//si tenemos preferencias ya guardadas, directamente vamos al saludo
    header("location:practica1-saludo.php");
}
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Formulario de recogida de preferencias</title>
</head>
<body>
<form action="practica2-saludo.php">
    <label for="nombre">Nombre</label>
    <input type="text" name="nombre" id="nombre"/><br/>
    <label for="apellidos">Apellidos</label>
    <input type="text" name="apellidos" id="apellidos"/><br/>
    <label for="fondo">Color de fondo</label>
    <input type="color" name="fondo" id="fondo" value="#FFFFFF"/><br/>
    <label for="frente">Color de letra</label>
    <input type="color" name="frente" id="frente"/><br/>
    <label for="letra">Tipo de letra</label>
    <select name="letra" id="letra">
        <option value="'Shadows Into Light', cursive">
            Shadows Into Light
        </option>
        <option value="'Slabo 27px', serif">Slabo 27px</option>
```

```
        <option value="'Roboto', sans-serif">Roboto</option>
    </select><br/>
    <button>Enviar</button>
</form>
</body>
</html>
```

Código de la página de saludo, **practica2-saludo.php**:

```
<?php
include "practica1-comprobar.php";
//nos dice si tenemos preferencias de usuario
$hayPreferencias=true;
//accede al array, sea GET o POST con las preferencias
$array=null;
if(hayCookie()==false){
    if(hayGet()==false){
        $hayPreferencias=false;
    }
    else{
        $array=$_GET;
    }
}
else{
    $array=$_COOKIE;
}
if($hayPreferencias==false)
    //si no tenemos preferencias, volvemos al formulario
    header("location:practica1-index.php");
else{
    //Recorremos el array con los datos de usuario
    //y creamos una variable con cada nombre de preferencia
    foreach($array as $indice=>$valor){
        $$indice=$valor;
    }
    //grabar cookies con las preferencias de usuario
    setcookie("nombre",$nombre,time()*60*60*24*30);
    setcookie("apellidos",$apellidos,time()*60*60*24*30);
    setcookie("fondo",$fondo,time()*60*60*24*30);
    setcookie("frente",$frente,time()*60*60*24*30);
    setcookie("letra",$letra,time()*60*60*24*30);
}
?>
```

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Saludo</title>
<!--Carga de las letras desde Google fonts -->
<link
href='http://fonts.googleapis.com/css?family=Shadows+Into+Light|Slabo+27px|Roboto'
    <style>
        body{
            background-color:<?=$fondo?>;
            color:<?=$frente?>;
            font-family:<?=$letra?>;
        }
        a{
            background-color:white;
        }
    </style>
</head>
<body>
<h1>Hola <?="$nombre $apellidos"?></h1>
<p><a href="practica1-borrar.php">Volver a cambiar las preferencias
</a></p>
</body>
</html>
```

5.6 PRÁCTICAS PROPUESTAS

Práctica 5.3: Zona restringida

- Crea una página PHP llamada ***practica3-restringida.php***, bastará con que simplemente muestre el mensaje ***Zona restringida.***
- Dicha página no se podrá ver por parte de ningún usuario que no se haya autentificado en la página ***practica3-autentificacion.php***, la cual pedirá una contraseña.
- Cuando entramos en ***practica3-autentificacion.php*** y escribimos la contraseña correcta, se nos enviará automáticamente a la página ***practica3-restringida.php.***
- Si intentamos acceder directamente a la zona restringida sin haber pasado nunca por la autentificación, no se nos permitirá entrar y se nos enviará a ***practica3-autentificacion.php*** para que escribamos la contraseña.
- Tras habernos autentificado, durante la sesión podremos acceder directamente a la zona restringida y no se nos pedirá la contraseña.

- **Nota.** Se pueden utilizar cookies o sesiones, pero la contraseña se deberá almacenar cifrada para una mayor seguridad, especialmente si se utilizan cookies.

Práctica 5.4: Lista de tareas

- Muestra una página web en la que dispongamos inicialmente de un formulario en el que tenemos un cuadro de texto y un botón con el texto ***Añadir tarea***.
- Ese formulario nos permitirá escribir el nombre de una tarea y al pulsar añadir, aparecerá la página mostrando una lista de tareas donde ya aparece la que hemos añadido.
- Podemos seguir añadiendo tareas y cada vez que lo hacemos, la lista va creciendo.
- Al lado de la lista de tareas añadidas, cada una de ellas tendrá un enlace que diga ***Quitar tarea***, dicho enlace nos permitirá quitar la tarea de la lista.
- En todo momento el usuario tiene que ver la misma página en la que la lista muestra las tareas que ha ido añadiendo y donde ya no aparecerán las que ha ido eliminando.

Práctica 5.5: Juego visual

- Una página web nos mostrará cinco palabras aleatorias. Solo nos mostrará esas palabras durante 10 segundos, después de los cuales se mostrará una segunda página.
- En esa segunda página aparecerán 5 cuadros de texto que nos preguntan por las 5 palabras que se nos han mostrado en la primera página.
- Tras escribir las 5 palabras, una tercera página nos permitirá saber cuántas hemos acertado y un enlace nos dará la posibilidad de volver a jugar de nuevo.
- **Nota**. Para que una página se redireccione a otra pasados 10 segundos, lo más cómodo es escribir este código JavaScript:

```
window.onload=function()
{
setInterval(function(){location="http://google.es"},1000);
}
```

Práctica 5.6: JavaScript y PHP para seleccionar un tono

- Esta práctica solo se puede realizar con ciertos conocimientos en JavaScript. Se trata de mostrar una página que divida la pantalla en 20 rectángulos iguales. Cada rectángulo mostrará un color aleatorio.
- Al hacer clic en uno de los rectángulos la página se recarga mostrando otros 20 colores y en una zona llamada colores elegidos se mostrará el color elegido.
- Cada vez que elegimos un color, éste se coloca en la barra de colores elegidos.

5.7 RESUMEN DE LA UNIDAD

- El protocolo http es un protocolo sin estado. Cada petición es independiente de la anterior.
- Sin embargo al crear aplicaciones web, se necesita en muchas ocasiones que lo realizado en una petición influya en las siguientes.
- Los mecanismos para conseguir grabar información de estado o sesión son:
  - Utilizar la dirección **IP** del cliente.
  - Enviar información usando la cadena de consulta de la URL (paso por **GET**).
  - Enviar información utilizando controles de formulario ocultos (paso por **POST**).
  - Almacenar información mediante **cookies**.
  - Almacenar información mediante archivos de **sesión** en el servidor.
  - Almacenar información en ficheros o **bases de datos**.
- Las cookies en PHP:
  - Se almacenan mediante el método **setcookie**.
  - Se recogen utilizando el array global **$_COOKIE**.
  - Permiten almacenar solo valores de tipo texto. Para almacenar otro tipo de valores utilizamos las funciones **serialize** y **unserialize**.
  - El método **setcookie** sirve también para borrar cookies
- Las sesiones en PHP:
  - Requieren el uso de un identificador de sesión que se puede grabar en una cookie o enviar como cadena de consulta en la URL.
  - Se generan o utilizan mediante el método **session_start**.
  - Se leen y editan variables de sesión mediante el array global **$_SESSION**.
  - Se borran utilizan **unset($_SESSION)** y el método **session_destroy**.
- El peligro de las cookies es que alguien acceda a sus datos, ya que se envían en el paquete http.
- El peligro de las sesiones es que alguien obtenga el identificador de sesión, ya que también se envía en el paquete http.

5.8 TEST DE REPASO

¿Cuáles de estas afirmaciones son ciertas?

- Cookies y sesiones son los únicos elementos para enviar datos de una página a otra
- Los datos de los controles ocultos de los formularios solo se pueden enviar mediante POST
- http es un protocolo que no permite almacenar el estado de la sesión
- Las cookies solo se pueden utilizar si la sesión no se ha cerrado

¿Cuál de estas afirmaciones es cierta?

- Las cookies se almacenan en el ordenador del cliente
- Las cookies se almacenan en el servidor
- Parte de los datos de las cookies se almacenan en el ordenador cliente y parte en el servidor
- Las cookies no requieren ser almacenadas

¿Cuál de estas afirmaciones es cierta?

- Tanto las cookies como las sesiones admiten valores binarios
- Las sesiones admiten solo strings, las cookies además admiten valores binarios
- Las sesiones admiten valores binarios, las cookies solo admiten strings
- Tanto las sesiones como las cookies solo permiten valores en forma de string

¿Cómo se pasa el identificador de una sesión?

- Mediante una cookie
- Por parámetros GET
- Por controles de formulario ocultos
- Puede ser tanto por una cookie como por GET

¿Dónde se almacenan los datos una sesión?

- En el ordenador del cliente
- En el servidor
- Parte en el ordenador del cliente y parte en el servidor
- No requieren ser almacenados

¿Qué función se debe invocar obligatoriamente para poder utilizar datos de sesiones?

a) `init_session`
b) `session_start`
c) `session_begin`
d) No se requiere invocar a ninguna función para utilizar datos de sesión

¿Qué función se debe invocar obligatoriamente para poder utilizar datos de cookies?

a) `init_cookies`
b) `cookies_start`
`cookies_begin`
No se requiere invocar a ninguna función para utilizar datos de cookies

Supongamos que deseamos borrar los datos de una cookie llamada *edad* ¿Qué instrucción lo haría?

```
unset($_COOKIE[“edad”])
cookie_destroy("edad");
cookie_destroy("edad",false);
setcookie(“edad”,false);
```

Supongamos que deseamos borrar los datos de un dato de sesión con nombre *edad* ¿Qué instrucción lo haría?

```
unset($_SESSION[“edad”])
session_destroy("edad");
session_destroy("edad",false);
$_SESSION[“edad”]=false;
```

¿Para qué sirve la función serialize?

- Convierte valores binarios en strings
- Convierte strings en valores binarios
- Convierte strings a formato Unicode
- Convierte strings en formato ASCII

UNIDAD 6

ACCESO A BASES DE DATOS MEDIANTE PHP

OBJETIVOS

- Analizar las ventajas que ofrece conectar con sistemas gestores de bases de datos en una aplicación web
- Determinar los elementos necesarios para utilizar bases de datos desde una aplicación web
- Capturar errores de bases de datos adecuadamente
- Ejecutar instrucciones SQL desde la aplicación web
- Transformar datos procedentes de la base de datos a una forma más humana y visual
- Crear estructuras relacionales apropiadas para almacenar la información de una aplicación web
- Crear aplicaciones web que requieran manipular bases de datos

CONTENIDOS

6.1 BASES DE DATOS

Nota: Se da por hecho en este tema que se tienen conocimientos al menos básicos sobre el lenguaje SQL y el funcionamiento de las bases de datos relacionales tales como: creación de tablas, uso de transacciones, consultas, modificación de datos, etc.

6.1.1 VENTAJAS DE LAS BASES DE DATOS

El uso de bases de datos es imprescindible en cualquier aplicación web de hoy en día. Algunas razones para utilizarlas son:

- **Proporcionan almacenamiento permanente.** Es decir, los datos que se almacenan, en principio, lo harán de forma permanente.
- **Añaden una capa más de seguridad.** Los datos están fuera del alcance directo del usuario, ya que no se almacenan en las peticiones http. Los datos se encuentran en un servidor de base de datos con el que se comunica el servidor web. Esta comunicación es totalmente opaca al usuario, el servidor PHP presenta los resultados de interaccionar con el servidor de bases de datos, pero no el código ni las claves para abrir la base de datos.

  Aunque hay técnicas para intentar acceder a la base de datos usando trucos, lo cierto es que las bases de datos son un software más preparado para la seguridad que si nosotros gestionáramos directamente la información.
- **Proporcionan herramientas avanzadas de gestión de datos y usuarios.** Los Sistemas Gestores de Bases de Datos son un software especializado en la gestión de datos, permite hacer consultas avanzadas, crear diferentes vistas y permitir su uso a ciertos usuarios. También aportan facilidad para hacer copias de seguridad de los datos, gestión de transacciones, validación avanzada de datos, etc.
- **Uso del lenguaje SQL.** Las bases de datos relacionales (todavía hoy en día son las mayoritarias) poseen el lenguaje estándar SQL para manipular los datos. Es un lenguaje muy conocido por todos los profesionales de la información y proporciona una poderosa sintaxis para gestionar la información.

  Hay que tener en cuenta que también hay bases de datos NoSQL, es decir, bases de datos no relacionales que utilizan otros lenguajes para manipular sus datos como es el caso de **MongoDB**, **Cassandra**, **CouchDB o HBase**. La mayoría de ellas también se pueden utilizar con PHP, aunque ciertamente no es lo habitual.

6.1.2 PROCESO DE ACCESO A UN SISTEMA DE BASES DE DATOS DESDE PHP

Para que podamos acceder a sistemas de bases de datos desde PHP, necesitamos instalar una interfaz software que comunique a ambos servidores. Cualquier lenguaje de servidor (aunque no

sea PHP) requiere lo mismo, un software (o incluso varios) que conecten al servidor web con el de la base de datos.

El proceso de conexión de un servidor PHP con un servidor de bases de datos suele conllevar estos pasos:

[1] Instalar un software de acceso al sistema gestor de bases de datos en el servidor web. No siempre este paso es obligatorio. Por ejemplo para acceder a servidores de bases de datos **Oracle** deberemos instalar el software conocido como **Oracle Instant Client**.

[2] Colocar la librería PHP que contiene el **API** (***Applications Programming Interface***, interfaz de programación de aplicaciones) de acceso al servidor de bases de datos, en el directorio de extensiones de PHP, que es donde se guardan las librerías. Un API no es más que el conjunto de funciones que permiten acceder y manipular las bases de datos del servidor correspondiente.

[3] Modificar el archivo **php.ini** para asegurar que se carga la librería que hemos instalado.

[4] Reiniciar el servidor Apache y comprobar que disponemos del API de acceso a la base de datos.

Realmente, cada base de datos supone su propio proceso de instalación, las más populares poseen documentación oficial en la dirección:

http://es1.php.net/manual/es/refs.database.vendors.php

En el caso de MySQL es muy fácil ya que dispone de una integración excelente con PHP. La instalación de MySQL se ha explicado en el Apartado 2.6 *"Instalación Y Configuración de mySQL"*, en la página 77.

6.1.3 PROCESO DE ACCESO A LAS BASES DE DATOS DESDE PHP

Suponiendo que tenemos correctamente instalado el acceso al servidor de bases de datos deseado desde el servidor de PHP, una aplicación que desee utilizar información de una base de datos necesita realizar los siguientes pasos:

[1] **Conectar con la base de datos**. Lo que suele implicar una función a la que se le pasa la llamada **cadena de conexión** con la base de datos. Esa cadena suele tener estos elementos:

- **Dirección IP o nombre del servidor de la base de datos**. En entornos de producción, el servidor de base de datos suele ser físicamente diferente del servidor PHP, por lo que necesitamos indicar cuál es.
- **Puerto de acceso**. La conexión requiere indicar con qué puerto debemos comunicar con la base de datos. Si no se indica, se tomará el puerto por defecto. MySQL utiliza por defecto el 3306.

- **Usuario y contraseña**. Evidentemente, si el sistema de bases de datos tiene un mínimo de seguridad, el acceso estará restringido a aquellas personas que tengan permiso. Para verificar este derecho, se requiere autentificar a dicha persona.
- **Nombre de la base de datos**. Esto difiere según la base de datos. En MySQL no es obligatorio para conectar, pero sí para acceder a los datos, ya que están organizados en bases de datos. Oracle, por ejemplo, tiene otro tipo de organización en la que cada usuario posee su propio esquema de datos.

[2] **Preparar la instrucción SQL que deseamos lanzar sobre la base de datos.** Esto supone que el servidor prepare los recursos necesarios para el acceso a sus datos. A veces, también se realizan tareas de seguridad, como por ejemplo evitar ataques por inyección de SQL.

[3] **Ejecutar la instrucción SQL.**

[4] **Recoger los resultados de la instrucción SQL y mostrarles al usuario.**

[5] A veces se requiere volver al paso 4 ya que los datos se van recuperando poco a poco.

[6] **Cerrar la instrucción.** No es obligatorio en todas las bases de datos, pero en las que lo permiten, se liberan los recursos que el sistema ha utilizado para ejecutar la instrucción.

[7] Podemos ejecutar más instrucciones, para lo cual volvemos al paso 2.

[8] **Cerrar la conexión** y liberar recursos.

6.2 GESTIÓN DE ERRORES

6.2.1 IMPORTANCIA DE LA GESTIÓN DE ERRORES

En los pasos de proceso de bases de datos comentados en el punto anterior, faltarían aquellos que permiten gestionar los errores.

Al utilizar bases de datos hay muchos posibles errores que pueden ocurrir. Ejemplos de errores son:

- Fallo al realizar la conexión: porque el servidor de bases de datos no está en funcionamiento, por fallo en la red, por indicar mal los datos de conexión, etc.
- Fallo en la ejecución de una instrucción SQL: por mala sintaxis, porque la instrucción incumple reglas de validación de datos, etc.
- Fallo al recoger la información de la instrucción SQL.
- Fallo al cerrar la conexión.
- Fallo por interrupción de la comunicación con la base de datos.

No gestionar estos errores significa depender de la gestión que el servidor PHP haga de ellos. Sin gestión y por defecto, el servidor muestra los errores como un mensaje Warning que aparece

en la página web. Esto es un error gravísimo, ya que esa información no ayuda en nada al usuario, el cual se asustará por un mensaje que no entiende en absoluto; además ese tipo de mensajes puede dar pistas a aquellas personas que intenten atacar nuestro sistema porque sabrán qué servidores estamos utilizando y, sabiendo sus vulnerabilidades, utilizar esa información para romper el sistema.

Todas las API de acceso a la base de datos poseen funciones con las que podemos capturar los errores y obrar de forma más apropiada con ellos. Es muy importante conocer esas funciones.

6.2.2 CONFIGURACIÓN DE REPORTE DE ERRORES

Dentro de la configuración de PHP (mediante el archivo **php.ini**) podemos especificar qué errores se mostrarán al usuario. La directiva **error_reporting** es la encargada de decidir qué errores se muestran y cuáles se ignoran, según su nivel de gravedad.

Inicialmente cuando estamos en proceso de depuración de nuestro código, nos interesa mostrar todo lo que ocurre, pero cuando pasamos a producción, solo lo más grave debería mostrarse. Por ello debemos calibrar bien la directiva **error_reporting** a nuestras necesidades. Sus posibles valores son:

- **E_ERROR**: Solo se mostrarán los errores fatales (errores irrecuperables, graves).
- **E_WARNING**. Se muestran las advertencias(***warnings***) que son errores más leves.
- **E_NOTICE**. Avisos en tiempo de ejecución (más leves que las anteriores).
- **E_CORE_ERROR**. Igual que **E_ERROR** pero solo muestra los errores procedentes del núcleo de PHP.
- **E_CORE_WARNING**. Igual que **E_WARNING** pero solo muestra los errores procedentes del núcleo de PHP.
- **E_COMPILE_WARNING**. Avisos de compilación de motores como por ejemplo Zend.
- **E_USER_ERROR**. Errores generados por el propio programador.
- **E_USER_WARNING**. Advertencias generadas por el propio programador.
- **E_USER_NOTICE**. Avisos generados por el propio programador.
- **E_STRICT**. Avisa del código utilizado por el programador que, aunque funciona, no es correcto (porque podría tener problemas en versiones futuras de PHP).
- **E_RECOVERABLE_ERROR**. Error fatal, pero que no impidió la ejecución del motor PHP al no considerarse inestable tras el fallo.
- **E_DEPRECATED**. Avisos sobre código obsoleto.
- **E_USER_DEPRECATED**. Avisos sobre código obsoleto lanzados por el propio programador.

- E_ALL. Muestra todos los errores (salvo los E_STRICT hasta la versión 5.4 de PHP).

Inicialmente el valor de **error_reporting** en el archivo php.ini suele estar establecido de esta forma:

```
error_reporting = E_ALL | E_STRICT
```

Por lo tanto se muestran todos los errores. Si queremos mostrar solo los errores graves podemos utilizar:

```
error_reporting = E_ERROR
```

En lugar de cambiar la directiva ***error_reporting*** en el archivo de configuración de PHPO, podemos utilizar la función **error_reporting,** que permite modificar el nivel de errores que se muestran, pero en tiempo de ejecución. Es decir, desde el momento en el que se invoca a la función decidiremos qué errores se mostrarán independientemente de la configuración de ***php.ini.***

Ejemplo:

```
error_reporting(E_ERROR);
```

6.3 USAR MYSQL DESDE PHP

6.3.1 CONEXIÓN A MYSQL DESDE PHP

Suponemos ya instalado y funcionando MySQL. MySQL dispone de tres API para PHP:

- La API clásica **mysql.**
- La API mejorada **mysqli** (disponible desde la versión 5.0).
- La api **PDO** (disponible desde la versión 5.1).

Se consideran activas las dos últimas. La recomendada actualmente es **mysqli.** Tanto **PDO** como **mysqli** utilizan objetos, pero **mysqli** permite también el acceso mediante funciones sin usar objetos, aunque la mayor parte de programadores utilizan objetos por ser más versátiles.

La conexión entre **PHP** y **MySQL** mediante **mysqli** requiere que se cargue con PHP la librería **php_mysqli** (en Windows **php_mysql.dll** y en Linux **php_mysq.so**). Para hacerlo hoy en día basta con quitar el comentario (si lo hubiera) referido a esa librería en el archivo de configuración **php.ini**, o bien añadir a mano esta línea (Windows):

```
extension=php_mysqli.dll
```

Por supuesto, debemos tener disponible esa librería en el directorio de extensiones. Lo normal es que en la mayoría de instalaciones no haga falta tocar nada, bien porque las instalaciones por paquetes o las de software completo, como XAMPP, vienen ya preparadas con mysqli. En la instalación en Linux mediante código fuente, se suele compilar el código PHP con la opción

--with-mysqli (véase la Unidad 2 para más información), lo que asegura la instalación correcta de la librería

6.3.2 USO DE MYSQLI. ¿FUNCIONES U OBJETOS?

Los usuarios de la librería mysql clásica de acceso a MySQL no suelen estar cómodos con el uso de objetos. Por ello se permite utilizar con mysqli la forma clásica que no implica el uso de objetos.

Sin embargo, los amantes de la programación orientada a objetos se encontrarán más cómodos trabajando con esta librería usando sus capacidades con los objetos. Para diferenciar las dos formas veamos este ejemplo:

```
$mysql = mysqli_connect("127.0.0.1", "root", "12345");
mysqli_select_db("prueba");
$resultado = mysqli_query("SELECT * FROM alumnos");
```

Esta es la forma procedimental de acceder, muy parecido a la librería clásica de acceso a MySQL. La forma orientada a objetos sería:

```
$mysqli = new mysqli("127.0.0.1", "root", "12345", "prueba");
$resultado = $mysqli->query("SELECT * FROM alumnos");
```

En este manual se ha optado por la segunda opción al ser la recomendada actualmente. A pesar de no haber explicado en este manual cómo funcionan los objetos, el uso de la librería es muy sencillo y entendible. También hay que observar que para acceder a una base de datos en el servidor local, se utiliza la dirección IP de máquina local: **127.0.0.1**, en lugar del nombre **localhost**, ya que la dirección es más seguro que funcione en todos los sistemas.

6.4 ESTABLECER CONEXIÓN CON MYSQL DESDE PHP

Si ya hemos instalado correctamente MySQL, establecer la conexión significa crear un nuevo objeto que representará la propia conexión con la base de datos. La sintaxis para crear una nueva conexión es:

```
objetoMySQL = new mysqli([servidor [,usuario [,contraseña
                        [, baseDatos [, puerto [,socket]]]]);
```

Los parámetros son todos opcionales. Se explican a continuación:

- ***servidor***. Nombre y puerto de la máquina a la que nos conectamos (es decir, el servidor **MySQL**). Se indica su nombre o IP y, opcionalmente el puerto. Si no se indica su valor se recoge de la directiva PHP (presente en el archivo de configuración **php.ini**) **mysql.default_host** que, normalmente, vale **localhost_3306** (es decir, conexión a servidor local MySQL mediante el puerto 3306, el puerto habitual de comunicación de MySQL).

- *usuario*. Nombre del usuario de la base de datos con el que nos conectamos (y que, por lo tanto, al menos tendrá permiso de conexión). De no indicarlo, se recoge de la directiva PHP **mysql.default_user**.
- *contraseña*. Contraseña del usuario con el que nos conectamos. Si no se indica se recoge de la directiva **mysql.default_pw** (por defecto vale nulo).
- *base de datos*. Indica el nombre de la base de datos con la que se trabajará en MySQL. Si no se elige, lo deberemos de hacer luego porque no se pueden ejecutar la mayoría de instrucciones SQL en MySQL sin seleccionar una base de datos.
- *puerto*. Puerto de conexión con MYSQL. Por defecto es el 3306 (puerto habitual de MySQL) que es el establecido en el parámetro **mysql.default_port**.
- *socket*. Indica el nombre del socket a utilizar. Si no, se usa el establecido en **mysql.default_socket**, que suele tomar el valor *MYSQL*.

Ejemplo:

```
$mysqli=new mysqli ("127.0.0.1","andrei","12345","prueba");
```

En el ejemplo conectamos con el servidor local de **MySQL** utilizando el usuario ***andrei*** con contraseña ***12345*** y conectamos usando la base de datos ***prueba***.

Si la conexión es correcta, la función devuelve como resultado el objeto de conexión (el enlace a datos) que se suele asignar a una variable (en el ejemplo anterior se los queda ***$con***). En el caso de que falle deberemos comprobar los errores como se explica en el Apartado 6.4.2 Control de errores en la conexión.

6.4.1 PERSISTENCIA DE LAS CONEXIONES

La librería **mysqli** admite que las conexiones sean persistentes. Sin persistencia, una conexión se cierra cuando se ha ejecutado el código PHP de una página. Sin embargo, una conexión persistente se mantiene abierta en diferentes páginas. Para ello, la conexión a la base de datos en las páginas se debe crear usando el mismo servidor, usuario y base de datos. De no ser así se creará más de una conexión.

Las conexiones persistentes ahorran recursos en el servidor de base de datos pero requiere de una mayor dedicación a cada conexión. Que se use o no persistencia depende de estas directivas del archivo de configuración (php.ini) de PHP:

- **mysqli.allow_persistent**. Puede valer 1 (se permite la persistencia) o O (no se admite la persistencia).
- **mysqli.max_persistent**. Máximo número de conexiones persistentes que se pueden crear. Con valor 0 son ilimitadas.
- **mysqli.max_links**. Máximo número de conexiones que se permite hacer.

6.4.2 CONTROL DE ERRORES EN LA CONEXIÓN

Hay tres métodos fundamentales en la comprobación de errores:

- **connect_error**. Recoge el error si lo ha habido (de otro modo valdrá falso).
- **connect_errno**. Devuelve el número del error
- **error_list**. Muestra la lista de todos los errores que han ocurrido desde que se realizó la conexión

Ejemplo:

```
$mysqli=new mysqli (localhost,'andrei','12345','prueba');
if ($mysqli->connect_error){
  echo "Error al conectar: ".$mysqli->connect_errno ." ".
                        $mysqli->connect_error;
}
```

Hasta la versión 5.3 de PHP no se podía utilizar la versión orientada a objetos de estas dos funciones por lo que se debía hacer así:

```
$mysqli=new mysqli ('localhost','andrei','12345','prueba');
if (mysqli_connect_error()){
  error "Error al conectar: ".mysqli_connect_errno()." ".
                        mysqli_connect_error();
}
```

Por último, el método **error_list**, que es accesible mediante `mysqli_error_list()` o con `$mysqli->error_list`) devuelve la lista de todos los errores provocados desde la conexión.

6.4.3 CERRAR LA CONEXIÓN CON LA BASE DE DATOS

El método **close** se encarga de cerrar la conexión. No lo debemos utilizar si deseamos que la conexión sea persistente, pero siempre es conveniente cerrar las conexiones cuando no prevemos utilizarla; de no hacerlo, estaremos malgastando recursos. MySQL con el tiempo cerrará todas las conexiones que se hayan abierto y no se utilicen, pero si dejamos que lo haga MySQL podría ocurrir que no tuviera tiempo de cerrar si abrimos demasiadas conexiones (y no cerramos ninguna).

Ejemplo (si ***$mysqli*** es el objeto creado para conectar):

```
$mysqli->close();
```

6.5 SELECCIONAR BASES DE DATOS

En MySQL las tablas pertenecen a una base de datos, por lo que es imperativo elegir la base de datos en la que vamos a trabajar. Se puede elegir durante la conexión, pero también después mediante el método **select_db()**, al que se le pasa el nombre de la base de datos.

Ejemplo:

```
$mysqli->select_db("almacenes");
```

El método devuelve **true** si no han ocurrido errores al elegir la base de datos y **false** en caso contrario.

6.6 EJECUCIÓN DE INSTRUCCIONES SQL

6.6.1 SQL GENÉRICO DE MYSQL

La función más versátil de PHP para utilizar bases de datos es la función **query.** Dicha función, recibe una instrucción SQL en forma de string y simplemente se la envía a MySQL utilizando la conexión y base de datos seleccionada. Ejemplo:

```
$mysqli->query(
   "CREATE TABLE personas(id_persona INTEGER,nombre VARCHAR(25))");
```

Esta instrucción crea, si el usuario con el que se ha conectado tiene permisos, una tabla en la base de datos en uso actualmente, de MySQL. Para comunicar con MySQL usa la variable de conexión ***$mysqli*** que, se supone, se creó anteriormente.

Si la instrucción no funciona, entonces **query** devuelve **false**. Si la instrucción es exitosa los resultados que devuelve dependen del tipo de instrucción SQL:

- Si es una instrucción de tipo **SELECT**, **SHOW**, **DESCRIBE** o **EXPLAIN**, dado que son instrucciones que devuelven un conjunto de resultados, retornan un objeto de tipo **mysqli_result**, lo que comúnmente se conoce como un ***result set*** u objeto de **conjunto de resultados**. Si la instrucción falla, el método **query** devuelve false.
- Con cualquier otro tipo de instrucción, devuelve **true** si ha sido exitosa y **false** de no ser así.

6.6.2 OBTENER EL NÚMERO DE FILAS MODIFICADAS EN INSTRUCCIONES DML

Ya se ha explicado en el apartado anterior que la función **query** es la que se utiliza para enviar instrucciones SQL a una base de datos MySQL. De esta forma las instrucciones **INSERT**, **DELETE** o **UPDATE**. Se pueden ejecutar directamente con la instrucción anterior.

Ejemplo:

```
$sql= "UPDATE personas SET salario=salario*1.1 WHERE cod_depart=9";
if($mysqli->query($sql)){
    echo "Se ha actualizado la tabla de personas";
else{
    echo "Ha fallado la instrucción";
}
```

En estas instrucciones es normal querer saber cuántas filas ha modificado la instrucción. La librería **mysqli** aporta un método para obtener esta información. Se trata de **affected_rows** que devuelve el número de filas que se han modificado. Así, podríamos mejorar el código anterior de esta forma:

```
$sql= "UPDATE personas SET salario=salario*1.1 WHERE cod_depart=9";
if($mysqli->query($sql)){
    echo "Se han modificado $mysqli->affected_rows personas";
else{
    echo "Ha fallado la instrucción";
}
```

Ahora indicamos cuántas personas han modificado la instrucción.

6.6.3 GESTIÓN DE ERRORES AL EJECUTAR INSTRUCCIONES SQL

Cuando falla una instrucción SQL, deberemos gestionar los errores para obrar de la forma más apropiada. Los métodos que gestionan este tipo de errores son:

- **error**. Contiene el mensaje de error procedente de MySQL.
- **errno**. Número de error.

Ejemplo:

```
$sql="CREATE TABLE notas(cod_persona INT(11),nota DECIMAL(5,2))";
if($mysqli->query($sql){
   echo "Tabla creada";
}
else{
   echo "Fallo al crear la tabla: ".
          $mysqli->error." nº ".$mysqli->errno;
}
```

Generalmente no debemos mostrar los mensajes de error de MySQL, más bien detectar qué error ha ocurrido (sea por el mensaje o por el número) y mostrar información que sea más entendible por nuestros usuarios.

6.7 OBTENER INFORMACIÓN MEDIANTE INSTRUCCIONES SELECT

6.7.1 USO DE LA SENTENCIA QUERY PARA EJECUTAR INSTRUCCIONES SELECT

Como se ha explicado en el Apartado 6.6.1, en las instrucciones de tipo **SELECT** (también en las instrucciones no estándar **DESCRIBE**, **SHOW** o **EXPLAIN**, propias de MySQL), se devuelve un **conjunto de resultados** (un ***result set***).

Un conjunto de resultados es una estructura que almacena resultados que tienen forma de tabla (filas y columnas). De forma práctica podemos decir que es una variable capaz de recoger el resultado de una consulta SQL. Un ejemplo sería:

```
$sql="SELECT nombre,apellido1,telefono FROM clientes";
$res=$mysqli->query($sql);
```

$res es la variable que recogerá el conjunto de resultados (valdrá **false**, si falla la instrucción SQL).

Realmente la función **query** tiene un segundo parámetro. Se trata de un número entero que indica la forma en la que vamos a recoger los datos de la consulta. Se usa con dos posibles constantes:

- **MYSQL_STORE_RESULT**. Es el valor por defecto y hace que en la variable se almacenen **todos** los resultados de la consulta. Esta forma de funcionar requiere bastante memoria y si hemos ejecutado una consulta con numerosos resultados, podemos provocar un error de **desbordamiento de memoria** y detener la ejecución de la aplicación. Si no son muchos los resultados devueltos, MYSQL_STORE_DEFAULT es la forma más conveniente de trabajar ya que es más rápida.
- **MYSQL_USE_RESULT**. Representa la forma de trabajo contraria. Los resultados se almacenan hasta llenar un búfer de resultados. El resto de resultados no se vuelcan y cuando los necesitemos, tendremos que hacer una nueva petición de datos a MySQL. La ventaja es que no gasta tanta memoria y la desventaja es que supone más peticiones a la base de datos.

Por lo tanto, si la instrucción anterior requiere almacenar mucha información, lo correcto sería:

```
$sql="SELECT nombre,apellido1,telefono FROM clientes";
$res=$mysqli->query($sql,MYSQL_USE_RESULT);
```

6.7.2 RECOGIDA DE LOS RESULTADOS

Los conjuntos de resultados son un paso previo para realmente obtener los resultados de una consulta SELECT. Hay varios métodos pertenecientes a los objetos de tipo **result set** que permiten recoger los resultados. El más interesante es **fetch_assoc**. Este método permite recorrer fila a fila los resultados de un conjunto de resultados. Cada vez que invocamos a **fetch_assoc** nos devuelve un array asociativo que contiene los datos de la fila de la consulta en la que nos encontramos actualmente; las claves de ese array son el nombre de cada columna.

Es decir, la forma habitual de recorrer los resultados de un SELECT desde PHP es:

[1] Usar **query** para lanzar la instrucción SELECT. Su resultado será un objeto de tipo ***result set*** al que le asignaremos una variable.

[2] Usar el método **fetch_assoc** de esa variable, el cual devuelve un array asociativo en el que los índices son los nombres de las columnas y los valores son los valores de la primera fila de la consulta resultado de la instrucción SELECT. Si no hay resultados, devuelve **false.**

[3] Si se vuelve a invocar a **fetch_assoc**, se cambia la posición a la fila siguiente del resultado, de la cual se devuelven los valores. Así seguiremos invocando a **fetch_assoc** hasta que devuelva **false,** indicando entonces que no hay más resultados.

Durante el proceso de lectura no se pueden ejecutar otras sentencias sobre esa conexión. Por lo que, tras leer los datos deseados, tenemos que invocar al método **close** del objeto de resultados, y así liberar los recursos de esa instrucción y, en definitiva, cerrarla.

En resumen, el proceso de lectura de datos de una consulta SELECT es el siguiente:

[1] Ejecutar la instrucción **SELECT** utilizando el método query y asignar el resultado a una variable. Si va a haber muchos resultados, utilizar como segundo parámetro de **query**, la constante **MYSQL_USE_RESULT**.

[2] Comprobar si la variable no tiene el valor **false**, que indicaría que ha fallado la instrucción, y por lo tanto, gestionar los errores (se explica más adelante como controlar los errores de la función **query**).

[3] Leer la primera fila de resultados utilizando la función **fetch_assoc** perteneciente a la variable en la que hemos almacenado el conjunto de resultados y almacenar ese resultado (un array) en una variable.

[4] Comprobar que la variable que almacena la fila no tiene valor **false**. Si vale false es que ya no hay más valores que leer.

[5] Mostrar o manipular la forma deseada los datos de la fila actual.

[6] Leer la siguiente fila invocando de nuevo a **fetch_assoc**.

[7] Volver al paso 4.

[8] Si hemos salido del bucle invocar al método **close** de la variable que almacena el conjunto de resultados, para cerrar y liberar la instrucción.

Veamos, por ejemplo, como mostrar los nombres y apellidos de una tabla de personas en forma de tabla:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
<?php
$mysqli = new mysqli("127.0.0.1", "root", "345Gtfa");
if ($mysqli->connect_error) {
    echo "Error al realizar la conexión";
}
else {
    $mysqli->select_db("trabajo");
    $sql= "SELECT nombre,apellido1 FROM personas";
    $res=$mysqli->query($sql,MYSQLI_USE_RESULT);
    if($res) {
        echo "<table><tr><th>Nombre</th><th>Apellido</th></tr>";
        $fila=$res->fetch_assoc();
        while($fila){
            echo "<tr><td>{$fila["nombre"]}</td>";
            echo "<td>{$fila["apellido1"]}</td></tr>";
            $fila=$res->fetch_assoc();
        }
        echo "</table>";
        $res->close(); //cierre de la instrucción
    }
    else{
        echo "Fallo al obtener la lista de personas ";
    }
    $mysqli->close(); //cierre de la conexión
}
?>
</body>
</html>
```

6.7.3 FUNCIONES INTERESANTES DE LOS CONJUNTOS DE RESULTADOS

- **num_rows.** Propiedad de los conjuntos de resultados, que devuelve el número de filas de la instrucción SELECT. Si la instrucción se generó con la constante MYSQL_STORE_RESULT, es decir, sin usar búfer de resultados, entonces, esta función podría fallar al

indicar las filas, ya que el cálculo exacto depende de que se hayan cargado todas las filas. En todo caso es más recomendable utilizar búferes.

- **data_seek**. Función que permite colocarlos en el número de fila que indiquemos. Por ejemplo, si ***$res*** es la variable que representa al conjunto de resultados, para colocarnos en la ultima fila de los resultados, utilizaríamos la expresión:

```
$res->data_seek($res->num_rows-1);
```

6.7.4 CODIFICACIÓN DE CARACTERES

Uno de los problemas más habituales en la lectura de datos procedentes de MySQL es la cuestión de que la página web muestre la información usando una codificación de texto (por ejemplo Unicode UTF-8) y la información se recibe de MySQL usando otra modificación (por ejemplo en formato Latín 1).

Lo normal hoy en día es que las páginas web se codifiquen en formato **UTF-8** de **Unicode**. Para conseguir mostrar texto de otras codificaciones en ese formato, podemos utilizar la función **utf8_encode**. Si, por ejemplo, en la lectura de datos del apartado anterior tuviéramos problemas con la codificación. Podríamos cambiar el bucle **while** de esta forma:

```
while($fila){
  echo "<tr><td>".utf8_encode($fila["nombre"])."</td>";
  echo "<td>".utf8_encode($fila["apellido1"])."</td></tr>";
  $fila=$res->fetch_assoc();
}
```

El método funciona, pero resulta un tanto pesado. Para evitar tener que invocar a esta función continuamente, lo lógico es hacer que todos los datos estén en formato utf-8 en MySQL y además que la comunicación con MySQL utilice ese formato para codificar el texto.

Esto último implica modificar el archivo de configuración de MySQL (**my.ini** o **my.cnf** normalmente). Hoy en día MySQL suele tener un apartado referido a UTF-8 en ese archivo, pero lo tiene entre comentarios. En todo caso para asegurar que la comunicación realmente es en UTF-8 en el apartado **[mysqld]** del archivo de configuración deberán aparecer estas líneas:

```
[mysqld]
......
## UTF 8 Settings
init-connect=\'SET NAMES utf8\'
default-character-set=utf8
collation_server=utf8_unicode_ci
character_set_server=utf8
skip-character-set-client-handshake
```

Si nuestra página PHP utiliza Unicode UTF-8 (es lo recomendable hoy en día), y estamos teniendo siempre la precaución de que dentro de MySQL trabajamos en Unicode, los datos se verán siempre de forma correcta.

6.7.5 PROBLEMAS DE SEGURIDAD. INYECCIONES DE SQL

Puesto que muchas veces los datos que se envían a través de SQL al servidor utilizan formularios que rellenan los usuarios, los hackers podrían tener un resquicio de acceso a la base de datos mediante técnicas de **inyección SQL**. Estas técnicas consisten en añadir de manera camuflada instrucciones SQL para obtener información de nuestra base de datos.

Como ejemplo sencillo y básico para hacernos idea de la importancia de tener en cuenta las inyecciones SQL, supongamos que hemos realizado una página de autentificación de usuarios a través de este formulario:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Formulario de acceso</title>
</head>
<body>
<h1>Datos de acceso</h1>
<form action="acceso.php" method="POST">
    <label for="usuario">Usuario</label>
    <input type="text" name="usuario" id="usuario"/><br/>
    <label for="pass">Contraseña</label>
    <input type="text" name="pass" id="pass"/><br/>
    <button>Enviar</button>
</form>
</body>
</html>
```

El formulario envía el usuario a través del parámetro POST de nombre ***usuario*** y la contraseña con el nombre ***pass***. La página ***acceso.php*** hará una consulta a la tabla de usuarios y si existe un usuario con ese nombre y contraseña dirá que el usuario es correcto; pero si no, nos dirá que es incorrecto. Supongamos que hemos creado este código para la página ***acceso.php***:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Página de usuario</title>
</head>
<body>
<?php
if(isset($_POST["usuario"]) && isset($_POST["pass"])) {
    $usuario=$_POST["usuario"];
```

```
    $pass=$_POST["pass"];
    $mysqli = new mysqli("127.0.0.1", "root", "s1D43We7");
    if($mysqli->connect_error) echo "Error al realizar la conexión";
    else {
         $mysqli->select_db("prueba");
         $sql="SELECT nombre,pass FROM usuarios WHERE ".
             "nombre='$usuario' and pass='$pass'";
         $res=$mysqli->query($sql);
         if($res){
             $fila=$res->fetch_assoc();
             if($fila)
                 echo "<p>Usuario y contraseña correcta</p>";
             else
                 echo "<p>Usuario o contraseña incorrecta</p>";
        }
        else echo "<p>Fallo al ejecutar la instrucción</p>";
    }
}
else header("Location:formulario.html");
?>
</body>
</html>
```

En principio todo va bien, cuando el usuario pone los datos correctos, entra correctamente y si pone una mala contraseña, no entra. Sin embargo, en cuanto en la contraseña colocáramos, por ejemplo, una comilla, el mensaje sería el de ***"Fallo al ejecutar la instrucción".*** Ese mensaje no es el habitual y da la pista de que algo raro está pasando. Y así, si el usuario (si es conocedor de las técnicas de inyección de SQL) escribe estos datos en el formulario:

Figura 6.1: Ejemplo de inyección SQL

Entonces el usuario entra, sin haber introducido contraseña alguna. La razón es que esta entrada genera la instrucción SQL:

```
SELECT nombre,pass FROM usuarios WHERE nombre='jorge' AND pass='12345
OR '1'='1'
```

El WHERE de esta instrucción siempre es verdadero porque la condición '1'='1' siempre lo es. Luego este código es muy problemático y está a expensas del ataque de terceras personas. La cuestión es cómo evitar este problema.

Estas medidas pueden ayudar:

- Utilizar la función **real_escape_string** que recibe un texto y le devuelve a su formato seguro. Lo que hará es que los caracteres peligrosos (como las comillas, saltos de línea, etc.) se pasan a su forma ***escapada.*** Por ejemplo, la comilla simple (') se convierte en (\'), con lo cual no se pueden delimitar nuevas instrucciones o elementos y estaremos más protegidos ante una inyección.

```
$sqlSeguro=$mysqli->real_escape_string($sql);
$mysqli->query($sqlSeguro);
```

- No utilizar la función **multi_query**. Esa instrucción es una variable de **query**, que admite ejecutar varias instrucciones SQL a la vez si se separa cada una de ellas con un punto y coma. Ante una eventual inyección de código SQL estaríamos en serio peligro porque si podemos ejecutar varias instrucciones a la vez, entonces, podemos añadir una instrucción SQL completa y malévola.
- Conectar, si es posible, con un usuario que tenga los mínimos privilegios posibles. Es muy mala idea conectar con el usuario root. Lo lógico en una web que accede a una base de datos solo para examinar la información, es crear un usuario con privilegios de lectura sobre las tablas que deseamos utilizar.
- Verificar o convertir los datos que se esperan sean numéricos. Teniendo en cuenta que éstos en SQL no utilizan comillas, la función **real_scape_string** no nos ayudaría. Por lo que antes de incrustar ese dato en nuestra instrucción SQL realmente habría que comprobar que es válido.
- Cifrar contraseñas y otros datos, tanto al almacenar la información como al enviarla durante la comunicación entre la base de datos y el servidor PHP. Para más información sobre cifrado de datos, véase el Apartado 4.5 *"Cifrado"*, en la página 206.

6.8 SOPORTE DE TRANSACCIONES

6.8.1 TRANSACCIONES EN MYSQL

En SQL estándar existen dos instrucciones que permiten anular o confirmar transacciones: son **ROLLBACK** (anular) y **COMMIT** (confirmar). Ambas trabajan con la transacción en curso. Una transacción comienza cuando se ejecuta una instrucción DML (**INSERT**, **DELETE** o **UPDATE**) y finaliza cuando se acepta o confirma (también puede finalizar por otras razones como cerrar la conexión o ejecutar una instrucción DDL por ejemplo).

MySQL soporta transacciones solo en bases de datos y tablas gestionadas por motores compatibles con el uso de transacciones. El motor actual, **InnoDB**, soporta transacciones de forma completa, lo que se conoce como soporte **ACID** de transacciones. ACID son las siglas referentes a:

- **Atomicidad**. Asegura que una instrucción no se pueda ejecutar a medias, o se ejecuta del todo o no hace ninguna labor.

- **Consistencia**. Asegura que el manejo de una transacción nunca deje a la base de datos en un estado incoherente.
- **Aislamiento** (*Isolation*). Asegura que las operaciones de una transacción no afectan a otras operaciones que se estén produciendo en la base de datos, permanecen independientes.
- **Persistencia** (*Durability*). Asegura que cuando confirmemos o anulemos una transacción, ese estado sea realmente definitivo y no temporal.

6.8.2 AUTOCOMMIT

Por defecto en PHP todas las instrucciones DML se confirman (**COMMIT**) al instante de forma automática, por lo que no pueden ser revocadas. Este estado es el habitual ya que es el más cómodo para evitar que una transacción quede abierta demasiado tiempo, ya que las transacciones son costosas de mantener por parte del servidor de bases de datos.

Pero el objeto de conexión (normalmente le llamamos ***$mysqli***) posee un método llamado **autocommit** que recibe como parámetro un valor booleano: true significaría que se confirma automáticamente cada instrucción DML y false que se gestionan transacciones, por lo que las instrucciones serán definitivas si se usa el método COMMIT.

6.8.3 CONFIRMAR Y ANULAR TRANSACCIONES

La forma de anular y confirmar transacciones es a través de dos métodos del objeto de conexión (***$mysqli***):

- **commit**. Confirma la transacción actual.
- **rollback**. Anula la transacción actual.

Ejemplo de funcionamiento:

```
<?php
$mysqli = new mysqli("miservidor.com", "alice", "45FdA", "almacenes");
$mysqli->autocommit(false); //se activa la gestión de transacciones

$mysqli->query("INSERT INTO ventas VALUES(12,300,'Samsung galaxy S4')");
//...
//supongamos que hemos detectado que hay que anular esa inserción:
$mysqli->rollback();//anula la transacción

//nuevas instrucciones
$mysqli->query("INSERT INTO ventas VALUES(12,350,'Samsung galaxy S4')");
$mysqli->query("UPDATE productos SET precio=precio*1.1");
//....
$mysqli->commit();//confirmamos las dos instrucciones anteriores
?>
```

6.9 PRÁCTICAS RESUELTAS

Práctica 6.1: Búsqueda de datos de localidades

- Para realizar esta práctica debemos descargar y ejecutar (con permiso de root en MySQL) los archivos de esta prácticas presentes en la dirección http://www.jorgesanchez.net/libro-iaw
- Una vez hayamos ejecutado esas instrucciones, nuestra instalación de MySQL tendrá una nueva base de datos llamada ***geografia*** que contiene datos de la mayoría de localidades de España.
- La estructura relacional de esas tablas es la siguiente:

- La práctica consiste en encontrar una localidad eligiendo primero la comunidad en la que está, luego una provincia de esa localidad y finalmente elegir la localidad de una lista de las localidades de esa provincia.
- Para ello un primer formulario nos muestra la lista de comunidades.
- Tras elegir la comunidad, un segundo formulario nos muestra las provincias de esa comunidad.
- Finalmente un tercer formulario nos muestra una lista con las localidades de la provincia elegida.
- Cuando elijamos la localidad deseada, se nos enseña la población que tiene esa localidad. Eso se hará debajo de la lista de localidades, la misma página nos muestra ese resultado.
- Si acudimos a la provincia sin haber pasado vía GET una comunidad, se nos devolverá al formulario de las comunidades. Si vamos a las localidades sin haber pasado una provincia, se nos enviará al formulario de provincias.

SOLUCIÓN: PRÁCTICA 6.1

Tras descargar los archivos de scripts (son ***crearTablas.sql*** y ***datos.sql***) se ejecutan desde cualquier programa que admita la ejecución de scripts, como **MySQL Workbench** o **PHPMyAdmin**. Incluso se puede desde la línea de comandos. Supongamos que hemos descargado los archivos en el directorio bin de MySQL. Si nos situamos en ese directorio desde la línea de comandos (indi-

ferentemente de si estamos en Windows o en Linux) y ejecutamos las instrucciones de la página siguiente.

```
./mysql -u root -p < crearTablas.sql
./mysql -u root -p < datos.sql
```

Tras cada instrucción tendremos que indicar cuál es la contraseña del usuario root, pero tras la ejecución (el segundo script tarda bastante en ejecutarse) tendremos los datos listos. Conviene crear un usuario al que solo permitamos la lectura de las tablas por seguridad. En el código de ejemplo a ese usuario le hemos llamado *geo*.

A continuación mostramos el código del primer formulario al que hemos llamado **comunidades-practica1.php**:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Elección de la comunidad autónoma</title>
    <link rel="stylesheet" href="estilos.css"/>
</head>
<body>
<?php
//conexión a  la base de datos, la haremos de
// tipo persistente por eso no la cerramos
$mysqli=new mysqli("127.0.0.1","geo","t1234Fa","geografia");
if($mysqli) {
    $res=$mysqli->query("SELECT nombre FROM comunidades".
                    " ORDER BY nombre");
    if($res) {
?>
        <form action="provincias-practica1.php">
            <label for="comunidad">Elija la comunidad deseada</label>
            <select name="comunidad" id="comunidad">
                <?php
                $fila=$res->fetch_assoc();
                while($fila){
                    echo "<option value='{$fila["nombre"]}'>".
                        "{$fila["nombre"]}</option>";
                    $fila=$res->fetch_assoc();
                }
                ?>
            </select>
            <button>Buscar provincias</button>
        </form>
```

```
<?php
    }
    else
        echo "<p class='error'>No se pudo obtener la lista de
            ".datos</p>";
}
else{
    echo "<p class='error'>Fallo de conexión</p>";
}
?>
</body>
</html>
```

Se puede observar que se incrusta varias veces código PHP dentro del código HTML, facilita la escritura del código, pero complica un poco su lectura. A continuación se presenta el código del formulario de **provincias-practica1.php**, es muy parecido al anterior solo que la instrucción SQL necesita buscar en dos tablas (por lo que es un poco más compleja) y que se detecta si se envía por GET el nombre de la comunidad (si no se recibe, se vuelve al formulario anterior):

```
<?php
if(isset($_GET["comunidad"])==false)
    header("Location:comunidades-practica1.php");
else $comunidad=$_GET["comunidad"];
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Elección de la comunidad autónoma</title>
    <link rel="stylesheet" href="estilos.css"/>
</head>
 <body>
<?php
$mysqli=new mysqli("127.0.0.1","geo","t1234Fa","geografia");
if($mysqli) {
    $sql="SELECT p.nombre provincia,c.nombre comunidad ".
         "FROM provincias p ".
         "JOIN comunidades c USING(id_comunidad) ".
         "WHERE c.nombre='$comunidad' ".
         "ORDER BY provincia";
    $res=$mysqli->query($sql);
    if($res) {
?>
```

```
        <form action="localidades-practica1.php">
            <label for="provincia">Elija la provincia deseada</label>
            <select name="provincia" id="provincia">
                <?php
                $fila=$res->fetch_assoc();
                while($fila){
                    echo "<option value='{$fila["provincia"]}'>".
                        "{$fila["provincia"]}</option>";
                    $fila=$res->fetch_assoc();
                }
                ?>
            </select>
            <button>Buscar localidades</button>
        </form>
<?php
    } else
        echo "<p class='error'>No se pudo obtener el listado</p>";
} else
    echo "<p class='error'>No se pudo conectar con la base de ".
        "datos</p>";
?>
</body>
</html>
```

El código del formulario final en el que se muestran las localidades de la provincia elegida, es el código más complejo. En él, si no se envía por GET un nombre de provincia, volvemos al formulario anterior (así evitamos que directamente se intente entrar en esta página). Mostraremos una lista combinada con todas las localidades, en orden alfabético, de la provincia elegida. Finalmente cuando elijamos una localidad, esta misma página detecta que se le ha enviado una localidad y tras el cuadro anterior se muestra la población de esa localidad.

La dificultad está en el momento de pasar la localidad, como queremos que se siga mostrando la lista de localidades de la provincia elegida para facilitar elegir otra localidad de la misma provincia, la solución está en utilizar un control de formulario oculto en el que se almacena la provincia que habíamos elegido. De esa forma cuando pasemos la localidad, pasaremos también la provincia y así la página es más grata para el usuario.

Se expone el código de **localidades-practica1.php**:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Elección de la comunidad autónoma</title>
    <link rel="stylesheet" href="estilos.css"/>
```

```
</head>
<body>
<?php
if(isset($_GET["provincia"])==false)
    //sin haber indicado provincia, volvemos al formulario anterior
    header("Location:provincias-practica1.php");
else {
    $mysqli = new mysqli("127.0.0.1", "geo","t1234Fa","geografia");
    if ($mysqli) {
        //hemos recibido la provincia , buscamos sus localidades
        $provincia = $_GET["provincia"];
        $sql = "SELECT l.nombre localidad " .
            "FROM localidades l " .
            "JOIN provincias p USING(n_provincia) " .
            "WHERE p.nombre='$provincia' " .
            "ORDER BY localidad";
        $res = $mysqli->query($sql,MYSQLI_USE_RESULT);
        if ($res) {
?>
            <form action="localidades-practica1.php">
                <label for="localidad">
                    Elija la localidad deseada</label>
                <select name="localidad" id="localidad">
                    <?php
                    $fila = $res->fetch_assoc();
                    while ($fila) {
                        echo "<option value='{$fila["localidad"]}'>" .
                            "{$fila["localidad"]}</option>";
                        $fila = $res->fetch_assoc();
                    }
                    ?>
                </select>
                <!-- la provincia la reenviamos como
                control oculto de formulario -->
<?="<input type='hidden' name='provincia' value='$provincia'/>"?>
                <button>Buscar localidades</button>
            </form>
        <?php
        } else
            echo "<p class='error'>No se pudo obtener el listado</p>";
        if (isset($_GET["localidad"])) {
            //hemos recibido una localidad, la buscamos
            // y mostramos los datos
```

```
            $localidad = $_GET["localidad"];
            $sql = "SELECT nombre,poblacion FROM localidades " .
                "   WHERE nombre='$localidad'";
            $res = $mysqli->query($sql);
            if ($res->num_rows == 0)
                echo "<p class='error'>No existe la localidad " .
                    "$localidad</p>";
            else {
                $fila = $res->fetch_assoc();
                echo "<p>Localidad $localidad, poblacion= " .
                    $fila["poblacion"] . " habitantes</p>";
            }
        }
    }//fin if de conexión
    else
        echo "<p class='error'>No se pudo conectar</p>";
}//fin else inicial
?>
</body>
</html>
```

Práctica 6.2: Paginación de resultados

- Utiliza los datos de la base de datos geografia. Para obtenerlos e instalarlos véase el enunciado e inicio de la solución de la Práctica 6.1.
- Muestra una primera página que pida mediante un cuadro de texto el nombre de una provincia española.
- Al enviar ese dato se busca la provincia. De no existir se indica que no existe. Hay que tener en cuenta que, al buscar, no tendremos en cuenta si la provincia se ha escrito en minúsculas o en mayúsculas.
- Si la provincia sí existe, se mostrarán los datos de todas sus localidades en orden alfabético y de forma paginada. Se mostrarán 25 localidades por página. Una serie de botones nos permitirán pasar a la página siguiente, a la anterior o a un número de página concreto.

SOLUCIÓN: PRÁCTICA 6.2

El código del primer formulario es sencillo. Es una página html en la que simplemente se pide el nombre de la provincia y se envía dicho código a la página listado-practica2.php que será la encargada de realizar todo el proceso de gestión del listado de las localidades.

Código de **form-practica2.html**:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Selección de provincia</title>
</head>
<body>
<form action="listado-practica2.php">
    <label for="provincia">Escriba el nombre de la provincia</label>
    <input type="text" name="provincia" id="provincia"/><br/>
    <button>Enviar</button>
</form>
</body>
</html>
```

La página que calcula el listado primero comprueba que la provincia existe. Para ello ejecuta la consulta SQL y si ésta no devuelve resultados, entonces es que la provincia no existe.

Para la paginación la idea es que a la página le podremos pasar un parámetro tipo GET llamado ***pg*** en el que indicamos la página que queremos ver. Si no se recibe página alguna, entenderemos que queremos ver la primera. Los enlaces con el número de página y el anterior y el posterior, simplemente apuntan a la misma página, pero añadiendo en la cadena de consulta el parámetro pg y el de la provincia que estamos viendo, es fundamental que no se nos olvide reenviar continuamente el nombre de la provincia.

Este es el armazón complejo de la práctica, pero hay que tener en cuenta que es muy habitual paginar resultados, ya que el usuario agradece mostrar así los resultados, por lo que merece la pena el esfuerzo.

La última cuestión es el cálculo de las páginas, para lo cual está la función num_rows que devuelve el número total de filas, así como la función seek nos permite situarnos en un número de fila concreto. Finalmente, es importante que los resultados se carguen en modo búfer para lo cual la función **query** debe utilizar la constante **MYSQLI_USE_RESULT**.

Código de ***listado-practica2.php***:

```
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Listado de localidades</title>
    <style>
        .error{
            color:red;
        }
```

```
        .pagina{
            background-color: lightgray;
            font-size:.7em;
        }
        table{
            border-collapse: collapse;
            width:100%;
        }
        td{
            border:1px solid black;
        }
        td:first-of-type{
            font-weight:bold;
        }
        td:last-of-type{
            text-align: right;
            padding-right:10px;
        }
        th{
            background-color: black;
            color:white;
        }
        .paginaActual{
            border:1px solid black;
        }
    </style>
</head>
<body>
<?php
if(isset($_GET["provincia"])==false)
    header("Location:form-practica1.php");
else {
    $mysqli = new mysqli("127.0.0.1", "geo", "4fRw2", "geografia");
    if ($mysqli) {
        $provincia = $_GET["provincia"];
        //paso a mayúsculas de la provincia
        $provincia=strtoupper($provincia);
        //ahora cambiamos los acentos, de otro modo fallan
        $provincia=strtr($provincia,'áéíóúü','ÁÉÍÓÚÜ');
        $sql = "SELECT l.nombre localidad, poblacion " .
            "FROM localidades l " .
            "JOIN provincias p USING(n_provincia) " .
            "WHERE UPPER(p.nombre)='$provincia' " .
            "ORDER BY localidad";
```

```
        $res=$mysqli->query($sql);
        if($res->num_rows==0){
            echo "<p class='error'>No existe esa provincia</p>";
        }
        else{
            //****Texto de cabecera de tabla indicando lo que vemos
            echo "<h1>Localidades de $provincia</h1>";
            //comprobación y preparación del nº de página
            if(isset($_GET["pg"])) {
                $pg = $_GET["pg"];
                if($pg<=0 || is_numeric($pg)==false) $pg=1;
            }
            else
                $pg=0;
            //*******preparación de la barra de paginación
            //variable con la parte común de los enlaces, falta calcular
            //la página para cada uno de ellos
            $direccion="listado-practica2.php?provincia=$provincia&pg=";
            echo "<p class='pagina'>Página: ";
            if($pg>1)
                //botón de página anterior, símbolo <
                echo "<a href='$direccion".($pg-1)."'>&lt;</a>";
            //cálculo del total de páginas
            $total_pg=(int)($res->num_rows/25+1);
            //botones para ir a una página concreta
            for($i=1;$i<=$total_pg;$i++){
                if($i==$pg) //para que salga diferente el nº actual
                    echo " <span class='paginaActual'>$i</span> ";
                else
                    echo "<a href='$direccion".$i."'>$i </a>";
            }
            if($pg<$total_pg)
                //botón de página siguiente, símbolo >
                echo "<a href='$direccion".($pg+1)."'>&gt;</a>";
            echo "</p>";
        //***********cálculo de la posición de fila según la página
deseada
            $posicion=($pg-1)*25;
            $res->data_seek($posicion);
        //*******listado de localidades, como mucho se deben listar 25
            $cont=1;
            $fila=$res->fetch_assoc();
            echo "<table><tr><th>Localidad</th>".
                 "<th>Población</th></tr>";
```

```
            while($fila && $cont<=25){
                echo "<tr><td>{$fila['localidad']}</td>";
                echo "<td>{$fila['poblacion']} habitantes</td>";
                $fila=$res->fetch_assoc();
                $cont++;
            }
            echo "</table>";
        }
    }
}
?>
</body>
</html>
```

Práctica 6.3: Servicio de mensajería

- Crea un pequeño servicio de mensajería instantánea accesible desde una aplicación web. De modo que el funcionamiento sea el siguiente:
  - En la página de acceso los usuarios tienen que poder darse de alta (simplemente indicando su nombre de usuario y contraseña) o simplemente decir que ya están dados de alta y entrar en su cuenta indicando el usuario y la contraseña.
  - Los nombres de usuario solo pueden contener letras y números (no vale espacio). La contraseña tiene que tener entre 6 y 30 caracteres.
  - Al entrar en su cuenta aparecen los mensajes que tienen y el nombre del usuario que les ha enviado el mensaje.
  - Desde cada cuenta se nos permite la posibilidad enviar un mensaje a otro usuario.
- Es imperativo controlar los errores para que el usuario sepa lo que ocurre en cada momento

SOLUCIÓN: PRÁCTICA 6.3

Esta es una práctica excelente para crear una aplicación que incorpore la mayor parte de elementos de PHP vistos en este manual: sesiones, paso de parámetros por GET, uso de bases de daztos, formularios, expresiones regulares, etc.

Antes de trabajar necesitamos crear la estructura de base de datos. Para ello crearemos una base de datos llamada ***mensajes*** dos tablas: una para los usuarios y otra para los mensajes. Este sería el código SQL necesario:

```
CREATE DATABASE mensajes;
USE mensajes;
CREATE TABLE usuarios(
   id_usuario INT NOT NULL AUTO_INCREMENT,
   usuario VARCHAR(30) NOT NULL,
   pass VARCHAR(100) NOT NULL,
   CONSTRAINT usuarios_pk PRIMARY KEY(id_usuario),
   CONSTRAINT usuarios_uk UNIQUE(usuario)
);

CREATE TABLE mensajes(
   id_mensaje INT NOT NULL AUTO_INCREMENT,
   texto VARCHAR(300) NOT NULL,
   id_remite INT NOT NULL,
   id_destino INT NOT NULL,
   CONSTRAINT mensajes_pk PRIMARY KEY(id_mensaje),
   CONSTRAINT usuarios_fk1 FOREIGN KEY(id_remite)
            REFERENCES usuarios(id_usuario),
   CONSTRAINT usuarios_fk2 FOREIGN KEY(id_destino)
            REFERENCES usuarios(id_usuario)
);
```

La gestión de errores la haremos de forma que cuando una página realiza una acción (por ejemplo, dar de alta a un usuario) si ésta falla, que devuelva un código de error (vía GET, que es más cómodo) a la página que invocó el servicio. Para centralizar y que sea más entendible el código, el documento ***errores-practica3.php*** tiene este código:

```
<?php
//constantes para errores
const ERROR_CONEXION=1;
const ERROR_DOS_PASSWORD=2;
const ERROR_USUARIO_EXISTE=4;
const ERROR_USUARIO_NO_EXISTE=8;
const ERROR_USUARIO_PASSWORD=16;
const ERROR_USUARIO_INVALIDO=32;
const ERROR_GRABAR=64;
const ERROR_PASSWORD_CORTA=128;

//array de mensajes, que relaciona
//códigos de error con su mensaje
$mensajeError=array(
    ERROR_CONEXION=>"Fallo en la conexión con la base de datos",
```

```
    ERROR_DOS_PASSWORD=>"Las contraseñas no coinciden",
    ERROR_USUARIO_EXISTE=>"Ese nombre de usuario ya se está ".
                          "utilizando",
    ERROR_USUARIO_NO_EXISTE=>"No existe ningún usuario con ese nombre",
    ERROR_USUARIO_PASSWORD=>"El nombre de usuario o la contraseña no "
                            "coinciden",
    ERROR_USUARIO_INVALIDO=>"Nombre de usuario incorrecto, ".
        "solo se admiten letras y números",
    ERROR_GRABAR=>"Fallo al grabar los datos",
    ERROR_PASSWORD_CORTA=>"La contraseña debe tener al menos 6 ".
                          "caracteres"
);
?
```

Los números de error se ponen en potencias de dos (1,2,4,8,16,etc) de modo que podamos, si es necesario, con un mismo número enviar varios mensajes de error. Así el número 5 indicaría que el usuario existe (error número 4) y un fallo de conexión (error número 1) a la vez .

A la página inicial, la llamaremos ***index-practica3.php*** y simplemente contiene dos formularios (cada uno ocupa la mitad de la pantalla) el primero para ir a nuestro buzón si ya estamos dados de alta y el segundo para darnos de alta.

Código de ***index-practica3.php***, observar como incorpora el código de los errores y como mostraría el error en el caso de que reciba un código de error.

```
<!doctype html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <title>Servicio de mensajeria, entrada</title>
   <link rel="stylesheet" href="estilos-practica3.css">
</head>
<body>
   <h1>Conexión con el servicio de mensajería</h1>
   <div id="izda"><h2>Entrar en la cuenta</h2>
      <form action="login-practica3.php" method="POST">
        <input type="text" name="usuario" placeholder="usuario"><br>
        <input type="password" name="password"
                    placeholder="contraseña"><br>
        <button>Validar</button>
      </form>
   </div>
   <div id="dcha"><h2>Nuevo usuario </h2>
      <form action="nuevo-practica3.php" method="POST">
        <input type="text" name="usuario" maxlength="30"
            placeholder="usuario" required><br>
```

```
        <input type="password" name="password" maxlength="30"
          placeholder="contraseña"><br>
        <input type="password" name="password2" maxlength="30"
          placeholder="Repita contraseña" required><br>
        <button>Alta de usuario</button>
      </form>
    </div>
    <div id="error">
      <?php
        include "errores-practica3.php";
        if(isset($_GET["error"])){
         $error=$_GET["error"];
        echo "<p><strong>Error:</strong> {$mensajeError[$error]}</p>";
        } ?>
    </div>
</body>
</html>
```

Vemos ahora el código para añadir un nuevo usuario. La cuestión es que ese usuario puede ya existir. Además, validamos que el nombre de usuario tenga solo letras y/o números. La contraseña debe tener al menos 6 caracteres también. Del usuario cogeremos los 30 primeros caracteres, al igual que de la contraseña (por si nos han intentado enviar más caracteres). La contraseña se guarda cifrada.

Si el usuario ya existe enviaremos de nuevo el flujo a la página ***index***, lo mismo si la contraseña se repite o hay otro tipo de error. Si los datos son correctos, el flujo se dirigirá al buzón del nuevo usuario, habiendo creado una sesión y grabando el usuario como variable de sesión. Por eso, el código de la página (se llama **nuevo-practica3.php**) es solo PHP:

```
<?php
session_start(); //soporte de sesiones
include "errores-practica3.php";
$error=0;//centinela de errores
if(isset($_POST["usuario"]) &&
    isset($_POST["password"]) &&
    isset($_POST["password2"]))
{
    //recogemos los parámetros asegurando que cogemos como
    //mucho el máximo tamaño permitido
    $usuario=substr($_POST["usuario"],0,30);
    $password=substr($_POST["password"],0,30);
    $password2=substr($_POST["password2"],0,30);
    if(preg_match("/(*UTF8)^[\p{L}\p{N}]{1,30}$/",$usuario)) {
        if ($password == $password2) {
            if(strlen($password)>=6) {
```

```
                $mysqli = new mysqli("127.0.0.1", "mensajes", "edsFG1",
                                     "mensajes");
                if ($mysqli) {
                    $cifrada = password_hash($password,
                                                PASSWORD_DEFAULT);
                    $sql = "INSERT INTO usuarios(usuario,pass) " .
                        "VALUES('$usuario','$cifrada')";
                    if ($mysqli->query($sql) == false) {
                        //1062=código MySQL de índice duplicado
                        if($mysqli->errno==1062)
                            $error=ERROR_USUARIO_EXISTE;
                        else
                            $error = ERROR_GRABAR;
                    }
                    else
                        $_SESSION["usuario"] = $usuario;
                } else $error=ERROR_CONEXION;
            }
            else $error=ERROR_PASSWORD_CORTA;
        }
        else $error=ERROR_DOS_PASSWORD;
    }
    else $error=ERROR_USUARIO_INVALIDO;
}
else{
    header("Location:index-practica3.php");
}
if($error!=0)
    header("Location:index-practica3.php?error=".$error);
else
    header("Location:buzon-practica3.php");
?>
```

El siguiente código PHP es el correspondiente a la página ***login-practica3.php*** que se encarga de validar a un usuario y si su contraseña y usuario son correctas, pasamos a su buzón. El funcionamiento es parecido al de la página anterior:

```
<?php
session_start();
include "errores-practica3.php";
if(isset($_POST["usuario"]) &&
    isset($_POST["password"]))
{
    $usuario=substr($_POST["usuario"],0,30);
    $password=substr($_POST["password"],0,30);
```

```
    if(preg_match("/(*UTF8)^[\p{L}\p{N}]{1,30}$/",$usuario)) {
        if(strlen($password)>=6) {
                $mysqli = new mysqli("127.0.0.1", "mensajes", "edsFG1",
                                     "mensajes");
            if ($mysqli) {
                $sql = "SELECT usuario,pass FROM usuarios ".
                       "WHERE usuario='$usuario'";
                $res=$mysqli->query($sql);
                if ($res == false)
                    $error = ERROR_CONEXION;
                else{
                    $fila=$res->fetch_assoc();
                    if($fila){
                        //verificamos contraseña
                        if(password_verify($password,$fila["pass"])){
                            //contraseña correcta, preparamos sesión
                            $_SESSION["usuario"]=$fila["usuario"];
                            $res->close();
                        }
                        else $error=ERROR_USUARIO_PASSWORD;
                    }
                    else $error=ERROR_USUARIO_NO_EXISTE;
                }
            } else $error = ERROR_CONEXION;
        }
        else $error=ERROR_PASSWORD_CORTA;
    }
    else $error=ERROR_USUARIO_INVALIDO;
}
else{
    header("Location:index-practica3.php");
}

if($error!=0)
    header("Location:index-practica3.php?error=".$error);
else
    header("Location:buzon-practica3.php");
?>
```

Indudablemente, la página más compleja es la dedicada al buzón de mensajes o espacio de usuario ya que ofrece cuatro funciones: Mostrar la lista de mensajes (si no hay mensajes, se indica con un texto), posibilitar enviar nuevos mensajes, cerrar la sesión de usuario y mostrar los errores ocurridos durante el envío de un mensaje (si les hubo). Cada una de estas funciones está

dentro de una capa DIV a la hora de crear esta página es bueno ir solucionando cada función por separado. El código que proponemos para esta página, ***buzon-practica.php*** es:

```
<?php
session_start();
include "errores-practica3.php";
if(isset($_SESSION["usuario"])==false)
    //si no hay sesion de usuario vamos al formulario
    header("Location:index-practica3.php");
else
    //buscamos a ver si ese usuario existe
    $usuario=$_SESSION["usuario"];
    $mysqli = new mysqli("127.0.0.1","mensajes", "edsFG1", "mensajes");
    if ($mysqli) {
        $sql = "SELECT usuario,id_usuario FROM usuarios " .
            "WHERE usuario='$usuario'";
        $res=$mysqli->query($sql);
        if($res){
            if($res->num_rows==0)
                //no se ha encontrado al usuario, algo raro pasa
                //devolvemos el flujo al formulario de entrada
                header("Location:index-formulario3.php");
            else{                   //recogemos el id_usuario
                $fila=$res->fetch_assoc();
                $id_usuario=$fila["id_usuario"];
                //cerramos instrucción, pero la conexión sigue abierta
                $res->close();
            }
        }
        else
           header("Location:index-practica3.php?error=".ERROR_CONEXION);
    }
    else
        header("Location:index-practica3.php?error=".ERROR_CONEXION);
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Buzón de mensajes</title>
    <link rel="stylesheet" href="estilos-practica3.css"/>
</head>
<body>
<h1>Hola <?=$usuario?></h1>
<div id="izda">
```

```
    <h2>Lista de mensajes</h2>
    <?php
    //a la izquierda mostramos la lista de mensajes
    //buscamos los mensajes del usuario
    $sql="SELECT usuario remite,texto FROM mensajes ".
         "JOIN usuarios ON(id_usuario=id_remite) ".
         "WHERE id_destino=$id_usuario  ORDER BY id_mensaje DESC";
    $res=$mysqli->query($sql);
    if($res){
        if($res->num_rows==0)
            echo "<p><strong>No hay mensajes</strong></p>";
        else{
            $fila=$res->fetch_assoc();
            echo "<ul>";
            while($fila){
                echo "<li><strong>De: </strong>{$fila["remite"]}<br>";
                echo $fila["texto"]."</li>";
                $fila=$res->fetch_assoc();
            }
            $res->close();
            echo "</ul>";
        }
    }
    ?>
</div>
<div id="dcha">
    <h2>Nuevo mensaje</h2>
    <form action="enviar-practica3.php" method="POST">
        <input type="text" name="destinatario" maxlength="30"
               placeholder="Destinatario" required><br>
  <!--de manera oculta pasamos el id_usuario como remite del mensaje-->
        <input type="hidden" name="id_remite" value="<?=$id_usuario?>"/>
        <textarea name="texto" id="texto" cols="30" rows="10"
                  placeholder="Texto del mensaje"></textarea><br/>
        <button>Enviar mensaje</button>
    </form>
</div>
<div id="error">
    <?php
    if(isset($_GET["error"])){
        $error=$_GET["error"];
        if(isset($mensajeError[$error]))
          echo "<p><strong>Error:</strong>{$mensajeError[$error]}</p>";
    } ?>
</div>
```

```
<div id="cerrar">
    <p><a href="cerrar-practica3.php">Cerrar sesión</a></p>
</div>
</body>
</html>
```

El código de la página de envío de mensajes **envio-practica.php** se encarga de comprobar que tiene todos los datos. Se debe asegurar que el destinatario existe, de otro modo se prepara un código de error para que el buzón de mensajes le muestre. Si ese destinatario existe, recogemos su identificador y como tenemos el del remite (que hemos pasado oculto desde el buzón) simplemente añadimos una nueva fila a la tabla de mensajes. Esta página solo muestra mensajes si todo ha ido bien, en caso contrario, entrega el error al buzón para que se encargue de mostrar el texto de error.

```
<?php
include "errores-practica3.php";
if(isset($_POST["id_remite"]) && isset($_POST["texto"])
    && isset($_POST["destinatario"])){
    $id_remite=$_POST["id_remite"];
    $destinatario=$_POST["destinatario"];
    $texto=$_POST["texto"];
    $error=0;
    $mysqli = new mysqli("127.0.0.1","mensajes", "edsFG1", "mensajes");
    if ($mysqli) {
        //búsqueda del destinatario
        $sql = "SELECT usuario,id_usuario FROM usuarios " .
            "WHERE usuario='$destinatario'";
        $res=$mysqli->query($sql);
        if($res){
            if($res->num_rows==0)
                $error=ERROR_USUARIO_NO_EXISTE;
            else {
              //el usuario existe, tomamos su id y añadimos el mensaje
              $fila = $res->fetch_assoc();
              $id_destino=$fila["id_usuario"];
              $res->close();
              $texto=$mysqli->real_escape_string($texto);
              $sql="INSERT INTO mensajes(texto,id_remite,id_destino) ".
                    "VALUES ('$texto',$id_remite,$id_destino)";
              if(($mysqli->query($sql))==false)
                    $error=ERROR_GRABAR;
              else
                    $mysqli->close();
            }
        }
```

```
        else $error=ERROR_CONEXION;
    }
    else $error=ERROR_CONEXION;
}
else  header("Location:buzon-practica3.php");

if($error!=0);
    header("Location:buzon-practica3.php?error=".$error);
?>
<!doctype html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
<h1>Mensaje enviado</h1>
<p><a href="buzon-practica3.php">Volver al buzón</a></p>
</body>
</html>
```

Finalmente el código PHP más sencillo es el de la página que cierra la sesión de usuario y devuelve el control al formulario de entrada. Código de **cerrar-practica3.php**:

```
<?php
//eliminación de la sesión
session_start();
unset($_SESSION);
session_destroy();
//volvemos al formulario de alta
header("Location:index-practica3.php");
?>
```

Solo falta el código de la hoja de estilos, que es claramente mejorable, pero que permite hacernos una idea de como se ven las diferentes áreas. Código del archivo **estilos-practica3.css:**

```
#izda{
    float:left;
    width:50%;
    max-height:600px;
    overflow-y: auto;
}
#dcha{
    float:left;
```

```
    width:50%;
}
#cerrar {
    float:left;
    width:100%;
}
#error {
    float:left;
    width:100%;
    background-color: red;
    color:white;
    margin-top:4em;
}
.error{
    float:left;
    color:red;
}
h1{
    text-align: center;
}
```

6.10 PRÁCTICAS PROPUESTAS

Práctica 6.4: Mejora del servicio de mensajería

- Consigue que dentro del buzón de cada usuario, en cada mensaje dispongamos de un botón o un enlace que nos permita eliminar el mensaje.
- Implementa una mejora tanto en la base de datos como en la aplicación web que nos permita detectar qué mensajes son nuevos y, por ejemplo, colorearles distintos del resto.

Práctica 6.5: Juego geográfico

- Utilizando la base de datos geográfica de la Práctica 6.1 y la Práctica 6.2, haz un juego de conocimientos geográficos.
- En este juego debe aparecer una localidad aleatoria de la lista de localidades de esa base de datos
- El usuario debe escribir en qué provincia está.
- En todo momento iremos diciendo al usuario cuántos aciertos lleva, cuántos errores lleva y cuál es el porcentaje de aciertos.

Práctica 6.6: Ocupación de cursos

- Crea en MySQL una base de datos llamada **cursos** con una sola tabla llamada también **cursos.** En la tabla almacenaremos:
  - El título del curso
  - El número de plazas disponibles
  - El número de plazas ocupadas
- Con esta tabla crear una aplicación que permita mostrar y añadir más ocupación a los cursos. La idea es que aparezca la lista mostrando el título de cada curso, las plazas ocupadas, las plazas libres y un enlace o botón que permita ocupar una plaza más en ese curso.
- Los cursos que tienen ocupadas todas las plazas aparecerán en formato tachado.
- Al final de la lista se muestra el total de plazas ofertadas, el total de disponibles y el porcentaje de ocupación.
- Ejemplo de listado:

Lista de cursos

Cursos disponibles	Plazas totales	Plazas disponibles	
~~Escalada~~	~~18~~	~~0~~	
Guitarra acústica	20	8	Añadir plaza
Guitarra española	15	11	Añadir plaza
Historia de Palencia	15	12	Añadir plaza
Marcha nórdica	15	1	Añadir plaza
Montañismo	15	15	Añadir plaza
~~Natación para bebés~~	~~10~~	~~0~~	
~~Natación para bebés 2~~	~~10~~	~~0~~	
Natación-avanzado	18	14	Añadir plaza
Natación-básico	30	7	Añadir plaza
Paddle	15	6	Añadir plaza
~~Running~~	~~15~~	~~0~~	
Running avanzado	25	1	Añadir plaza
Spinning	15	12	Añadir plaza
Yoga	15	7	Añadir plaza

Resumen de ocupación:

- Plazas totales ofertadas: 251
- Plazas ocupadas: 157
- Porcentaje de ocupación: 62.549800796813%

Práctica 6.7: Tienda on line

- Implementa una tienda on line.
- Con ella mostrarás una lista de artículos con su descripción, al menos una foto y el precio.
- En cada artículo un enlace nos permitirá meterlo en el carrito de la compra.
- En el carrito podremos ver lo que hemos comprado, así como modificar la cantidad comprada de cada artículo o bien quitar al artículo del carrito. Incluso borrar el carrito.
- El carrito debe mantener una semana los artículos por si el usuario cierra la sesión, que pueda seguir con el mismo carrito.

6.11 RESUMEN DE LA UNIDAD

- Las bases de datos aportan importantes ventajas a la creación de aplicaciones web, fundamentalmente: almacenamiento permanente, una capa extra de seguridad, herramientas avanzadas de gestión de datos y el uso de un lenguaje especializado en datos (normalmente SQL).
- El acceso a las bases de datos desde PHP requiere cargar un módulo que contiene la API de acceso a la base de datos concreta (por ejemplo ***mysqli***) al arrancar PHP. En ocasiones se requiere también instalar un software conector en el servidor (por ejemplo el ***Oracle Instant Client*** para conectar con servidores ***Oracle Database***).
- Los datos necesarios para conectar con una base de datos son: la IP o nombre del servidor de base de datos, el puerto que utiliza, el nombre de la base de datos a utilizar, un usuario y su contraseña de acceso.
- Para acceder a MySQL las librerías recomendadas son **mysqli** y **pdo**. La primera es la que más se utiliza actualmente.
- El proceso de acceso a una base de datos MySQL con la librería **mysqli** es:
  - [1] Crear el objeto de conexión (normalmente se llama ***$mysqli***).
  - [2] Ejecutar la instrucción SQL con ayuda del método **query**.
  - [3] Recoger los datos de la instrucción si es de consulta con ayuda del método **fecth_assoc**. Si no es de consulta basta con comprobar si se ha ejecutado bien.
  - [4] Cerrar la conexión.
- Un problema de seguridad habitual es la inyección de SQL, que puede permitir un acceso no deseado a la base de datos. Para evitar este (y otros problema) hay que tomar las medidas pertinentes.
- Es posible utilizar instrucciones de transacción con MySQL, siempre y cuando el motor de la base de datos lo permita.

6.12 TEST DE REPASO

¿Cuál de los siguientes enunciados no es una ventaja de las bases de datos?

- Añaden una capa más de seguridad
- Utilizan el propio lenguaje PHP para extraer los datos que poseen

¿Cuál de los siguientes datos no es obligatorio indicar en la cadena de conexión a una base de datos MySQL?

a) Nombre o dirección IP del servidor
b) Nombre del usuario
c) Contraseña
d) Nombre de la base de datos

¿Qué puerto normalmente utiliza MySQL para comunicarse?

a) 3306
b) 1521
c) 4096
d) 80
e) 8080

¿Qué directiva en el archivo de configuración php.ini se encarga de indicar el nivel máximo de errores que se mostrarán en el navegador cuando haya fallos en el código?

a) error_reporting
b) e_reporting
c) error_level
d) e_level
e) e_error

¿Cuál de los siguientes valores para la directiva anterior hace que se muestren menos errores?

a) E_ERROR
b) E_WARNING
c) E_NOTICE
d) E_CORE_ERROR
e) E_USER_ERROR
f) E_ALL

¿Cuál de las siguientes librerías de acceso a MySQL desde PHP ya no se aconseja utilizar?

- mysql
- mysqli
- pdo

En el caso de que la siguiente instrucción se llegue a ejecutar ¿Qué devolverá como resultado?

```
$mysqli->query(
 "CREATE TABLE salarios( ".
     "id_persona INTEGER,".
     "valor NUMERIC(8,2))"
);
```

a) true
b) false
c) un objeto de tipo mysql_result
d) un array con la estructura de la tabla creada

En el caso de que la siguiente instrucción se llegue a ejecutar ¿Qué devolverá como resultado?

```
$mysqli->query(
    "SELECT * FROM salarios");
```

a) true
b) false
c) un objeto de tipo mysql_result
d) un array con la estructura de la tabla creada

¿Para qué sirve el método fetch_assoc?

a) Devuelve un array con todos los valores de una consulta
b) Devuelve un array con los valores de una fila de la consulta
c) Devuelve un objeto de tipo mysql_result
d) Devuelve el número de filas de una consulta

UNIDAD 7

SISTEMAS DE GESTIÓN DE CONTENIDOS

OBJETIVOS

- Identificar las ventajas de los CMS
- Reconocer la utilidad que aportan los CMS a la implantación de aplicaciones web
- Clasificar los diferentes tipos de CMS según su funcionalidad
- Reconocer los CMS comerciales más utilizados en la actualidad y las funciones que aportan
- Diferenciar las características de los principales CMS
- Seleccionar el CMS más apropiado en función de los requisitos de un determinado sitio web

CONTENIDOS

7.1 VENTAJAS Y CARACTERÍSTICAS DE LOS CMS

7.1.1 ¿QUÉ ES UN CMS?

CMS es la abreviatura del término *Content Management System,* **Sistema de Gestión de Contenidos**. Se trata de un software que trata de facilitar la creación y organización de los contenidos de uno o más sitios web. Su objetivo es paliar las dificultades que tiene crear sitios con numerosos contenidos en la web.

Si esos contenidos tienen que actualizarse a menudo, si además pueden ser creados por diferentes personas y, sobre todo, si queremos que puedan publicar directamente contenidos personas sin conocimientos técnicos, entonces claramente estamos obligados a utilizar un sistema de gestión de contenidos.

Con un CMS podemos, además, diferenciar entre los contenidos y la estructura del sitio web: menús, enlaces, almacenamiento físico, código, administración de usuarios, etc. Esta estructura es lo que se conoce como el **back-end**. Por otro lado, la visión que tienen los usuarios, tanto los que publican contenidos, como los que simplemente lo ven, se le conoce como **front-end**.

7.1.2 HISTORIA DE LOS CMS

Inicialmente los contenidos que aparecían en la web se trabajaban directamente desde el código de las páginas utilizando editores simples. Evidentemente, esta forma de trabajar impide poder crear sitios con gran cantidad de contenidos. Además, el proceso de añadir contenido se hace desde el código fuente, lo que pone en riesgo la estructura crítica de la aplicación. Lo que da lugar a que solo ciertos usuarios pudieran generar contenido.

La aparición de editores visuales como Adobe Dreamweaver permite concentrarnos en el diseño ocultando el código. Pero sigue siendo insuficiente.

La aparición de lenguajes en el lado del servidor dotó de la posibilidad de crear verdaderas aplicaciones que permitieran la gestión de usuarios y contenidos. Así el sitio de noticias **CNET** desarrolló su propio sistema de administración y publicación de contenidos en 1995. Ese mismo año, la empresa **Red Dot** crea el que es considerado el primer CMS comercial.

Poco a poco a finales de los 90 muchas empresas, especialmente de noticias, utilizaron sistemas de gestión de contenidos. A la vez aparecieron CMS tanto comerciales como de código abierto como **Allaire Spectra**, **Typo 3** o **Mambo.**

Quizá el primer CMS que obtuvo realmente éxito entre los creadores de aplicaciones web, fue **PHP-Nuke**. Se trata de un sistema de gestión de sitios de noticias muy utilizado a principios de este siglo.

El éxito de los CMS, realmente, empezó en torno al año 2007 cuando productos como **Joomla!**, **Drupal** o **Wordpress** maduraron y empezaron a utilizarse extensamente. En esto contribuyó enormemente el éxito de los blogs; sitios web donde los contenidos se organizan en base a su

fecha de publicación, formando una especie de ***cuaderno de bitácora***. Este tipo de sitios propició que personas sin conocimientos avanzados en las tecnologías web, pudieran crear contenidos de apariencia muy profesional y fácilmente con ayuda de estos CMS.

Poco a poco han ido mejorando las prestaciones de todos los CMS y ahora tenemos una extensa oferta tanto de sistemas de código abierto y gratuito como de sistemas de pago, que se adaptan a casi cualquier necesidad concreta.

7.1.3 VENTAJAS DE LOS CMS

- **Facilitan la generación de contenido**. Añadir un contenido a nuestro sitio web se realiza, en este tipo de sistemas, a través de sencillos formularios y menús. No hace falta tener conocimientos técnicos para hacerlo y, sobre todo, aun teniendo conocimientos técnicos es muy rápido añadir contenidos de esta forma.
- **Actualización de contenidos más rápida**. Tiene que ver con la ventaja anterior, como es fácil añadir contenidos y éstos se publican al instante, el resultado es una web muy dinámica donde los contenidos pueden cambiar muy a menudo.
- **Facilidad para determinar el diseño**. Hacer un sitio web con un diseño profesional es una tarea difícil. Los CMS proporcionan numerosas plantillas de aspecto profesional que se aplican rápida y fácilmente. Lógicamente la pega es la falta de libertad para diseñar.
- **Posibilidad de personalizar el entorno**. Esta es una capacidad que pretende aliviar el problema de la falta de libertad comentada en el apartado anterior. Al final, los CMS utilizan los lenguajes habituales de creación de aplicaciones web: HTML, CSS, JavaScript, PHP, etc. Por lo que, para aquellas personas con conocimientos avanzados, disponemos de la posibilidad de adaptar el diseño a nuestras necesidades.
- **Administración de usuarios**. Quizá es la ventaja fundamental. Los CMS están preparados para gestionar usuarios y organizarles en grupos, dotando a cada uno de diferentes capacidades. Incluso, en la mayoría de sistemas, podremos hacer que unos usuarios puedan publicar contenido en unas partes del sitio web y otros usuarios en otras.
- **Adaptación a la necesidad concreta**. Muchas veces los sitios y aplicaciones web de Internet ofrecen un servicio muy concreto: un blog, una plataforma educativa, una tienda on line, una revista o periódico, etc. La cuestión es que disponemos de CMS especializados para ese tipo de tareas: hay CMS que facilitan la escritura de blogs, CMS que sirven para crear sitios de tipo wiki, etc.
- **Mantenimiento de gran cantidad de documentos**. La administración y organización de sitios con grandes volúmenes de información es una de las capacidades fundamentales de este tipo de sistemas. No solo tendremos organizados los documentos en areas o secciones, sabremos quién y cuándo publicó el contenido y otra serie de meta datos que enriquecen el conocimiento que tendremos de cada contenido publicado.
- **Consistencia visual**. Ante una gran cantidad de páginas en un sitio es dificultoso mantener la misma apariencia visual y política estética de la empresa concreta. Los CMS

construyen un armazón común para los contenidos. Gracias a lo cual que no tendremos sensación de caos o de falta de profesionalidad.

- **Facilidad para añadir nuevas funcionalidades.** Casi todos los CMS disponen de la posibilidad de añadir, extensiones, plugins o componentes, en definitiva, que aportan nuevas funcionalidades. Podemos, fácilmente, incluir módulos realizados por otras personas y que aporten valor a nuestra aplicación

7.2 ESTRUCTURA DE UN CMS

7.2.1 VISTAS DE UN CMS

Como ya se ha comentado antes un CMS es un sistema de gestión de contenidos que presenta una estructura compuesta de dos visiones:

- **El back-end.** Visión relegada a los administradores. Mediante esta vista se generan los aspectos ocultos al usuario, pero fundamentales en la gestión del CMS. Entre ellos están:
  - La organización de grupos y usuarios
  - La estructura física de los archivos y directorios del CMS
  - La organización lógica de menús, enlaces y componentes
  - Tareas de limpieza
  - Copias de seguridad
  - Aspectos de posicionamiento en buscadores
  - Etc.
- **El front-end.** Visión del sitio que tienen los visitantes y también los usuarios registrados sin privilegios administrativos (editores de contenido). Desde esta vista se puede ver el resultado final de la aplicación web y también modificar el contenido (si tenemos permiso para ello).

7.2.2 ELEMENTOS DE UN CMS

Sea del tipo que sea, un CMS suele incorporar estos elementos para realizar sus labores:

- **Administrador de usuarios.** Vista, dentro del back-end, que nos permite crear, modificar y eliminar usuarios. También nos permite organizarles en grupos y, fundamentalmente, designar sus permisos y privilegios.
- **Editor WYSIWYG de contenidos.** WYSIWYG es el acrónimo de ***What You See Is What You Get*** (lo que ves es lo que obtienes) y en este caso se trata de que los contenidos se editen sin necesidad de tocar el código HTML. En su lugar, se usa un editor que permite modificar y examinar el contenido viéndolo tal cual aparecerá en la página. No obstante, la mayoría de editores permiten también modificar el código HTML por si deseamos tener más control sobre él.

- **Editor de menús, categorías o jerarquía del sitio**. Es parte del back-end y con él generamos la estructura organizativa. Sin duda es uno de los aspectos más determinantes, clave de la facilidad de uso del sitio que estamos construyendo. Además, en muchos CMS, podemos hacer que ciertas secciones de la organización sean editables por ciertos usuarios y otras no.
- **Editor y administrador de temas**. Es el componente que se encarga de gestionar los temas; es decir, la apariencia visual del entorno. Suele encargarse de elegir un tema y de dotar las herramientas para crear temas propios o bien modificar o personalizar los existentes.
- **Administrador de extensiones y plugins**. Encargado de añadir, modificar o eliminar componentes externos al sitio web.
- **Gestor de redes sociales**. Cada vez más presente en el CMS. Permite que los contenidos puedan ser fácilmente enlazables a las redes sociales, o bien autentificar mediante las cuentas de las redes más populares a los usuarios que quieran hacer comentarios, etc.
- **Gestor de taxonomías**. Permite gestionar los temas y palabras claves de los contenidos. De esta forma, podremos hacer búsquedas en base a esas palabras para encontrar contenidos relacionados con ellas.

En definitiva, un CMS aporta numerosas herramientas para cubrir todo lo que se necesita en la gestión de un sitio web donde hay mucho contenido.

7.2.3 TECNOLOGÍA SUBYACENTE EN LOS CMS

Un CMS no es más que una aplicación web que estará creada en una determinada tecnología de servidor y de cliente. La mayoría están creados para un servidor web concreto. Los datos que guarda el CMS es la base de su funcionamiento porque ahí estará la lista de usuarios, los contenidos, metadatos, etc.

Esto implica que para implementar un CMS se nos exigirá tener al menos un servidor de aplicaciones y otro de base de datos; y además se nos exigirá un producto concreto.

La mayoría de CMS están construidos en PHP. Ya hemos visto que es un lenguaje muy poderoso pero también que no facilita precisamente la escritura estructurada de código, por lo que para programadores poco experimentados, puede producir un código final poco eficiente. Evidentemente los principales CMS tienen un código de calidad. Además de en PHP, hay CMS construidos en **Ruby on Rails, Python, .NET, Java, ColdFussion** o **Perl**.

En cuanto al sistema de bases de datos, es absolutamente dominante **MySQL**. Pero hay bastantes CMS que utilizan **SQL Server** (especialmente los que son de pago y creados para la plataforma .NET de Microsoft), **Oracle, PostgreSQL, SQLite** o **DB2**.

El resultado final, como en toda aplicación web, son instrucciones HTML, CSS y JavaScript que cualquier navegador puede traducir y mostrar. Así pues, estamos ante una aplicación típica de tres niveles (véase 1.5.3 *"Arquitectura de tres niveles"*, en la página 25). En la que cabe destacar

que la capa de presentación se ha ido renovando en casi todos los niveles para dar cabida al nuevo estándar HTML5 y a las librerías JavaScript más populares como **jQuery**.

En general, la arquitectura de funcionamiento de un CMS se puede esquematizar de esta forma:

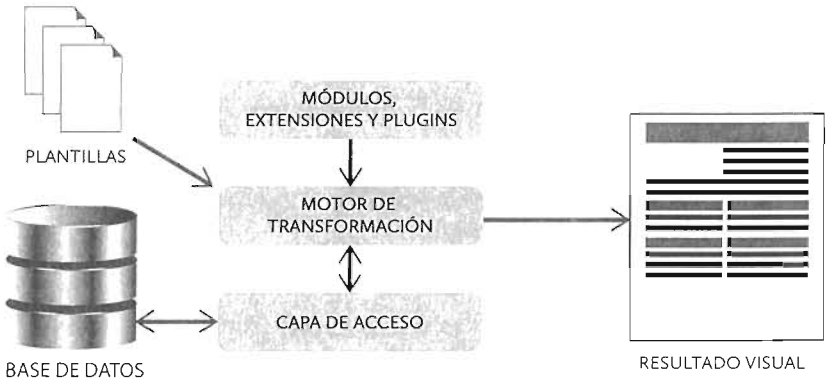


Figura 7.1: Funcionamiento de un CMS

El **núcleo** de un CMS es el motor de transformación. Es el componente de la aplicación que, utilizando los contenidos que están en la base de datos y a los que se accede a través de una capa de acceso, se les da formato y organización utilizando los módulos/plugins/extensiones instalados y la plantilla o plantillas utilizadas. Además se debe utilizar la información organizativa que también está en la base de datos. Con todo ello, el motor genera lo que el usuario verá, una página web con los contenidos debidamente formateados.

7.3 TIPOS DE CMS

Hay que tener en cuenta que hay CMS que son de código abierto y además gratuitos y CMS que son de pago. La mayor parte de los CMS de pago aportan soporte por parte del fabricante y valores añadidos a los CMS de código abierto. Al ser productos comerciales trabajan mucho la apariencia y la robustez.

No obstante, los CMS de código abierto poseen una comunidad de desarrolladores y usuarios enorme que continuamente realizan mejoras al producto, con lo que no solo no tienen mucho que envidiar a los de pago, sino que incluso les superan en muchos aspectos.

Al final la elección del CMS tendrá que ver con la necesidad concreta que tengamos y el presupuesto. Hay CMS que solucionan muy bien ciertos aspectos, pero no otros, por lo que lo mejor es estudiar el que más nos interese.

Por otro lado, hay que tener en cuenta que, puesto que un CMS es una aplicación web, podemos crearla nosotros mismos (eso sí con mucha pericia) o bien pedir a una empresa que nos cree un CMS a medida. Eso es lo que ocurre, por ejemplo, en los grandes sitios de noticias donde decenas de periodistas crean contenidos que rápidamente aparecen, y en la ubicación adecuada, en la web. Crear un CMS por parte de la propia empresa solo está a disposición de empresas con

un departamento de desarrollo de aplicaciones. Crear un CMS a medida también implica poseer un importante presupuesto para esta materia.

El presupuesto es un factor clave y teniendo en cuenta que las necesidades de la mayoría de empresas y entidades no suelen ser excesivas, un CMS de código abierto es una excelente opción.

Hay CMS que incluso tienen una versión gratuita, con funcionalidad más limitada, pero a veces suficiente para nuestras necesidades, y otra de pago con más funciones posibles.

7.3.1 CMS DE PROPÓSITO GENERAL

Se trata de los CMS en los que se suele pensar inmediatamente al hablar de este tipo de software. Se dedican a todo tipo de sitios, tengan la temática que tengan. Se les llama también CMS para portales.

Su objetivo es implementar webs donde varios tipos de usuarios generan contenido y la estructura de la publicación de contenidos es muy jerárquica e implica una administración avanzada. Revistas, periódicos, sitios corporativos, entidades públicas, sitios orientados a publicar eventos y, en definitiva, sitios donde predomina la información estructurada en temas y categorías, en la que ciertos usuarios pueden escribir en ciertas partes, pero no en otras.

7.3.1.1 JOOMLA!

Es gratuito y de código abierto y muy utilizado en docencia como ejemplo fundamental de CMS. Se ha creado con PHP, MySQL y Apache, lo que le hace cómodo para modificar por parte de muchos desarrolladores.

Permite, control de usuarios, una buena administración y gestión del back-end, una buena interfaz para publicar en el front-end, numerosos plugins y mejoras,etc. En definitiva, hay poco que le podamos pedir a un CMS que Joomla! no aporte.

Ha tenido problemas serios de seguridad en algunas versiones que, aunque se han ido solucionando, le han causado cierta mala fama.

Su virtud es la facilidad de uso, que permite crear sitios complejos rápidamente. Quizá el cuello de botella (comparado especialmente con Drupal, su principal rival) es la dificultad para hacer sitios escalables.

Todo esto en teoría, porque Joomla! es el CMS con el que se han muchas webs de alto nivel en Internet: la cadena MTV, Barnes and Noble, Aeropuerto de Heathrow, La torre Eiffel, etc.

7.3.1.2 DRUPAL

Es el tercer CMS más popular en número de sitios, sin embargo, domina en sitios corporativos de gran tamaño e instituciones de alto nivel. Cada vez se está implantando más y está muy cerca de superar a Joomla! Tiene una filosofía un tanto diferente de Joomla! En este caso la facilidad de uso es menor, es más difícil gestionar un sitio con Drupal al principio.

Pero la virtud de Drupal es la escalabilidad. Podemos definir sitios mucho más versátiles y con muchas más posibilidades. Como ocurría con Joomla!, no hay tarea que deseemos de un CMS que no esté presente en Drupal. Incluso podemos ir más allá, ya que Drupal proporciona un marco de trabajo para desarrolladores que permite personalizar totalmente el resultado final y que además está construido de una forma muy atractiva para los desarrolladores.

Al igual que Joomla! está creado en PHP y la base de datos suele ser MySQL, aunque se admiten PostgreSQL o SQLite.

Ejemplos de sitios que usan Drupal son: Twitter, Pinterest, El gobierno de EEUU, la cadena NBC, Box.com, Nokia.com o Tesla.

7.3.1.3 CMS MADE SIMPLE

La filosofía de este CMS, también gratuito y de código abierto, queda clara por su propio nombre. Intenta ser el CMS más sencillo de utilizar. Es difícil determinar si es tan sencillo, pero parece clara que su virtud es la flexibilidad y fácil adaptación para crear sitios dinámicos. Como los anteriores está programado en PHP y MySQL.

7.3.1.4 TYPO3

Fue uno de los primeros CMS y sigue siendo bastante usado. Es de código abierto, creado para PHP y MySQL. Proporciona las mismas posibilidades que los anteriores, pero tiene unas extensiones menos logradas. Muchas universidades alemanas, la empresa Bayer, Loewe y otras muchas, utilizan este CMS. La FAO, Konika Minolta y Volkswagen son ejemplos de webs creadas con la ayuda de Typo3.

7.3.1.5 PLONE

Este CMS también es veterano y, a diferencia de los anteriores, se ha creado en Python. Presume de facilidad de uso y rapidez. Como los anteriores posee todas las características exigibles a un CMS, pero dispone de muchas menos plantillas y extensiones. La Universidad Humboldt de Berlín, Correos de Brasil y ACM son ejemplos de webs que usan Plone.

7.3.2 ORIENTADOS A BLOGS

Se trata de CMS especializados principalmente en la publicación de blogs. Los blogs están, por lo general, mantenidos por particulares y son simplemente una serie de entradas que se ordenan cronológicamente haciendo que se muestren primero las entradas (llamadas comúnmente post) más reciente. También es cada vez mas habitual utilizar blogs corporativos.

Muchísimas de las web que hay en Internet son blogs, tuvieron un éxito impresionante en la década pasada y todavía ahora se considera una de las formas comunicativas más importantes de Internet. Normalmente los usuarios del blog no solo leen las entradas, sino que se les permite publicar para comentar dichas entradas.

Entre las necesidades que se le pide a un CMS especializado en blogs, están:

- Publicar post, entradas en el blog.

- Capacidad de establecer categorías, así como de especificar metadatos en las entradas que faciliten su búsqueda y ordenación.
- Añadir elementos multimedia de todo tipo
- Poder buscar entradas según el tema o por la fecha.
- Cambiar la apariencia estética fácilmente
- Exportación del contenido a otros formatos, por ejemplo RSS.

7.3.2.1 WORDPRESS

Indudablemente se trata del CMS más exitoso. Utilizado inicialmente solo para construir blogs, su facilidad de uso, orientación semántica, el hecho de ser código abierto construido en PHP, respeto a los estándares, así como su extensa y dinámica comunidad de usuarios son quizá las claves de su éxito.

En la actualidad ha ampliado sus capacidades y ya se considera también un CMS para crear todo tipo de sitios web, con lo que compite directamente con Joomla! o Drupal. Es también, sin duda, el CMS que posee más extensiones y plantillas para crear las páginas web. Además, tiene también grandes posibilidades para ser utilizada como plataforma de creación de aplicaciones web.

Según las estadísticas de la página w3tech.com, una de cada cuatro páginas de Internet está creada con WordPress. No solo se utiliza en el ámbito personal, grandes sitios como el sitio web de la revista New Yorker, Sony Music, Google Ventures, Mark & Spencer, o la página social de Ford, por poner unos ejemplos, utilizan WordPress. En el caso de los blogs corporativos, se usa casi unánimemente.

7.3.2.2 BLOGGER

Actualmente es propiedad de Google. No es un software CMS que requiera instalación, sino que es un servicio de alojamiento de blogs on line. Es competidor, como servicio, de Wordpress aportando facilidad de uso y, sobre todo, al ser parte de la red social de Google, facilidad de creación.

Tiene éxito en blogs de tipo personal. En todo caso es una buena opción para crear un primer blog de forma sencilla.

7.3.2.3 MOVABLE TYPE

Mucho menos conocido que los anteriores, es un CMS que apareció en el año 2001 (fue de los primeros) y está creado en lenguaje Perl aunque utiliza la habitual base de datos MySQL. Se le reconoce como el principal oponente de WordPress.

Es gratuito para la creación de blogs personales, pero se paga si el uso es comercial o institucional. Es muy fácil de utilizar y contiene todo lo que se podría pedir a este tipo de productos.

Tiene menos facilidad para la personalización así como un catálogo mucho menor de plantillas que WordPress.

7.3.2.4 EXPRESSIONENGINE

Creado en lenguaje PHP y utilizando MySQL, es un CMS que se organiza en canales, aportando una gran capacidad para la administración de usuarios. Es gratuito y de código abierto y está construido utilizando **CodeIgniter**, plataforma de código abierto creada por la misma empresa (**Ellis Lab**) propietaria de ExpressionEngine. CodeIgniter tiene mucho éxito entre los desarrolladores PHP por la facilidad y potencia que aporta para crear aplicaciones web avanzadas.

Su punto fuerte es la construcción de la apariencia del sitio a partir de unas potentes plantillas PHP, fácilmente personalizables. Presume de una gran potencia y facilidad de personalización del resultado final. Es una clara alternativa a WordPress. Es un CMS gratuito en su versión básica, pero de pago cuando aumentan sus prestaciones. Tiene opción de soporte por un pago mensual. Ford, Disney o Nike figuran entre sus clientes.

7.3.3 ORIENTADOS A COMERCIO ELECTRÓNICO

Facilitan la creación de tiendas en línea. Por lo que su orientación se basa en la administración de productos, clientes, categorías, precios y, sobre todo, gestión de carrito de la compra. EL objetivo en definitiva es facilitar la venta de productos a través de Internet.

7.3.3.1 MAGENTO

CMS de código abierto, actualmente propiedad de eBay, que apareció en 2011. Está creado en PHP y usa como base datos a MySQL. Es el cuarto CMS más popular de Internet tras Wordpress, Drupal y Joomla!, lo que le convierte en la solución más utilizada como gestor de contenidos de comercio electrónico.

Se le reconoce una gran robustez y seguridad debido en parte a su modelo de datos (**EAV**) y a los requisitos que impone en su implementación. Es muy interesante, a nivel comercial, la capacidad de integrarse con software de tipo **ERP** (sistemas de planificación de recursos).

Sus críticos apuntan que no es un CMS muy rápido (debido a la complejidad de su modelo de datos). Su código fuente también es enormemente críptico. Posee una versión gratuita (***Community Edition***) y una versión comercial (***Enterprise Edition***) con todas las funcionalidades de esta plataforma.

7.3.3.2 PRESTASHOP

Es el competidor más importante de Magento. Cada año aumenta su número de usuarios, por lo que poco a poco se va acercando a los niveles de Magento. Es gratuito y de código abierto y está construido en la típica tecnología PHP+MySQL, aunque en el lado del cliente utiliza intensivamente AJAX.

Su instalación e implantación es muy sencilla y consume pocos recursos. Por sus características, es un producto que tiene una comunidad muy dinámica, por lo que encontrar documentación es muy fácil.

7.3.3.3 OSCOMMERCE

Ha sido durante bastante tiempo el CMS para sitios de comercio electrónico más popular. Sin embargo, los dos anteriores le han superado en número de usuarios. Gracias a su veteranía dispone de numerosas extensiones y plantillas, aunque no todas se integran fácilmente. Las críticas vienen de su sensación de falta de seguridad y estabilidad; pero, sobre todo, de haberse quedado detrás en innovación respecto a sus competidores.

7.3.3.4 OPENCART

Es otra opción semejante a las anteriores que ha ganado bastante popularidad en los últimos años. Sus valores son la sencillez, rapidez y escasos recursos requeridos. Compite directamente con PrestaShop como solución para empresas de tamaño pequeño o medio (las empresas de tamaño alto parecen estar monopolizadas por Magento).

7.3.4 ORIENTADOS A SITIOS WIKIS

Un sitio wiki está formado por una gran cantidad de documentos web construidos por la colaboración de los propios usuarios del sitio. El ejemplo más famoso de página Wiki es la Wikipedia, enciclopedia con cientos de miles de artículos construidos por miles de editores, que no son más que usuarios de la misma.

Las wikis aportan rapidez y son una interesante herramienta educativa para fomentar el aprendizaje colaborativo. Los CMS especializados en este área aportan facilidad de creación de contenidos, así como herramientas avanzadas para la revisión de los contenidos que requiere además de varias categorías de usuarios para conseguir que algunos tengan más poder, sobre todo, para el caso de que se discuta la veracidad de los contenidos.

7.3.4.1 MEDIAWIKI

Software de código abierto escrito en PHP y que permite utilizar MySQL, SQLite o PostgreSQL como base de datos. Es el software con el que se ha implementado la wikipedia, por lo que el aspecto es muy familiar para todos los usuarios y no requiere aprendizaje de manejo por parte de los usuarios que visitan sitios creados con esta herramienta.

7.3.4.2 DOKUWIKI

Alternativa a la anterior que se sitúa como segunda opción en el mercado para implementar aplicaciones wiki. Está escrita en PHP pero utiliza ficheros planos para almacenar datos; es decir, no utiliza un servidor de base de datos para guardar los contenidos. Esto le permite requerir únicamente un servidor PHP para funcionar, por lo que su implementación es extremadamente sencilla. Posee plantillas para extender su funcionamiento y una gestión de usuarios basada en listas de control de acceso. Fuera de la popularidad del motor de la Wikipedia es la opción más utilizada.

7.3.4.3 TIKIWIKI O TIKI

Aunque se orienta a las páginas wiki, realmente aporta características suficientes para ser considerado como una opción para crear sitios de todo tipo, por lo que se asienta en una zona más cercana a WordPress, Joomla! o Drupal. En todo caso su número de usuarios va en claro descenso y su elección como motor de wikis está claramente marcada si deseamos que nuestra página sea algo más que una página wiki.

Es también una solución gratuita y abierta creada en PHP con base de datos MySQL.

7.3.5 ORIENTADOS A FOROS DE DEBATE

Los foros de debate son sitios web donde los usuarios discuten sobre un tema. Los usuarios publican comentarios que el resto de usuarios contestan. Suelen agrupar los hilos de discusión para que sea más fácil la navegación, así como etiquetar con palabras clave cada hilo a fin de poder buscarlos con más facilidad. Es fundamental la figura de un moderador capaz, entre otras cosas, de vetar a usuarios que no aportan cosas positivas al foro.

Los CMS para crear foros de debate además de incorporar herramientas para él establecimiento de un orden jerárquico en los comentarios, aportan la posibilidad de que los usuarios puedan crear mensajes privados, posibilitar que los usuarios se suscriban a temas concretos o realizar de estadísticas y encuestas sobre los contenidos alojados.

Indudablemente pueden ser una buena herramienta educativa o utilizarse como sitio web orientado a comentarios sobre productos o cultura: libros, música, cine, etc.

Muchos CMS de este tipo utilizan el lenguaje etiquetado BBCode, para los comentarios del foro. Es un lenguaje especialmente pensado para foros de debate que se parece a HTML, pero que facilita la creación de formato en estas páginas. No es traducible por un navegador, por lo que es el propio CMS el que le traduce al HTML correspondiente.

7.3.5.1 PHPBB

Software de código abierto escrito en PHP y que permite usar MySQL, Oracle, PostgreSQL o SQLite como base de datos. Se mantiene desde hace años como líder dentro de los CMS para foros de código abierto.

Posee todas las características deseadas: organización de foros y subforos, posibilidad de mensajes privados, búsqueda avanzada, organización avanzada de usuarios, agrupación por temas de los comentarios, etc.

7.3.5.2 SIMPLE MACHINES FORUM

Escrito en PHP, utiliza MySQL como base de datos. Es semejante en prestaciones y capacidades al anterior del que es claro competidor. Hace más hincapié en la sencillez.

7.3.5.3 VBULLETIN

Software de pago para crear foros de debate creado en PHP y con MySQL como sistema de bases de datos. Es el CMS más popular, más que su inmediato rival phpBB. Ofrece la posibilidad de ser utilizado en la nube en lugar de instalar el sistema en nuestros servidores. Su clara desventaja es el precio, pero a cambio presume de estabilidad, seguridad y un gran control de la moderación de los foros. Consigue webs optimizadas para los dispositivos móviles (algo en lo que sus rivales también van mejorando).

7.3.5.4 DISCUZ!

CMS creado en PHP para MySQL, actualmente propiedad del gigante chino **Tencent**, es el CMS de grupos de debate que más crece. Dispone de una versión gratuita y otra de pago. No tiene versión oficial en otro idioma que no sea el chino, pero eso no ha minado su creciente popularidad.

7.3.6 ORIENTADOS A APRENDIZAJE EN LÍNEA

Se les conoce también como herramientas de eLearning y también se les llama **LMS** (Learning Management System, sistema de administración de aprendizaje). Permiten crear cursos on line en los que los usuarios se apuntan y se les puede monitorizar y asignar recursos de aprendizaje.

Entre sus labores está la gestión de cursos, profesores y alumnos, administración y publicación de material educativo, gestión de agendas de trabajo, evaluaciones, tareas, etc. También suelen incorporar herramientas de colaboración entre estudiantes, especialmente foros de debate y cuadros de anuncio.

7.3.6.1 MOODLE

Es el CMS más conocido para crear cursos en línea. Incorpora todos los elementos comentados para esta labor. Su instalación es sencilla y dispone, gracias a su popularidad, de numerosos plugins y plantillas. Admite importar cuestionarios de todo tipo de formatos. Está creado también en PHP con base de datos MySQL. Es de código abierto y gratuito.

7.3.6.2 DOKEOS

Lejos todavía de la popularidad de Moodle, es una clara opción ya que ofrece prácticamente las mismas características. Incorpora algunas funciones extra comparado con Moodle, pero tiene menos plugins y extensiones, debido a que su comunidad de usuarios es más pequeña.

7.3.6.3 CANVAS

Es un LMS creado en **Ruby on Rails** con PostgreSQL como base de datos, propiedad de la empresa Instructure. Es un software propietario, pero podemos utilizarle gratuitamente para crear cursos si somos un institución educativa. El código ha sido recientemente liberado.

Sin embargo, la solución conocida como Canvas Network es un servicio en la nube orientado a crear **MOOCs**, ***Massive Online Open Courses*** cursos masivos abiertos de aprendizaje, cursos

en la nube en los que pueden apuntarse miles de usuarios. La creación de cursos en esta herramienta on line es gratuita para instituciones académicas. Es un software potente, fácil de manejar y administrar que, además, se utiliza por parte de numerosas entidades.

7.3.6.4 BLACKBOARD

Es el software de pago más popular para crear cursos on line. Es un LMS fácil de utilizar, con una gran apariencia creado en Java y que utiliza como bases de datos Oracle o SQL Server. Se jazta de su robustez y gran capacidad de gestión como sistema gestor de herramientas de aprendizaje completa.

7.3.6.5 OPEN EDX

Versión de código abierto del software que utiliza la popular plataforma de cursos de tipo MOOC EdX auspiciada por las Universidad de Harvard y el MIT. La idea es que sea utilizada por instituciones académicas.

7.3.7 ORIENTADOS A LA COLABORACIÓN

Se trata de CMS pensados, fundamentalmente, para labores internas de una empresa con la finalidad de facilitar la colaboración entre los trabajadores. Su campo principal de trabajo es, pues, las Intranets. Se les llama también **EMS** (***Enterprise Managemente System***, sistemas de administración de contenidos empresariales)

Ayudan a gestionar flujos de trabajo, grupos y usuarios, documentación colaborativa, contactos y comunicación con sistemas ERP (sistemas de planificación de recursos).

7.3.7.1 ALFRESCO

Se trata de un software empresarial que tiene un versión Community, gratuita y de código abierto. Tiene capacidad para gestionar todo el contenido empresarial, así como organizar y publicar contenidos en un sitio web. Es una solución completa que, en su versión propietaria, ofrece capacidades completas para hacer toda la gestión empresarial. Está creado en Java. Genesys, NHS Education, la página de la PGA (asociación de golf), la cadena Fox o el grupo Amadeus son ejemplos de empresas que utilizan este CMS.

7.3.7.2 NUXEO

Es un software ya veterano que está creado en Java y que hace especial hincapié en crear aplicaciones internas para la gestión documental y la colaboración digital entre trabajadores.

7.3.8 ORIENTADOS A LA CREACIÓN DE GALERÍAS

Son CMS que facilitan la creación de galerías fotográficas o de otro tipo. La idea es crear álbumes fotográficos o de vídeo en línea. La idea es la publicación de fotografías y vídeos por parte de los usuarios haciendo que otros usuarios puedan comentar o evaluar ese material. Es habitual que se permita crear zonas privadas donde solo ciertos usuarios pueden acceder, lo que implica una labor de administración de usuarios que el software tiene que permitir.

7.3.8.1 GALLERY

Es de código abierto, nuevamente, se creo en PHP usando como sistema de base de datos a MySQL. Actualmente el proyecto se ha cerrado, aunque ha través de foros podemos seguir teniendo soporte. Ha sido, indudablemente, el CMS para galerías de fotos más popular.

7.3.8.2 DRAGONFLY CMS

CMS de construcción de galerías fotográficas, basado en PHPNuke, uno de los primeros CMS construidos en PHP. Es todavía muy popular. Es sencillo de implementar pero su apariencia no da sensación de modernidad.

7.3.8.3 COPPERMINE GALLERY

Como los anteriores usa PHP y MySQl y es de código abierto y gratuito. Es quizá el más robusto y de aspecto más agradable. La gestión no es precisamente sencilla, pero también aporta comunicación con muchos de los CMS más populares como Drupal, Joomla!, Simple Machines Forum, phpBB o vBulletin.

ACTIVIDAD 7.1:

- W3Techs.com es una reputada web de estadísticas sobre tecnologías de aplicaciones web.
- Instala el plugin w3tech para Firefox o Chrome. Navega por algunas de las páginas sugeridas en los apartados anteriores, observa que el plugin te dice todo lo que sabe de ella, incluido el CMS que usa.
- Un plugin parecido, aunque más visual, está disponible en la URL: https://wappalyzer.com/download
- Compara ambos plugins para ver lo que ambos te dice en cada web que visites.

7.4 ELECCIÓN DEL CMS

7.4.1 POPULARIDAD

Productos hay muchísimos y muchos especializados para tipos de aplicaciones concretas. No obstante, muchos CMS se salen de su categoría, es posible con WordPress hacer sitios web que no sean blogs (de hecho es lo habitual hoy en día) y, por ejemplo, Alfresco permite crear sitios de todo tipo, no solo de colaboración empresarial.

Los tres CMS a nivel general más populares son WordPres, Joomla! y Drupal. Pero su nivel de implementación es muy desigual. Según los datos de W3techs, el 40% de los sitios web de Internet utiliza algún CMS. Esto da una idea del éxito de este tipo de software. Pero de los CMS que se utilizan en Internet la cuota se reparte de esta forma:

- 60,2% WordPress. De hecho el número total de páginas creadas con WordPress en Internet es del 24,1%

- 7,1% Joomla!. Es el segundo CMS más popular, pero a una gran distancia de WordPress
- 5,2% Drupal.
- 2,9% Magento. Primera solución comercial, orientada a las tiendas en línea
- 2,8% Blogger
- 1,6% Typo3

Y así hasta una infinidad de CMS todos ellos con cuotas del 0,5% o el 0,6%. La popularidad de nuestro CMS es un factor clave, porque si mucha gente lo utiliza tendremos el soporte de la propia comunidad de usuarios, que se muestra día a día más importante para tener una experiencia grata con el producto.

ACTIVIDAD 7.2:

- En la dirección http://w3techs.com/technologies/overview/content_management/all puedes ver la estadística completa de uso de CMS en Internet.
- Haciendo clic, en esa página, sobre un producto concreto, se detalla el uso por cada versión del software, gráficas sobre el uso histórico o posición en el mercado según el tipo de sitio.

7.4.2 PRECIO

Una de las grandes virtudes de los sistemas CMS es que hay muchas opciones, y de gran calidad, totalmente gratuitas. Las empresas se suelen plantear implantar sistemas de pago cuando el soporte es un factor crítico, porque la aplicación construida a través del CMS es básica para nuestro negocio.

En ese sentido las tiendas on line no suelen arriesgar y por eso en ese sector los sistemas de pago, como Magento, tienen más éxito. Otras acuden a software de reputada fama por la empresa que lo gestiona, como es el caso ,por ejemplo de **Adobe Expression Manager**, que cuenta con el aval y la experiencia de la empresa Adobe, aunque el precio del producto sea más caro.

7.4.3 TIPO DE NECESIDAD

Esto es lo fundamental. Si lo que queremos es un blog de noticias para dar a conocer las novedades en los productos de una empresa, es absurdo utilizar CMS como Drupal que aportan características muy potentes, pero con el que costaría echar a rodar este tipo de servicio. Lo lógico es utilizar CMS más orientados a esta necesidad.

Realmente necesitamos una fase de análisis detallado porque las cosas no son tan sencillas. Tenemos que decidir detalles sobre la audiencia del sitio, si va a tener que colocar contenidos de todo tipo, si solo es texto, si necesita foros de debate, si algunos usuarios utilizarían chat, si el diseño es fundamental,etc. Estos detalles (y por supuesto muchos más) nos irán empujando hacia un CMS concreto.

7.4.4 FACILIDAD PARA LA PERSONALIZACIÓN

Hay productos muy sencillos que se instalan rápido y con los que podemos a empezar a funcionar casi al instante. Pero en muchas ocasiones necesitaremos ir más allá de lo que el CMS aporta por defecto, para crear nuestros propios elementos, más allá de las plantillas, extensiones o componentes presentes en el CMS. Si este es un factor crítico, empezaremos a acercarnos a CMS que nos permiten utilizarles como marcos de trabajo para crear aplicaciones web absolutamente personales.

En los grandes sistemas CMS esto es perfectamente posible. Drupal es especialmente famoso por este hecho, pero también muchos otros como WordPress y Joomla! aportan posibilidades en este sentido.

7.4.5 EXPORTACIÓN

Realmente al final la decisión va depender de factores muy personales. Además, al principio no tendremos la experiencia de conocer cuáles son los problemas de cada CMS. Puede ocurrir que con el tiempo deseemos cambiar de CMS.

La cuestión es que si utilizamos otro, ¿qué pasa con el contenido que hemos hecho? Es bueno que conozcamos de antemano las posibilidades de exportar la información. Desgraciadamente este es un factor poco cuidado por parte de los CMS, pero es crítico, ya que puede anclarnos al mismo sistema indefinidamente al no poder permitirnos perder el contenido.

No obstante hay que recordar que la mayoría de CMS usa MySQL como base de datos, estudiando sus tablas podremos obtener el contenido. Es una labor difícil, pero posible.

ACTIVIDAD 7.3:

- La página http://www.opensourcecms.com/ permite ver una demostración visual sobre la mayoría de CMS.
- Entra en la página y pulsa en **All CMS Demos**, después haz la prueba de funcionamiento en al menos tres de los CMS que se han nombrado en esta unidad.
- Observa tanto la página principal (botón Demo Main Page) como la de administración (Demo Admin Page)

7.5 CREAR NUESTRO PROPIO CMS

Nada nos impide crear un sistema propio que se adapte a la necesidad concreta. Evidentemente, la dificultad está en que tendremos que crear un sistema que, desde luego, es difícil de programar.

Solo empresas de gran tamaño con un departamento de desarrollo muy bien formado sería capaz de acometer algo así. Aunque tampoco se trata de crear un software tan complejo como WordPress o Drupal, sino algo mucho más sencillo que se adapte a una necesidad más completa. Eso sí, todo el mantenimiento, arreglo de errores y responsabilidad corre de nuestra parte.

7.6 RESUMEN DE LA UNIDAD

- Los CMS pertenecen a una categoría de software orientada a facilitar y organizar sitios web que exponen gran cantidad de contenido.
- Permiten definir el **front-end** (visión que los usuarios tienen del sitio web) y el **back-end** (estructura del sitio: usuarios, menús, apariencia, etc.)
- Aportan numerosas ventajas:
  - Facilidad para generar contenido.
  - Actualización rápida de contenidos.
  - Personalización del entorno.
  - Administración de usuarios.
  - Mantenimiento del contenido por grande que sea.
  - Apariencia profesional y consistente.
- Tienen un panel de control de gestión del back-end y herramientas que facilitan la escritura del contenido.
- Las tecnologías fundamentales que utilizan los CMS son Apache+PHP+MYSQL. Pero hay todo tipo de tecnologías que usan.
- Hay numerosos tipos de CMS:
  - De propósito general. Como Drupal y Joomla!
  - Orientados a blogs. Como WordPress o Blogger.
  - Orientados a comercio electrónico: Como Magento, Prestashop u OSCommerce.
  - Orientados a sitios Wikis.
  - Orientados a formación (LMS). Como Moodle.
  - Orientados a fotos: Como PHPBB o Simple Machines.
  - Colaboración empresarial: Como Alfresco o Nuxeo.
  - Etc.
- La elección de un CMS tiene en cuenta aspectos como la popularidad (por el soporte), precio, necesidad final, personalización y facilidad de exportación.
- Además, disponemos de la opción de crear un CMS propio para adaptarlo a nuestras necesidades.

7.7 TEST DE REPASO

¿Cuáles de estos elementos de un sitio web formarían parte del back-end?

- Estructura de los menús
- Artículos publicados
- Comentarios de los usuarios
- Organización de los usuarios
- Estructura física de los archivos

¿Qué tecnologías son las más dominantes en los CMS actualmente?

- PHP y MySQL
- Ruby on Rails y SQLite
- Java y Oracle Database
- JavaScript y MongoDB

3. ¿Cuál de las siguientes no es una ventaja de un CMS?

- Facilidad para administrar usuarios
- Facilidad para establecer la estética del sitio
- Mínimo tamaño de carga de las páginas finales
- Consistencia visual
- Facilidad para añadir nuevas funcionalidades

4. ¿A cuál de los siguientes CMS no se le considera de propósito general?

- Joomla!
- Drupal
- WordPress
- CMS Made Simple
- Plone

5. ¿Cuál de estos CMS no está orientado al comercio electrónico?

- OSCommerce
- Open Cart
- ExpressionEngine
- PrestaShop
- Magento

¿Cuál de las siguientes no es una característica de los sitios Wiki?

- Rapidez en la generación de contenidos
- Generación colaborativa de contenidos
- Estética semejante a la Wikipedia
- Facilidad para crear contenidos

7. ¿A qué están orientados los CMS Moodle, Dokeos y Canvas?

- A crear sitios wiki
- A crear foros de debate
- A crear sitios de aprendizaje en línea
- A crear blogs

8. ¿Qué son los EMS?

- Sistemas de gestión de contenidos empresariales
- Sistemas de gestión de contenidos con licencia privada
- Sistemas de gestión de contenidos extendidos
- Sistemas de gestión de contenidos para e-commerce

9. ¿Cuál es el CMS más utilizado en la actualidad?

- WordPress
- Joomla!
- Drupal
- Magento

10. ¿Cuál de los siguientes aspectos no es fundamental para elegir un CMS?

- Popularidad
- Precio
- Facilidad para la personalización
- Tipo de necesidad

11. ¿Cuál es el mejor CMS?

- Joomla!
- Drupal
- WordPress
- Depende del sitio que queramos crear

UNIDAD 8

INSTALACIÓN DE SISTEMAS DE GESTIÓN DE CONTENIDOS

OBJETIVOS

- Reconocer los elementos fundamentales de un CMS utilizando a WordPress y Drupal como ejemplos
- Reconocer los requisitos de instalación de un CMS tomando como ejemplo a WordPress y Drupal
- Distinguir entre las diferentes maneras de instalar un CMS
- Crear la base de datos en la que se almacenan los datos del CMS y el usuario con el que se enlazará con la base de datos
- Instalar los CMS Drupal y WordPress asimilando las distintas formas y pasos necesarios
- Establecer pautas que mejoren la seguridad de la instalación
- Identificar la forma de acceso y utilidad de los paneles de administración

CONTENIDOS

8.1 CARACTERÍSTICAS DEL CMS WORDPRESS

8.1.1 INTRODUCCIÓN A WORDPRESS

Como ya se comentó en la unidad anterior WordPress es el CMS más popular. Además es gratuito y de código abierto, lo que le convierte en una de las primeras opciones a considerar por todo tipo de usuarios y entidades, para ser utilizado como el sistema de gestión de los contenidos web.

WordPress está construido en PHP y utiliza MySQL como sistema gestor de los datos. Al ser tecnologías tan populares para los desarrolladores web, dispone de una gran comunidad de desarrolladores que continuamente trabajan en mejoras para este entorno. Además, para ellos aporta una API (interfaz de programación) sencilla y potente con la que se puede modificar todo el funcionamiento de WordPress.

Comparado con sus rivales, **Drupal** o **Joomla!**, WordPress no parece tener la misma potencia para gestionar sitios web complejos o muy grandes. De hecho, la clave de su éxito es el hecho de ser la herramienta idónea para gestionar **blogs**. Debido a que los blogs son el tipo de web más popular de Internet y que WordPress seguramente es el mejor software para crearlos, el número de páginas que utilizan WordPress es mucho mayor que las que usan el resto de CMS juntos.

WordPress ha dejado, además, de ser un CMS orientado solo a blogs y se utiliza para crear todo tipo de webs. Su comunidad de usuarios y desarrolladores, muy entusiasta y dinámica, mejora casi día a día sus prestaciones y está consiguiendo que WordPress se utilice cada vez más como Framework de desarrollo de aplicaciones web, compitiendo directamente con Drupal, que siempre ha sido considerado el mejor CMS en ese campo.

A este respecto, WordPress aporta una API (***interfaz de desarrollo de aplicaciones***) que se puede utilizar para personalizar absolutamente el uso de Wordpress y crear sitios aprovechando las ventajas de WordPress y la versatilidad de trabajar directamente con nuestro código.

Seguramente los puntos fuertes de WordPress que le permiten seguir siendo el CMS más popular son:

- **Facilidad de instalación y de gestión**. WordPress se jacta de que su instalación no lleva más de 5 minutos. El panel de gestión también es considerado muy intuitivo y práctico.
- **Diversidad de opciones de hosting**. WordPress está presente como opción preinstalada en prácticamente todos los proveedores de hospedaje en Internet. Además la propia página oficial de WordPress, nos permite directamente gestionar nuestra web en su nube en lugar de instalarlo en nuestro sitio personal.
- **Gran número de plugins o extensiones**. Existen miles de temas, componentes, módulos, etc. Lo cual nos facilita enormemente el trabajo de gestión y personalización de nuestro sitio web.

- **Estética**. Muchos creadores de contenido acaban decidiendo entre cada producto por la apariencia estética. Esto no es una decisión poco profesional, la estética es fundamental y es clave en el éxito de la mayoría de sitios web en Internet. WordPress, casi al instante, consigue crear sitios de gran apariencia estética.
- **Documentación**. La página http://codex.wordpress.org posee documentación sobre todos los aspectos de manejo del CMS. Además se van traduciendo a diversos idiomas, entre ellos el español.
- **Facilidad para la personalización**. En los últimos años se ha convertido en uno de los pilares de WordPress. Numerosos desarrolladores utilizan el CMS como plataforma para la creación de aplicaciones web complejas.

8.1.2 DOCUMENTACIÓN

Como se ha comentado antes, en la dirección https://codex.wordpress.org/ WordPress dispone de documentación extensa sobre todos los aspectos de trabajo con este CMS. En https://codex.wordpress.org/es:Main_Page se dispone de la documentación traducida al español.

8.1.3 TÉRMINOS RELACIONADOS CON WORDPRESS

- **Entrada (*Post*)**. Es el nombre que se da a la publicación de un artículo que hace cada usuario de WordPress. Puesto que WordPress siempre ha estado orientado a la publicación de blogs, un post sería cada entrada del blog.

  Cada post tiene almacenada su fecha exacta de publicación (fundamental para la ordenación de los post). Además, posee un título y un contenido que, además del texto, puede incorporar todo tipo de elementos multimedia: enlaces, imágenes, vídeo, audio, etc.

- **Página (*page*)**. En la nomenclatura de WordPress es fundamental diferenciar una página de un post. Una página no tiene asignada una fecha de publicación, no se ordena por esa fecha. Dicho de forma más llana: no es una entrada de un blog, sino que se trata de un contenido estático dentro del sitio web. Los post se publican constantemente, las páginas se crean a un ritmo muchísimo menor.

- **Página de inicio (*Home page*)**. Se trata de la página que se muestra cuando el usuario accede a la raíz del sitio web. Está presente en cualquier sitio web, no solo en los gestionados por WordPress. Es la página que hay que cuidar más, ya que da esa primera y fundamental impresión, sobre nuestro sitio web.

  Por defecto WordPress la configura al estilo de los blogs, mostrando inicialmente los post más modernos. Naturalmente podemos modificar completamente su forma y funcionalidad.

- **Comentarios (*Comments*)**. Son los textos que los usuarios que leen el blog pueden incorporar al mismo para publicar sus propias impresiones sobre cada entrada. En definitiva, otorgan interactividad y el uso social de la web.

- **Taxonomía.** Literalmente es la ciencia que se encarga de la clasificación. Se trata del componente de un CMS que permite la agrupación de entradas en base a un determinado concepto. Las taxonomías más habituales se basan en categorías y en etiquetas, pero se pueden crear taxonomías totalmente personalizadas.
- **Categorías.** Temas por los que podemos organizar los contenidos de la web.
- **Etiquetas (*Tags*).** Palabras claves que se asignan a los post y mediante las cuales, permiten a los lectores encontrar post relacionados con esas palabras.
- **Menús.** Relacionado con los dos elementos anteriores, se trata de un elemento organizador del contenido de una web. WordPress les ha incorporado recientemente (desde la versión 3) ya que permiten una mejor organización del sitio web, haciendo que WordPress se convierta en un CMS con posibilidad de gestión de sitios web que no estén orientados a la publicación de blogs.
- **Enlaces (*Links*).** La idea es la misma que la de cualquier enlace de una página web. En el caso de WordPress se suele hablar de links para referirse a los enlaces dentro de la estructura del sitio, que permiten acceder a direcciones externas. Los enlaces se pueden categorizar.
- **Metadatos (*meta data*).** Se trata de los datos asociados a cada publicación, fundamentalmente a cada post, que proporcionan información sobre la misma y que puede ser útil para generar taxonomías u otras labores de consulta. El autor, la fecha de publicación, el tipo de codificación del texto, etc. son metadatos habituales. Algunos de esos metadatos provocan que el código fuente HTML incorpore elementos de tipo **meta**, para dar cabida a esa información.
- **Temas (*Themes*).** Se trata de formatos de maquetación de la web que se pueden utilizar para dar una apariencia concreta a nuestra web. Hay miles, muchos disponibles en la dirección https://wordpress.org/themes/ bajo el aval del propio WordPress. Hay que tener en cuenta que mediante temas para WordPress, se pueden incorporar códigos dañinos. Hay también numerosos temas ***premium***, de pago, construidos por empresas privadas que aportan un resultado muy profesional y vistoso.
- **Plugins.** Extensiones que se añaden a WordPress para mejorar sus prestaciones.
- **Widgets.** Componentes visuales que podemos añadir a nuestra web. En realidad son plugins, pero que son distinguibles visualmente. Dicho de otra forma más práctica: elementos que aparecen en las barras laterales para facilitar la navegación.
- **RSS.** Se trata del acrónimo de ***Really Simple Syndication***, un protocolo de sindicación de contenidos basado en XML. Aunque realmente en la actualidad está cada vez en más desuso, la relación de WordPress con este protocolo permite que los usuarios se suscriban a la web de modo que reciban en este formato, mucho más veloz, los posts del sitio web sin tener que entrar en la propia web.
- **Usuarios.** El punto culminante que convierte a un CMS en un verdadero entorno de gestión de contenidos complejo, es el hecho de diferenciar usuarios. WordPress en cada ver-

sión mejora la gestión de usuarios, permitiendo una mayor diferenciación de las tareas de cada usuario.

- **Roles**. Permiten aglutinar capacidades de modo que al asignar un determinado rol a un usuario, éste tendrá disponibles todas las capacidades de dicho rol.

8.2 INSTALACIÓN DE WORDPRESS

8.2.1 CREACIÓN DE UN SITIO ONLINE

La primera decisión cuando queremos implantar un sitio web mediante WordPress es si queremos utilizar un alojamiento WordPress ya preparado online o si deseamos configurar nuestro servidor web para implementar una instalación manual de WordPress.

La URL https://es.wordpress.com/ nos permite, gratuitamente, almacenar nuestro propio sitio web gestionado por WordPress online. En la opción gratuita tendremos 3GB para nuestro sitio web. Las opciones de pago (***premium*** y ***business***) nos dan más posibilidades de gestión, más espacio y otras mejoras, como los temas de tipo premium.

La ventaja, es la facilidad de instalación, el hecho de directamente disponer de alojamiento y, fundamentalmente, no necesitar administrar nuestro servidor web.

El proceso para crear nuestro sitio web online es muy sencillo. Bastará con darse de alta como usuarios de **Wordpress.com**, elegir la dirección de nuestro sitio web y elegir también la apariencia.

Otra opción es utilizar los asistentes online de los que disponen muchos proveedores de hosting. En realidad es hacer una instalación manual sobre el servidor en el que tenemos alojado nuestro sitio web, pero mediante un asistente que automatiza la instalación.

ACTIVIDAD 8.1:

- Realiza la Práctica 8.1: Creación de un sitio web online mediante WordPress.com.
- Navega por todas las opciones de administración del sitio creado para comparar las diferencias con la instalación manual.
- Entra en el sitio como un usuario normal de Internet para examinar la forma en la que cualquier persona vería dicho sitio.

8.2.2 INSTALACIÓN MANUAL DE WORDPRESS

En este caso disponemos de nuestro propio servidor web en el que realizaremos la instalación del CMS. Es una opción más compleja, pero permite personalizar y controlar absolutamente la instalación. Es lo recomendable cuando deseamos tener gestión real del funcionamiento de WordPress.

8.2.2.1 REQUERIMIENTOS INICIALES

WordPress requiere tener instalado un servidor web con PHP y además un servidor de bases de datos **MySQL** (o **Maria DB**). Además necesita una base de datos, que será donde almacene toda la información del sitio web, y un usuario con todos los permisos de creación, lectura, escritura, etc, sobre esa base de datos.

Una muy mala práctica es utilizar como usuario para WordPress el usuario **root** de la base de datos MySQL. Este usuario podría incluso eliminar la base de datos. Por otro lado, tampoco conviene dar como nombre de la base de datos ni para el usuario, el típico nombre **wordpress**. Cualquier ataque sobre nuestra base de datos, intentaría sonsacar la contraseña de ese usuario. Por ello debemos procurar que los nombres de la base de datos y del usuario no sean típicos.

Como tareas previas en MySQL, comunicando con ella, bien desde un entorno visual (como **MySQL Workbench** o **PHPMyAdmin**) o, bien desde la línea de comandos, debemos realizar las siguientes:

[1] Crear una base de datos para WordPress. Por lo dicho anteriormente, debemos evitar nombres como por ejemplo ***wordpress*** o ***blog***.

[2] Crear un nuevo usuario con un nombre también críptico. Si además sabemos que el servidor de bases de datos y el servidor web están en la misma máquina, hacer que ese usuario se asocie a ***localhost*** y de ese modo solo cuando accedamos desde el servidor local, podremos acceder a la base de datos.

Si el servidor de base de datos y el servidor web son máquinas distintas, entonces, se debe crear un usuario asociado al host cuya IP coincida con la del servidor web. Solo desde ese servidor se podrá acceder a la base de datos.

[3] Asignar al usuario anterior una contraseña lo más compleja posible.

[4] Dar todos los privilegios al usuario anterior sobre la base de datos creada en el punto 1

Ejemplo de preparación de base de datos para WordPress desde la consola del sistema. Suponemos que hemos accedido a MySQL con un usuario administrativo:

```
mysql> CREATE DATABASE dbA3D;
mysql> CREATE DATABASE dbA3D;
mysql> GRANT ALL PRIVILEGES ON dbA3D.* TO
    -> "usradminA3D"@"localhost" IDENTIFIED BY "A1s2D3f4G5h6_!";
mysql> FLUSH PRIVILEGES;
```

Mediante este código se crea en MySQL una nueva base de datos llamada ***dbA3D***, y un usuario llamado ***usradminA3D*** con permisos totales sobre esa base de datos (pero que no podrá acceder a ninguna otra base de datos).

8.2.2.2 DESCARGA Y PREPARACIÓN DEL CÓDIGO DE WORDPRESS

WordPress realmente es una aplicación creada en PHP, por lo que simplemente necesitamos descargar todos los archivos de código y colocarlos en el servidor PHP que deseemos.

La descarga se realiza desde la dirección http://wordpress.org/download. En esa dirección se nos permitirá descargar la última versión de WordPress en formato **zip** o bien en formato **tar.gz** (el formato habitual de archivo comprimido de Linux). La instalación desde esa página está en idioma inglés. La última versión de WordPress siempre está disponible en estas URL:

- http://wordpress.org/latest.zip
- http://wordpress.org/latest.tar.gz

Si deseamos realizar la instalación en español, la página oficial es https://es.wordpress.org/. Hay que tener en cuenta que la traducción al español de WordPress no siempre está al día y que puede contener algunas erratas.

Si estamos en un sistema que trabaja en modo consola de Linux, la descarga de la última versión (en inglés) podría hacerse con el comando:

```
wget http://wordpress.org/latest.tar.gz
```

Tras la descarga, el archivo descargado se debe descomprimir. El resultado de esa operación es un directorio que se llamará **wordpress**.

Ese directorio wordpress se debe colocar en el servidor web. La ubicación exacta dependerá de qué URL queremos asignar al servicio WordPress. Podemos hacer que el directorio wordpress sea la raíz de nuestro sitio web, en cuyo caso todo el sitio web será de tipo WordPress, pero podemos colocarlo a otra ubicación para formar una URL tipo http://www.misitio.com/wordpress.

Para ubicar el directorio wordpress, podemos subirlo a nuestro servidor mediante FTP o realizar la copia desde la consola, si tenemos acceso a ella; en definitiva, dependerá de la forma de acceder a nuestro servidor que tengamos a nuestra disposición.

Una vez más, por razones de seguridad, nos conviene cambiar el nombre del directorio de WordPress para que sea menos vulnerable.

Estructura de Archivos de WordPress

El directorio de WordPress recién descomprimido presenta la estructura que muestra la Figura 8.1. Se pueden observar los siguientes directorios:

- **wp-admin**. Contiene los archivos que permiten a WordPress realizar las principales labores administrativas, como son:
  - Conexión a la base de datos
  - Funciones del panel de administración de WordPress
  - Administración de páginas y post

- Actualización de WordPress

- **wp-includes**. Librerías y extensiones de WordPress que son parte de su funcionamiento fundamental. La mayoría del front-end se basa en este directorio.

Figura 8.1: Contenido del directorio de WordPress recién descargado de Internet

En definitiva, es donde se almacenan los archivos de uso común por parte de todo WordPress. Nunca se debe modificar este directorio.

- **wp-content**. Directorio en el que se almacena los archivos personales de la instalación de WordPress. Por ejemplo, dentro de este directorio tenemos los archivos de temas, plugins y archivos multimedia que el administrador de WordPress haya ido incorporando. Concretamente este directorio posee los siguientes subdirectorios:
  - **Plugins**. Cada plugin descargado ocupa un directorio dentro de éste.
  - **Themes**. Con los temas que se hayan descargado.
  - **Uploads**. Contiene los archivos multimedia que se hayan subido a WordPress.
  - **Upgrade**. Para las actualizaciones de WordPress.
  - Además puede haber otros directorios que son creados por plugins que necesiten almacenar información extra (galerías de imágenes, iconos, caché, etc.)

WordPress se distribuye con su código fuente que, evidentemente, podemos modificar. Aunque de hacerlo, arriesgamos a un mal funcionamiento de nuestro sitio web. Solo el directorio **wp-content** está preparado para añadir información de nuestra propia cosecha. Modificar los archivos de WordPress permiten personalizar aun más la instalación, pero hay que tener muy claro las implicaciones de esa modificación, por lo que está reservado a usuarios que conocen muy bien la arquitectura de WordPress.

8.2.2.3 CONFIGURACIÓN DE LA INSTALACIÓN DE WORDPRESS. ARCHIVO WP-CONFIG.PHP

Tras descomprimir y copiar el directorio de WordPress, debemos preparar el archivo de configuración. Dicho archivo se llama **wp-config.php**, y se debe ubicar en el directorio de WordPress (o bien en la raíz general del sitio web). Inicialmente este archivo no existe, pero disponemos de un archivo llamado **wp-config-sample.php**. Podemos copiar ese archivo, renombrar la copia con el nombre ***wp-config.php*** y modificar su contenido para indicar nuestra configuración de WordPress. La configuración mínima que tenemos que indicar es:

- Nombre de la base de datos de MySQL en la que se almacenará la información de WordPress

- Nombre y contraseña de un usuario con permisos de lectura y escritura sobre esa base de datos
- Nombre que tiene el servidor web que aloja el servicio WordPress
- Codificación del texto de las tablas de WordPress (lo aconsejable es UTF-8)
- Prefijo que se le pondrá a todas las tablas de WordPress (suele ser *wp_*)

Todos esos datos se pueden cambiar ya que, el archivo de configuración ,es un archivo PHP que define una serie de constantes (a través del comando **define**) mediante las cuales se modifican todos esos parámetros. Ejemplo de archivo de configuración:

```
define('DB_NAME', 'wordpress');
define('DB_USER', 'user_wordpress');
define('DB_PASSWORD', 'A1s2D3f4G5h6_!');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');
$table_prefix = 'wp_';
```

Otra opción, en lugar de modificar a mano este archivo, es directamente, tras la descompresión y copia del directorio con el código WordPress en el servidor web, acudir a la dirección de WordPress en el servidor web. Entonces se lanzará un asistente que nos permitirá determinar estos parámetros a través de un formulario.

Hay una razón para modificar a mano el archivo y es modificar la configuración del cifrado de claves de los usuarios de WordPress. Se trata de un apartado del archivo referido a las claves de autentificación y códigos salt que, en el archivo inicial de configuración, tendrá este código:

```
define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');
```

Mediante estas líneas podemos indicar claves aleatorias de cifrado y códigos ***salt*** (véase el apartado 4.5 *"Cifrado"*, en la página 206, para más información sobre el cifrado en PHP), a fin de incrementar la seguridad de WordPress. Lo más sencillo es generar claves y códigos aleatorios a través de la dirección: https://api.wordpress.org/secret-key/1.1/salt/ la cual nos proporciona las líneas que debemos sustituir en el archivo **wp-config.php** a fin de que el cifrado de claves en WordPress sea más seguro.

Hay otras opciones más avanzadas que podemos indicar en el archivo de configuración, algunas de ellas se rellenan automáticamente al modificar la configuración desde la propia página WordPress.

También hay que indicar que podemos no rellenar a mano el archivo de configuración. Si accedemos, mediante su URL, a la raíz de WordPress desde el navegador (por ejemplo, con http://misitio.com/wordpress) nada más haber copia el directorio de WordPress en nuestro servidor, WordPress detecta que no hay archivo de configuración y un asistente nos ayudará a rellenarlo pidiéndonos los datos necesarios (base de datos, usuario, etc.) desde un formulario. No es lo más recomendable porque dejaríamos sin rellenar las opciones sobre el cifrado de claves comentadas anteriormente, con lo que dejamos menos protegida la instalación.

8.2.2.4 INSTALACIÓN EN SÍ DE WORDPRESS

Con las acciones anteriores realizadas, falta instalar en sí WordPress. Esa instalación prepara las tablas necesarias en la base de datos MySQL y las rellena con datos de ejemplo (un primer post, un primer comentario, etc.).

Si los pasos anteriores se han realizado correctamente, al invocar la dirección URL de WordPress en nuestro servidor, por ejemplo: http://www.miservidor.com/wordpress, aparecerá el llamado ***Asistente en 5 minutos de WordPress***. Dicho asistente simplemente requerirá esta información, la cual se almacenará en la base de datos MySQL que hemos asociado con WordPress:

- **Título**. Es el título que tendrá nuestro sitio WordPress
- **Nombre de usuario**. Es un usuario que tendrá permisos de administración de WordPress. No es recomendable darle el típico nombre de **admin**, se debe indicar un nombre más críptico.
- **Contraseña**. Para el acceso del usuario anterior. Evidentemente como medida de seguridad, se debe indicar una contraseña compleja.
- **E-mail**. Lo utiliza WordPress para indicar notificaciones.
- **Privacidad**. Marcada, hace que nuestro sitio quede accesible a los buscadores, como Google por ejemplo. Desmarcada, el sitio queda fuera de toda posibilidad de búsqueda del mismo a través de Internet.

Simplemente con esta información, el asistente indicará que WordPress ya está instalado.

Estructura de la base de datos WordPress

En la base de datos MySQL que se indicó en la instalación de WordPress, tras la instalación, se habrán creado una serie de tablas. En ellas se guardan los datos que los usuarios del CMS WordPress van creando: post, páginas, comentarios, usuarios, taxonomías, enlaces, etc.

El diagrama relacional de WordPress se puede observar en la Figura 8.2. suponemos que se ha instalado WordPress con el prefijo ***wp_***.

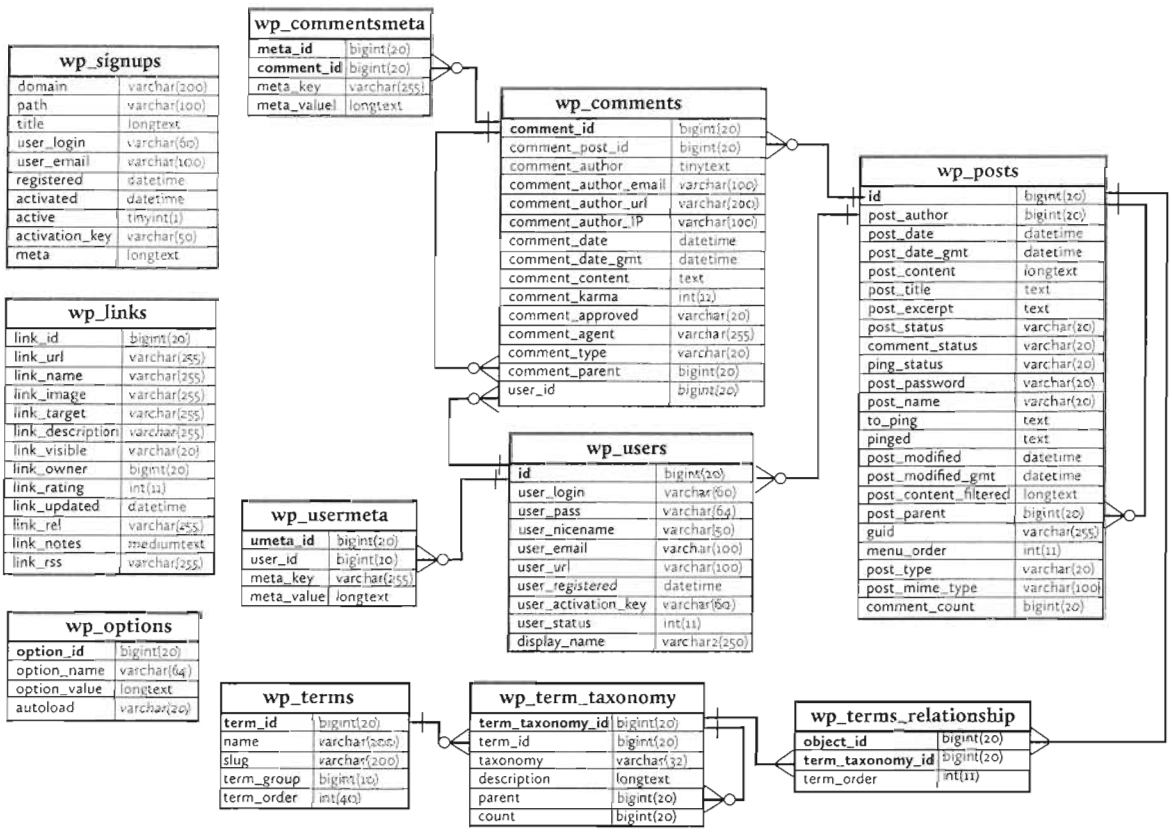


Figura 8.2: Esquema relacional de las tablas de WordPress

Las tablas que WordPress crea en la base de datos son:

- **wp_options**. Contiene la información general del sitio web, la cual se indicó durante el proceso de instalación.
- **wp_post**. Tabla central de WordPress en la que se almacenan las entradas (post) de los usuarios. Cada fila representa un post en nuestro sitio web. En ella disponemos de toda la información que WordPress almacena de cada post.
- **wp_postmeta**. Contiene los metadatos que asignamos a cada post. Estos son datos extra de nuestra cosecha que deseamos almacenar sobre cada post a fin de dotarles de más semántica.
- **wp_users**. Contiene los usuarios creados en WordPress.
- **wp_usermeta**. Contiene los metadatos de cada usuario (apellidos, nivel de usuario, dirección, departamento, etc.)
- **wp_comments**. Almacena los comentarios realizados en cada post.
- **wp_commentsmeta**. Permite almacenar metadatos para cada comentario.
- **wp_links**. Contiene todos los enlaces creados en la zona de administración de WordPress.

- **wp_terms**. Lista de todos los términos taxonómicos creados en WordPress.
- **wp_terms_taxonomy**. Tabla que permite enlazar cada término con la taxonomía con la que se relaciona.
- **wp_terms_relationships**. Tabla encargada de asociar cada página o post del sitio con los términos taxonómicos que deseemos.

Lo cierto es que, al final, la base de datos de WordPress es una base de datos MySQL normal que podremos utilizar en cualquier aplicación PHP que creemos, ya que conocemos el usuario y contraseña de acceso. Pero, hay que ser muy cautelosos a la hora de manipular los datos ya que cualquier error podría ser desastroso para nuestro sitio web. Modificar los datos, de entrada, no es buena idea, pero hacer consulta de ellos puede ser muy interesante.

8.2.3 DESINSTALAR WORDPRESS

Si hemos instalado WordPress mediante un proveedor de hosting en Internet, dispondremos de opciones automáticas para la desinstalación. Si deseamos eliminar WordPress de forma manual hay que realizar estos pasos:

[1] Eliminar el directorio de WordPress dentro de nuestro servidor web.

[2] Eliminar la base de datos y el usuario creados para WordPress en MySQL. Por ejemplo, mediante los comandos:

```
mysql> DROP DATABASE wordpress;
mysql> DROP USER user_wordpress@localhost;
```

Estas acciones eliminan WordPress completamente, incluyendo todos los post, comentarios, configuraciones, usuarios, etc. Si no deseamos perder esos datos, debemos realizar copia de ellos.

Si solo realizamos el primer paso, al volver a instalar WordPress e indicar el mismo usuario y base de datos, dispondremos de toda la información que teníamos previamente.

ACTIVIDAD 8.2:

- Instala WordPress en un servidor web local realizando los pasos de la Práctica 8.2.
- Tras realizar esa práctica, desinstala WordPress eliminando su directorio dentro de la carpeta de documentos de Apache y además elimina la base de datos y el usuario creados en MySQL para WordPress.
- Realiza la Práctica 8.3, que realiza una instalación manual más segura.

8.3 CONFIGURACIÓN BÁSICA DE WORDPRESS

8.3.1 EL PANEL DE ADMINISTRACIÓN DE WORDPRESS

Una vez instalado WordPress, nos proporciona una URL para realizar toda la configuración y administración del mismo. Se realiza añadiendo **/wp-admin** a la URL del sitio web donde se instaló WordPress. Por ejemplo: http://misitio.com/wordpress/wp-admin/

Al acceder a esa dirección aparece el escritorio o ***dashboard*** de WordPress:

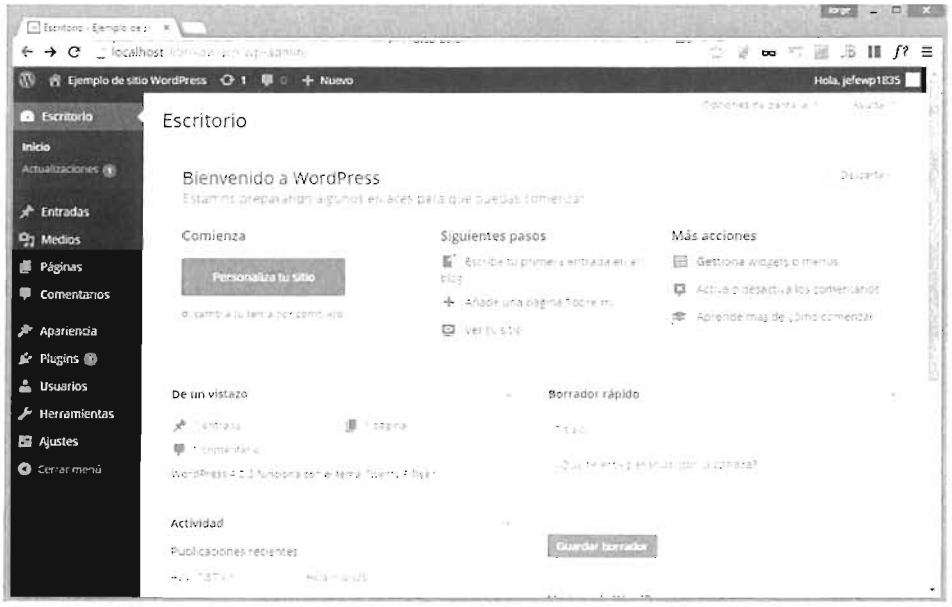


Figura 8.3: Aspecto del escritorio de WordPress

Desde el escritorio se pueden realizar todas las tareas administrativas, el menú izquierdo contiene todos los elementos de gestión del entorno WordPress.

Mediante los menús y enlaces desde la zona administrativa o escritorio de WordPress, podremos:

- Revisar las entradas (post) del sitio.
- Crear y editar las páginas estáticas del sitio.
- Gestionar los comentarios.
- Elegir y modificar los temas y la apariencia.
- Elegir y gestionar los ***widgets***.

- Configurar los menús.
- Añadir o eliminar Plugins.
- Gestionar los usuarios.
- Importar y Exportar el contenido del sitio.
- Modificar los ajustes generales
- Actualizar WordPress.

8.3.2 MODIFICAR LOS AJUSTES GENERALES DE WORDPRESS

La información proporcionada durante la instalación de WordPress se puede modificar desde la zona administrativa. Para ello basta con hacer clic en el apartado **Ajustes** (o ***Settings***) y luego elegir **Generales**. Desde ese apartado podremos realizar los siguientes cambios:

- **Modificar el título del sitio** (***Site title***), el cual inicialmente se indicó durante la instalación de WordPress.
- **Indicar una descripción corta del sitio** (***Tagline***). Tanto ese dato como el anterior son importantes de cara a que el texto realmente describa correctamente el sitio y, a través de dicho texto, el sitio sea fácilmente localizado por los buscadores.
- **URL de WordPress** (***WordPress address***). Dirección que posee el servicio WordPress.
- **Dirección del sitio** (***Site Address***). Normalmente tiene el mismo valor que la URL anterior, pero se puede indicar otra dirección, la cual se convertirá en la página de inicio de nuestro sitio.
- **Dirección de correo electrónico** (***email address***). A ella irán todas las notificaciones de WordPress.
- **Indicar si cualquier persona se puede registrar en el sitio** (***Membrership***).
- **Perfil predeterminado para nuevos usuarios**. Por defecto se asigna el de **suscriptor** (***subscriber***).
- **Zona horaria, y formato de fecha y Hora.**
- **Idioma.**

8.3.3 AJUSTE DE PERMALINKS

WordPress denomina ***permalinks*** a enlaces que permiten acceder de forma amigable a un determinado post, página, categoría, tag, etc. Los permalinks funcionan si está activado el módulo **mod_rewrite** de Apache. Si no lo está, hay que añadir esta línea al archivo de configuración de Apache:

```
LoadModule rewrite_module mod_rewrite.so
```

En muchas distribuciones Linux es más cómodo utilizar el comando de activación de módulos:

```
# a2enmod rewrite
```

Además, hay un archivo **.htaccess** en la raíz de WordPress que contiene los ajustes necesarios para que funcionen los permalinks (además de otros ajustes). Por lo tanto, en el archivo de configuración de nuestro servidor Apache, debemos asegurar que se ha permitido el uso de archivos **.htaccess** a través de la directiva **AllowOverride** (véase 2.4.5.2 *"Archivos .htaccess"*, en la página 59) a la que daremos, por ejemplo, el valor **All**.

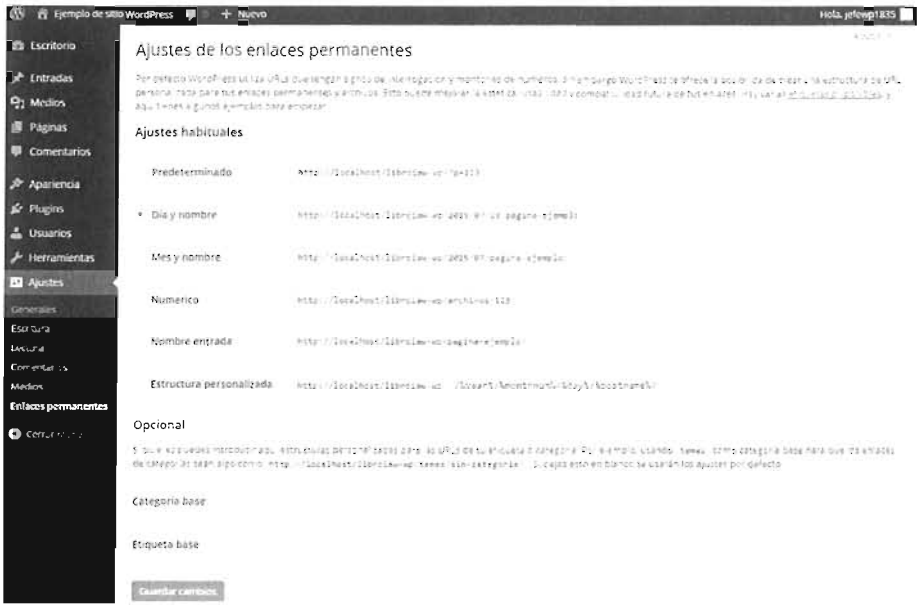


Figura 8.4: Zona de ajustes de Permalinks o enlaces permanentes del panel de ajustes de WordPress

La forma de estos enlaces, por defecto, es de tipo: http://misitiowordpress.com/p=2014 de modo que se añade a la ruta a la raíz de nuestro WordPress una cadena de consulta con una variable llamada ***p*** a la que se le asigna un número que es el que permite acceder a un elemento concreto de nuestro WordPress.

Podemos utilizar otra estructura si desde la zona de administración vamos a **Ajustes** (***Settings***), **Enlaces permanentes** (***Permalinks***).

Como nos recuerda el propio panel, es más fácil para los usuarios que los enlaces a las entradas de WordPress, utilicen una estructura más entendible para acceder a un determinado post. Podemos incluso utilizar una estructura personalizada a través de variables predefinidas. Las cuales son:

- **%year%.** Año.
- **%monthnum%.** Número de mes actual (12=diciembre).

- **%day%.** Día del mes.
- **%hour%.** Hora del día (de 0 a 23).
- **%minute%.** Minuto del día
- **%second%.** Segundo del día.
- **%post_id%.** Identificador único del post.
- **%postname%.** Versión normalizada del título del post. Si el título es ***mi primer post***, la versión ***postname*** será ***mi-primer-post***.
- **%category%.** Versión normalizada del nombre de la categoría. Si el post se ha relacionado con varias categorías, solo se puede utilizar una en el permalink. Las categorías se ordenan alfabéticamente, por lo que la primera, así ordenada, será la que se utilice en el permalink.
- **%author%.** Versión normalizada del nombre del autor.

Los permalinks aportan ventajas muy interesantes:

- Aumentan el posicionamiento del sitio web en los buscadores gracias a aportar URLs más semánticas.
- Direcciones amigables, que permiten ser recordadas con más facilidad.
- Estructuración del sitio web que facilita su navegabilidad. Además de que esa estructura de enlaces pueda ser fácilmente replicable a otros motores CMS como Joomla o Drupal que también tienen capacidad de utilizar permalinks.

8.3.4 ELECCIÓN DE TEMAS

La apariencia estética es uno de los aspectos fundamentales de un sitio web en Internet. Para facilitar la labor, WordPress proporciona un conjunto de temas a modo de plantilla para nuestro sitio web. La plantilla se elige en el apartado **Apariencia** (***Appearance***)-**Temas** (***Themes***).

Inicialmente aparecen tres temas. Haciendo clic en **Añadir un tema nuevo** (o ***Add Theme***) podremos descargar de la librería oficial de WordPress en Internet uno de sus muchos temas en línea.

Antes de elegir un tema, podemos observar su vista previa para tener una noción más completa de como será el resultado, antes de hacer clic en **Instalar** y que el tema aparezca instalado en nuestro sitio.

Sobre los temas instalados podemos hacer clic en **Activar** y entonces ese tema dictará la apariencia de nuestro sitio web. Podemos cambiar de tema las veces que deseemos. Así como podemos tener descargados todos los temas que deseemos, aunque solo uno puede ser el activo en cada momento.

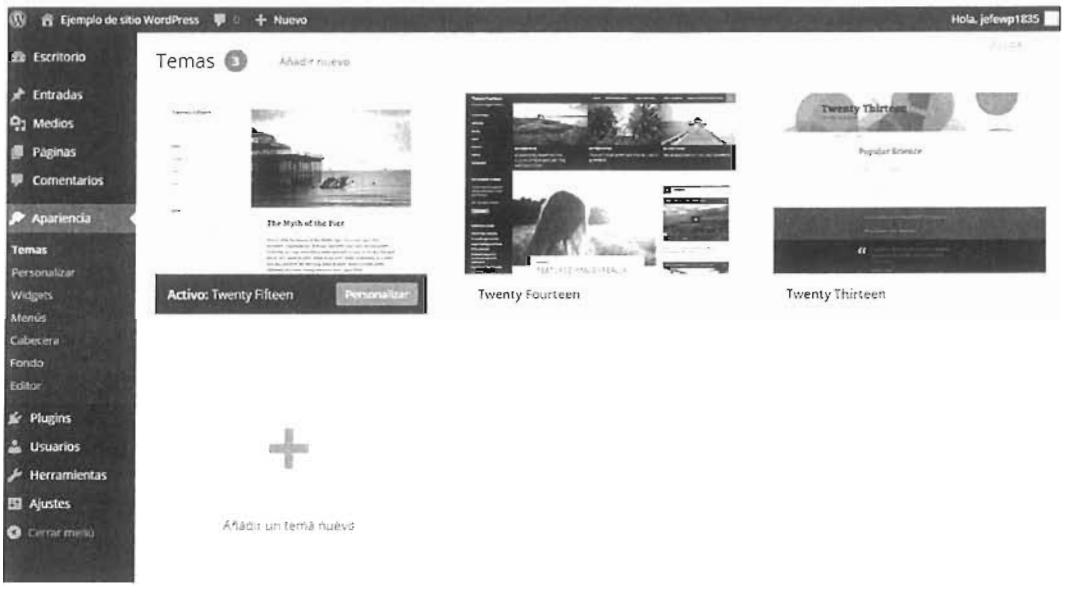


Figura 8.5: Panel de temas en WordPress

8.4 EXTENDER LAS CAPACIDADES DE WORDPRESS. PLUGINS

8.4.1 ¿QUÉ SON LOS PLUGINS?

Un plugin es una aplicación software creada para dotar de más funciones a un determinado software. Es decir, es un aplicación inmersa en una aplicación más grande. Los plugins de WordPress, por lo tanto, son aplicaciones creadas para dar más capacidades al propio WordPress.

Como el caso de los temas, los plugins los crea y distribuye la comunidad de desarrolladores de WordPress. La URL https://wordpress.org/plugins/ proporciona numerosos plugins que, además, podemos considerar fiables ya que están bajo el paraguas de WordPress. Hay muchos más plugins (algunos de pago) en otras direcciones de Internet.

8.4.2 EXAMINAR PLUGINS INSTALADOS

El apartado plugins del panel de control nos permite examinar los plugins que hay actualmente instalados. Sobre cada uno podemos:

- Ver los **detalles** del mismo.
- **Activar o desactivar.** Un plugin desactivado sigue estando disponible en los menús y ocupando espacio en disco.
- **Borrar.** Primero hay que desactivar el plugin. Una vez borrado si deseamos utilizarlo de nuevo, tendremos que volver a instalarlo.

- **Editar**. Mediante una ventana podremos ver el código fuente del plugin y modificarlo, si nos vemos capaces de ello. Un cambio típico que se suele hacer es cambiar los mensajes para que aparezcan en nuestro idioma.

Figura 8.6: Panel de plugins instalados

8.4.3 INSTALACIÓN DE PLUGINS

8.4.3.1 INSTALACIÓN DESDE EL PANEL DE PLUGINS

[1] Ir al apartado **Plugins** del panel de administración de WordPress y elegir **Añadir nuevo** (***Add new***).

[2] Elegir el plugin desde los menús donde aparecen agrupados y hacer clic en **Instalar plugin**.

[3] El plugin estará disponible en la lista de plugins disponibles. Podemos activarlo cuando queramos.

Figura 8.7: Panel de adición de plugins de WordPress

8.4.3.2 INSTALACIÓN MANUAL

No todos los plugins están disponibles en el panel de WordPress. Otros los podemos obtener en otras ubicaciones de Internet. Los pasos para la instalación manual son:

[1] Descargar el plugin de la página de WordPress o de otras páginas, con cuidado sobre la confiabilidad de las mismas. La descarga resulta en un archivo ZIP.

[2] Ir al apartado **Plugins** del panel de administración de WordPress y elegir **Añadir nuevo** (***Add new***).

[3] Hacer clic en el botón **Subir plugin** (***Upload plugin***) que se encuentra en la parte superior.

[4] Pulsar el botón de selección de archivos y seleccionar el archivo ZIP previamente descargado.

[5] Pulsar **Instalar ahora.**

[6] El plugin estará en la lista de plugins instalados.

8.5 CARACTERÍSTICAS DEL CMS DRUPAL

8.5.1 INTRODUCCIÓN A DRUPAL

Si WordPress es el CMS más popular, Drupal está considerado el CMS de código abierto más potente. Curiosamente, la tecnología de la que parte es la misma que WordPress: **PHP** y **MySQL** (aunque en lugar de MySQL puede utilizar **PostgreSQL** o **SQLite**). Sin embargo, la orientación de Drupal es distinta.

Como ya se comentó en la unidad anterior, Drupal es un CMS de propósito general. Sirve para crear todo tipo de sitios web: blogs, foros de debate, sitios de e-commerce, sitios de contenido estáticos, etc. Es verdad que, cada vez más, los principales CMS (y entre ellos WordPress) ya no están orientados a un tipo concreto de sitio ya que la mayoría permite su absoluta personalización, pero indudablemente la vocación de Drupal parte de esa idea, lo que le hace más complejo de gestionar pero con muchas más posibilidades.

Drupal se basa en un motor conocido como **Drupal Core** que es la base de la potencia y prestaciones del CMS básico. Drupal Core contiene los elementos principales de Drupal, pero es posible extender mucho más este núcleo.

8.5.2 ELEMENTOS FUNDAMENTALES DE DRUPAL

Al final no difieren mucho de los comentados en el Apartado 8.1.3 *"Términos relacionados con WordPress"*, en la página 321; pero Drupal aporta nuevos conceptos. Los principales términos que maneja Drupal son:

- **Temas.** Con el mismo sentido que tenían en WordPress. Drupal también posee numerosos temas para asignar a la estética del sitio, los cuales podemos utilizar y personalizar.

- **Taxonomía.** Términos que se asocian al contenido para facilitar la búsqueda y que se puede jerarquizar. No difiere la idea que teníamos en WordPress.
- **Nodo.** Este sí es un término propio de WordPress. Un nodo es un elemento de contenido en el sitio web: una página estática, un post, una noticia, etc. En Drupal hay muchos más tipos de nodos (de hecho, tantos como queramos).
- **Contenido.** Empieza a ser un concepto indistinguible del anterior. Todo lo que rellena el front-end de la página. El matiz que se habla de nodos para entender el funcionamiento interno de Drupal; sin embargo, se habla de contenido para referirse al aspecto que presenta el nodo realmente en la página. Aunque en las últimas versiones de Drupal se habla indistintamente de nodo y de contenido.
- **Usuarios, roles y permisos.** En este caso Drupal ofrece más posibilidades que Wordpress. Tiene más capacidad para la administración de usuarios y el establecimiento de sus capacidades y permisos.
- **Menús.** Enlaces organizados para acceder a las diferentes partes de sitio. Es decir, lo que comúnmente pensamos al utilizar este término en un contexto informático.
- **Regiones.** Secciones de una página creada en Drupal. Cabeceras, barras laterales, áreas de contenido, etc, son diferentes regiones.
- **Bloques.** Elementos en los que podemos dividir cada región de una página. Pueden ser menús de navegación, barras de información,etc. La diferencia es que la región hace referencia a una zona o área y el bloque hace referencia a un trozo de código que muestra algún tipo de información dentro de ese área.
- **Vistas.** Actualmente forman parte del núcleo de Drupal, pero antes de la versión 7 requerían ser instalados aparte. Se trata de un componente que permite crear listas de contenido (galerías de imágenes, fuentes RSS, etc). No todos los sitios tienen vistas.
- **Módulos.** Plugins. Componentes que aumentan la funcionalidad de Drupal.
- **Rutas.** Las URL de recorrido de las páginas del sitio. Especialmente la parte que va más allá de la URL raíz del sitio. En Drupal las rutas también pueden ser amigables.

8.5.3 ESTRUCTURA FUNCIONAL DE DRUPAL

Drupal establece un modelo de 5 capas en cuanto a sus funcionalidades:

- En la base hay una primera capa de datos. Se trata de la colección de nodos del sitio, es decir, el conjunto del contenido. Es la capa que se relaciona directamente con la base de datos.
- La segunda capa es la de los módulos. A través de ellos, los nodos se personalizan creando tipos de nodos que se muestran de diferente forma.
- La tercera capa es la que establece la navegación del sitio a través de bloques y menús. Los menús generan las URLs para llegar a todos los contenidos.

- La capa cuarta es la de los permisos de usuario. A través de ella, que actúa sobre la anterior se indica lo que un usuario es capaz de realizar o ver.
- La última capa es la que establece la estética final del sitio a través de plantillas (***templates***) o temas.

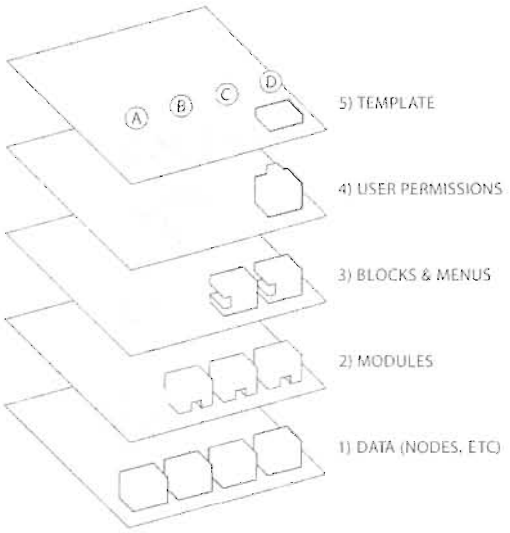


Figura 8.8: Modelo de cinco capas de Drupal. Imagen original: https://www.drupal.org/files/drupal_flow_0.gif

8.5.4 DOCUMENTACIÓN DE DRUPAL

En la URL https://www.drupal.org/documentation, disponemos de la documentación oficial de Drupal. La pega es que está solo en inglés. Hay diversas páginas en Internet que contienen información sobre Drupal en español (como http://drupal.org.es/), pero no disponen de documentación tan actualizada.

8.6 INSTALACIÓN DE DRUPAL

8.6.1 REQUISITOS PREVIOS

Drupal se puede instalar en un servidor que tenga Apache, Nginx o Microsoft IIS. En general, el servidor web habitual es Apache (se requiere al menos la versión 1.3). Drupal posee (al igual que WordPress) de archivos **.htaccess**, por lo que se necesita activar la directiva **AllowOverride All** en el archivo de configuración de Apache (véase Apartado 2.4.5.2 *"Archivos .htaccess"*, en la página 59).

Además se requiere tener activo el módulo de PHP con al menos la versión 5.2.5. Algunos módulos de Drupal pueden fallar con las versiones más modernas de PHP.

Figura 8.9: Contenido del directorio de Drupal en su distribución Core, una vez descomprimido

Como bases de datos, Drupal, a diferencia de WordPress, admite tres servidores: **MySQL** o compatibles (al menos la versión 5, Drupal 8 requiere la 5.5), **PostgreSQL** al menos con versión 8.3 y **SQLite** con al menos la versión 3.3.7. Lo habitual es utilizar MySQL (aunque PostgreSQL es una base de datos con muchas más posibilidades).

8.6.2 PREPARACIÓN DEL DIRECTORIO DE DRUPAL

Al igual que ocurría con WordPress, la instalación de Drupal consiste en colocar el código fuente en el servidor web, en la ruta desde la que deseemos acceder a Drupal.

[1] **Descargar los archivos.** Drupal posee muchas distribuciones, la habitual es Drupal Core que es la contiene los archivos fundamentales. Pero hay distribuciones preparadas para crear una tienda online, foros de debate, blogs, etc. En el momento de escribir estas línea, la versión 7.38 de Drupal es la recomendada para descargar, la versión 8 de Drupal aún está en fase Beta. Conviene, en todo caso, descargar la versión recomendada.

Las descargas se realizan mediante FTP con la ruta:

http://ftp.drupal.org/files/projects/drupal-x.x.tar.gz

Donde las *equis*, hay que modificarlas por la versión que queramos descargar. Otra opción es acudir a la página de Drupal: https://www.drupal.org/download y desde ahí descargar la versión deseada en formato **zipo tar.gz**.

[2] **Descomprimir los archivos.** Obtendremos un directorio.

[3] **Mover el directorio al servidor.** Deberemos colocar ese directorio en el directorio del servidor que deseemos. De modo que si queremos que Drupal funcione desde la raíz de nuestro sitio, se colocaría en el directorio raíz (por ejemplo **htdocs** o **/var/www**).

8.6.2.1 ESTRUCTURA DE ARCHIVOS DE DRUPAL

En la Figura 8.9, se puede observar el aspecto habitual del directorio de Drupal una vez descomprimido. De esa estructura se pueden destacar estos archivos y directorios:

- **Directorio *includes*.** Contiene el código de las funciones y clases PHP que forman el núcleo de la API de Drupal.
- **Directorio *misc*.** Archivos auxiliares de todo tipo (JavaScript, CSS, imágenes, controles jQuery, etc).
- **Directorio *modules*.** Módulos del núcleo de Drupal. Los módulos externos no se almacenan ahí.

- Directorio *profiles*. Perfiles predefinidos para la instalación de Drupal.
- Directorio *scripts*. Archivos con código **Shell Script** para facilitar la realización de tareas a los administradores de Linux.
- Directorio *sites*. Es el directorio fundamental de trabajo. Todos los archivos añadidos por los usuarios (con privilegios) de Drupal, se colocan en este directorio. Dentro de este directorio podemos destacar estos subdirectorios:
  - *all*. Se compone de estos subdirectorios.
    - *libraries*. Librerías de terceros añadidas a Drupal.
    - *modules*. Módulos de terceros o propios añadidos por los usuarios de Drupal.
    - *themes*. Temas añadidos por terceros o creados por los usuarios de Drupal.
    - *files*. Directorio en el que se almacenan los archivos que suban los usuarios.
  - *default*. Directorio principal de Drupal. Contiene el archivo de configuración **settings.php**.
- Directorio *themes*. Temas del núcleo de Drupal.

8.6.3 PREPARACIÓN DEL USUARIO Y BASE DE DATOS DE DRUPAL

Simplemente deberemos crear una base de datos para que Drupal la utilice para meter todos los datos que se generen en el CMS. Usuarios, entradas, comentarios, enlaces, menús, etc., es decir, todo lo que tenga que ver con el contenido del sitio web.

Una vez creada la base de datos hay que dar permiso a un usuario para que pueda leer y escribir en cualquier tabla de esa base de datos. Ese usuario será el que utilizará Drupal para acceder a la base de datos.

La forma de hacerlo dependerá de si utilizamos PostgreSQL, MySQL o SQLite como sistema de bases de datos. En la raíz de Drupal disponemos de tres archivos de texto: **INSTALL.mysql**, **INSTALL.postgresql** e **INSTALL.sqlite** con las instrucciones para realizar estas acciones en cada uno de los sistemas de bases de datos.

8.6.3.1 ESTRUCTURA DE LA BASE DE DATOS

Drupal es un CMS que tiene más complejidad que WordPress. El esquema relacional está disponible en la URL https://www.drupal.org/files/drupal7-db-schema.png. En él se puede observar la cantidad de tablas que posee. La versión Core instala nada menos que 74.

Aunque es complejo es fácil determinar dónde se guarda la información. Así, por ejemplo, la tabla **users** guarda la información sobre los usuarios y la tabla **nodes la información** sobre los nodos. La instalación de Drupal creará todas esas tablas en nuestra base de datos e irá incorporando datos en ella a medida que lo necesite. Al final, en un sitio que actualiza a menudo sus contenidos, la base de datos se modifica continuamente.

8.6.4 PREPARACIÓN DEL ARCHIVO DE CONFIGURACIÓN

Drupal posee un archivo de configuración, cuya ruta (desde la raíz del directorio de Drupal) es **/sites/default/settings.php** (equivalente al **wp-config.php** de WordPress). En Drupal 8 hay un segundo archivo de configuración llamado **service.yaml** (YAML es un formato de texto para almacenar datos que cada vez es más popular).

Inicialmente no existe archivo de configuración. Pero disponemos de un archivo que sirve como plantilla llamado **/sites/default/default-settings.php**. Simplemente habrá que copiarlo en el mismo directorio y llamar a la copia **settings.php**. No se debe eliminar el archivo original **default-settings.php**, ya que se usa también para configurar Drupal. De ahí que haya que copiar el archivo y no renombrarlo.

Hay que asegurar que el archivo de configuración tiene permisos de escritura en los sistemas Linux/Unix. Pero tras la instalación, se debería dejar solo permisos de lectura para no comprometer la seguridad del sistema.

8.6.5 PREPARAR LA INSTALACIÓN EN OTRO IDIOMA

Drupal, por defecto, se instala en inglés. Para instalar en otro idioma (como el español o el catalán) hay que realizar estos pasos:

[1] Descargar el archivo de idioma desde la URL https://localize.drupal.org/download eligiendo y descargando el archivo de idioma deseado. Para el archivo en español en la versión 7 de Drupal la URL sería de este tipo:

http://ftp.drupal.org/files/translations/7.x/drupal/drupal-7.38.es.po

La versión de Drupal (en el enlace anterior es la 7.38) se cambiaría por la que tengamos.

[2] Mover ese archivo al directorio **/profiles/standard/translations/** (desde la raíz de Drupal). Suponemos que la versión Standard de Drupal será la que instalemos (ver apartado siguiente).

Por supuesto podemos instalar varios archivos de idioma. El problema es que no siempre las traducciones son completas, por lo que muchos administradores prefieren instalar en el inglés original.

8.6.6 EJECUTAR LA INSTALACIÓN

Suponiendo que el servidor web con PHP y el servidor de bases de datos estén funcionando y que ya hemos copiado y preparado los archivos de Drupal, tendremos ruta hacia la raíz de Drupal en ese servidor. La cual podría ser, por ejemplo, http://www.miservidorcete.com/drupal. Entonces, desde esa raíz invocaremos en el navegador al fichero **install.php**. Usando la ruta de

Drupal usada como ejemplo antes, sería: http://www.miservidorcete.com/drupal/install.php. En una instalación local típica sería http://localhost/install.php si Drupal está en la raíz del sitio.

Desde el navegador aparecerá un asistente encargado de configurar el archivo config.php de Drupal y cuyos pasos serán:

[1] Indicar el tipo de instalación. Para usuarios que empiezan con Drupal lo lógico es elegir ***Standard*** ya que viene con módulos preparados que facilitan su gestión. Los usuarios más avanzados eligen ***Minimal*** ya que instala Drupal de forma más eficiente.

[2] Elegir el idioma.

[3] Drupal verificará que el sistema cumple los requisitos previos: servidores web y de bases de datos con versiones correctas y funcionando, archivo de configuración preparado, etc. De no ser así, indica, mediante mensajes, los problemas para poder instalarse. Los cuales debemos de arreglar para continuar la instalación.

[4] Establecer los ajustes para comunicar con la base de datos: tipo de servidor, usuario, base de datos, contraseña, nombre o IP del servidor de base de datos, puerto y si deseamos prefijo para las tablas. El prefijo es útil para distinguir las tablas de Drupal de otras que pudiera haber en esa base de datos, y para encriptar un poco el nombre de las mismas y hacerlas menos predecibles ante ataques.

[5] Tras preparar la instalación se nos pregunta por:

- Nombre del sitio web.
- Dirección de e-mail que utilizará Drupal para comunicarse con el administrador.
- Nombre, contraseña y e-mail del usuario que será súper administrador de Drupal.
- Opcionalmente país y zona horaria.

Dispondremos del sitio Drupal creado con el aspecto por defecto.

Figura 8.10: Portada de Drupal tras finalizar la instalación de tipo Standard y sin conectar como usuario

8.6.7 ACCIONES TRAS LA INSTALACIÓN

8.6.7.1 DIRECTORIO *FILES*

En la ruta **sites/default** (dentro de la raíz de Drupal) hay que asegurar que tenemos un directorio llamado **files.** Si no existe, lo debemos crear porque es donde se suben los archivos que los usuarios carguen. Además, hay que asegurar que Drupal podrá escribir en él, lo que implica asignar privilegios de escritura sobre este directorio.

8.6.7.2 PROTEGER ARCHIVO DE CONFIGURACIÓN

Es importante que el archivo de configuración esté protegido contra la escritura.

8.7 CONFIGURACIÓN BÁSICA EN DRUPAL

8.7.1 PANEL DE ADMINISTRACIÓN

Inicialmente Drupal ofrece la vista de la Figura 8.10. Normalmente conectará con el usuario administrador, pero en el caso de no hacerlo aparece una barra lateral que nos permite indicar nombre de usuario y contraseña. Eso permite acceder a una barra de menú con opciones de control (equivalente al escritorio de WordPress). Es el **menú de administración.**

Figura 8.11: Barra de control o menú de administración de Drupal. Solo aparece para algunos usuarios registrados

Cualquier acción sobre el contenido (sea ***front*** o ***back end***) se lleva a cabo desde esa barra superior.

Podemos acceder a pantalla completa al panel de administración si añadimos ***?q=admin*** a la URL de drupal. Por ejemplo: http://localhost/drupal/?q=admin o http://localhost/drupal/admin si estamos utilizando la forma amigable para las URL.

8.7.2 INSTALAR TEMAS

Los temas de Drupal se encuentran en el apartado **Apariencia** del menú superior de control. Ahí veremos los temas instalados y cuál es el actual tema predeterminado. Si queremos elegir otro, hemos de hacer clic en el enlace **Activar** (si no lo está ya) y luego en **Establecer como predeterminado**. Si deseamos instalar más temas los pasos son:

[1] Ir a una página de confianza para descargar temas. Por ejemplo a la dirección URL http://drupal.org/project/themes.

[2] Descargar el archivo ZIP o TAR.GZ con el tema deseado.

[3] Dos opciones:

- Descomprimir el archivo descargado. El resultado será un directorio. Subir o copiar el directorio a la ruta **sites/all/themes**.
- También podemos ir al menú Apariencia y hacer clic sobre **Añadir nuevo tema**. Un panel nos permitirá directamente elegir el archivo comprimido.

[4] Activar y establecer como predeterminado si deseamos que funcione ya para dar formato a nuestro Drupal.

8.7.3 PERMITIR URL LIMPIAS

Las rutas por defecto de Drupal tienen el mismo problema que las de WordPress. Por ejemplo, para acceder a un nodo concreto la ruta sería algo como: http://server/drupal?q=node/83. Una URL limpia sería http://server/drupal/node/83.

Para que funcionen las URL limpias lo tiene que permitir el servidor web. En Apache debe estar activado la librería **mod_rewrite** y dejar cargar la configuración del archivo **.htaccess** que está en la raíz de Drupal, al igual que ocurría con los permalinks en WordPress (véase 8.3.3 *"Ajuste de Permalinks"*, en la página 332).

Para activar las URL limpias en Drupal hay que ir a **Configuración-Búsqueda y metadatos-URL limpias** (***clean URL***) en los menús de administración. Después simplemente, si no hay problemas, se activa la casilla correspondiente (si no lo está) y se guarda la configuración.

8.7.3.1 ALIAS

Una opción, más parecida aún a los permalinks de WordPress, es utilizar alias. Los alias no se pueden utilizar si no funcionan las URL limpias. Un alias aporta acceder a un determinado recurso a través de una URL a medida. Para añadir un alias a una de las direcciones existentes, podemos ir a **Configuración-Búsqueda y meta datos-Alias de URL**.

Veremos como en ese apartado ya hay alias que el propio Drupal Core (a través de su módulo **Path**) ya habrá activado.

8.7.4 GESTIÓN DE MÓDULOS EN DRUPAL

8.7.4.1 INTRODUCCIÓN A LOS MÓDULOS

Los módulos de Drupal son equivalentes a los Plugin de WordPress. Sirven para extender las capacidades de Drupal. Hay módulos que vienen ya preinstalados, y que simplemente hay que activar si deseamos utilizarlos, y módulos que son desarrollados por terceros. Siempre hay que tener cuidado con la descarga de módulos ajenos, ya que pueden ser capaces de introducir código malicioso en nuestro sistema.

8.7.4.2 GESTIÓN DE MÓDULOS

El apartado **Módulos** del menú de administración permite realizar todas las tareas que tienen que ver con los módulos. Nada más abrir ese panel aparece una lista de todos los módulos del núcleo, algunos estarán instalados y otros no.

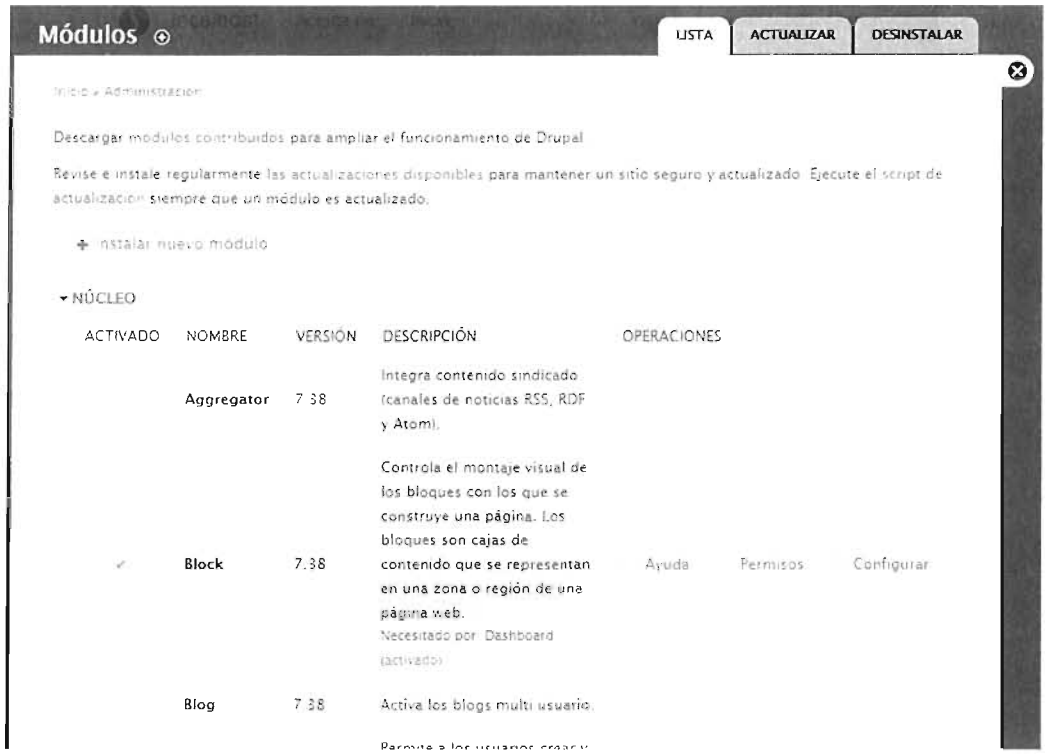


Figura 8.12: Panel de gestión de módulos de Drupal

Hay módulos que dependen de otros, por ejemplo, para activar el módulo **Forum** (que permite crear foros de debate) se necesitan activados otros cuatro (Taxonomía, Opciones, Campos, campos de almacenamiento SQL y comentarios).

Activar un módulo se realiza simplemente marcado la casilla de verificación que está en la columna **Activado**. Para que la activación se lleve a cabo, se debe hacer clic en el enlace **Guardar configuración** que aparece al final de la pantalla.

Hay que tener en cuenta que todas las acciones que Drupal es capaz de hacer, se representan con módulos. Por ejemplo, hay un módulo para gestionar los usuarios, si lo desactivamos no podremos gestionarlos desde el panel de administración; sí podríamos, manipulando el código de Drupal.

Hay módulos que permiten configuración, para lo cual tenemos un botón dedicado. También se permite establecer permisos de uso según el usuario. Esos botones dependen del módulo en sí, no todos disponen de esos botones.

8.7.4.3 INSTALAR MÓDULOS CONTRIBUTIVOS

Drupal llama módulos contributivos (también hay temas contributivos) a los módulos creados por terceros. Instalar un módulo de ese tipo es simplemente seguir estos pasos:

[1] Descargar el archivo comprimido (***zip*** o ***tar.gz***) del módulo. Asegurándonos de que es compatible con nuestra versión de Drupal.

[2] Descomprimir el archivo. Tendremos un directorio con el nombre del módulo.

[3] Copiar o subir ese directorio a la ruta **sites/all/modules**, partiendo desde la raíz de Drupal.

[4] Activar, si lo deseamos, el módulo desde el panel de administración de módulos.

Otra opción es no descomprimir el archivo y seleccionar el archivo desde el cuadro de **Añadir Módulos** cuyo enlace está en el panel de gestión de módulos.

La URL http://drupal.org/project/modules nos permite buscar y descargar módulos de terceros. Puesto que es una web que está gestionada por la propia empresa Drupal, podemos estar más tranquilos al instalar esos módulos.

8.8 PRÁCTICAS RESUELTAS

Práctica 8.1: Creación de un sitio web online mediante WordPress.com

- Crea un nuevo sitio web utilizando el servicio online de Wordpress, disponible en la URL https://es.wordpress.com

SOLUCIÓN: PRÁCTICA 8.1

[1] Vete a la dirección https://es.wordpress.com

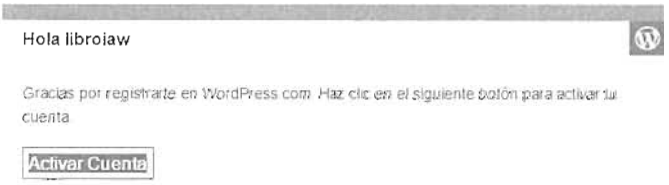
[2] Tras hacer clic en el botón *Hacer sitio web*, escribe la dirección deseada para nuestro sitio (obligatoriamente debe llevar el sufijo ***wordpress.com***). La propia página verificará si el nombre está disponible:

[3] Tras hacer clic en el botón *Crear tu sitio y continuar*, escribe tu email, nombre de usuario y contraseña. El nombre de usuario no puede coincidir con ninguno de los existentes.

[4] Al avanzar al siguiente paso, la cuenta estará creada. En el paso siguiente se nos ofrece dar de alta un nombre de dominio asociado al nuevo sitio, pagando el precio correspondiente. Si no lo deseamos, avanzar haciendo clic en el botón ***No gracias***.

[5] En el siguiente paso escogeremos un tema para nuestro nuevo sitio. Más adelante se podrá modificar si lo deseamos.

[6] Lee nuestro correo electrónico. Necesitamos verificar un mensaje que ha llegado de wordpress.com y que nos permita verificar el correo. Sin esa verificación, el sitio no quedará activado.



[7] Tendremos el sitio listo para usar. Podemos publicar un primer post de prueba pulsando en el botón del lapicero.

Práctica 8.2: Instalación manual de WordPress en un servidor PHP

SOLUCIÓN: PRÁCTICA 8.2

Suponemos instalado y configurado correctamente nuestro servidor web. También suponemos que disponemos de un servidor MySQL en funcionamiento y ya asegurado. En la segunda unidad de este libro puedes consultar cómo realizar estas acciones.

Pasamos a realizar los pasos previos: creación de una base de datos y un usuario preparados para ser utilizados por WordPress.

[1] **Conectar con MySQL como administrador.** Desde la línea de comandos (suponemos que tenemos acceso a los binarios de MySQL) del sistema, escribimos el siguiente comando:

```
> mysql -u root -p
```

De este modo conectamos con MySQL utilizando el usuario **root** (administrador). Tras el comando escribiremos la contraseña del administrador.

[2] **Crear la base de datos para WordPress.** Deberemos utilizar un nombre para la base de datos un poco críptico. Yo he llamado a la base de datos ***libroiaw*** y como la he creado a las *18:35*, le he añadido la hora:

```
mysql> CREATE DATABASE libroiaw1835;
```

[3] **Crear el usuario que utilizará WordPress para acceder a la base de datos.** En el mismo comando aprovecharemos para otorgarle todos los privilegios sobre la base de datos:

```
mysql> GRANT ALL PRIVILEGES ON libroiaw1835.* TO
    -> "jefeiaw1835"@"localhost" IDENTIFIED BY "A1s2D3f4G5h6_!";
mysql> FLUSH PRIVILEGES;
```

Con esto ya tenemos realizadas las tareas previas. Solo quedan realizar los pasos más sencillos, los correspondientes a la instalación en sí del CMS WordPress.

[4] **Descargar WordPress.** Para descargar los archivos en español podemos hacerlo desde la dirección https://es.wordpress.org

Desde esa página basta con hacer clic en el botón **Descargar WordPress 4.4.2**. El resultado de la descarga es un archivo comprimido.

[5] Descomprimir el archivo ZIP descargado anteriormente. El resultado es un directorio que se llama ***wordpress***.

[6] Copiar el directorio, resultado de la descompresión anterior, dentro de nuestro servidor web. En este caso supondremos que tenemos un servidor web local Apache con PHP, por ejemplo implementado mediante XAMPP, con lo cual basta copiar el directorio descomprimido al clásico directorio htdocs de Apache.

[7] Tras copiar el directorio, le renombraremos como ***libroiaw-wp*** en lugar de ***wordpress***.

[8] Suponiendo que tenemos en ejecución el servidor Apache y MySQL, desde el navegador web ir a la dirección http://localhost/libroiaw-wp. Aparecerá la pantalla de instalación de WordPress, pulsar el botón ***¡Vamos a ello!***

[9] En la siguiente pantalla rellenar los datos referidos a la base de datos y usuario MySQl creado anteriormente:

[10] Finalizado ese asistente, pasamos a ejecutar realmente la instalación de WordPress. Parar ello pulsamos el botón **Ejecutar Instalación**, esto nos lleva al conocido instalador de 5 minutos de WordPress.

[11] En la pantalla siguiente escribimos el título de nuestro sitio web, el nombre del administrador, la contraseña, etc.

[12] WordPress nos indicará que está instalado. Ya podremos acceder a todas sus funciones.

[13] Comprobamos la instalación desde la dirección http://localhost/libroiaw-wp/wp-admin:

[14] Tras introducir el usuario y contraseña, accederemos al ***dashboard*** de WordPress.

Práctica 8.3: Instalación manual de WordPress mediante archivo de configuración

SOLUCIÓN: PRÁCTICA 8.3

En realidad es el mismo tipo de instalación que en la práctica anterior, solo que, en lugar de lanzar el asistente con las preguntas sobre la instalación editaremos el archivo de configuración de WordPress manualmente. Estos son los pasos

[1] Realizar los pasos 1 al 7 de la práctica anterior

[2] Copiar el archivo ***wp-config-sample.php*** y llamar a la copia ***wp-config.php***. El archivo se encuentra en el directorio de wordpress (que si se han seguido los puntos anteriores se llamará ***libroiaw-wp***).

[3] Editar el archivo wp-config.php y modificarlo para que tenga estos valores:

```
define('DB_NAME', 'libroiaw1835');
define('DB_USER', 'jefeiaw1835');
define('DB_PASSWORD', 'A1s2D3f4G5h6_!');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8');
```

[4] Ir a la dirección https://api.wordpress.org/secret-key/1.1/salt/ y copiar el código que nos ofrece esa página. Pegar ese código sustituyendo las líneas correspondientes en el archivo ***wp-config.php***. Por ejemplo el código podría ser:

```
define('AUTH_KEY','hF{7>WOY/L++#hbA6uZy$3ijW1==+5CX+Kx!:Ub`3siF474^@R&fbPeL#lm.3K.=');
define('SECURE_AUTH_KEY','8|7-(_Z+kQ=>G9=S,NFKP-,{Vt7cw.A~4eL9I|X}bhcO^~qL,]JKjEy_YS,QRYz|');
define('LOGGED_IN_KEY',' dM;<n3EujzY#g}q<6gCh1w#+.cu#9nNnC;F {n4$2Fr~auI#O8P=JP8F8[MNb0&');
define('NONCE_KEY','An2=rt 1qu-c?J@VxveWZ7gWG~|rW)MgBjF#p=-,mgT4K3]gU]lx_t)hP BVSq3e');
define('AUTH_SALT','zWW^`4H~L@tQ(`YNJ[;N**W<Zc_>tYt%W6-0TwvNLt-aT~#EdA.5RBZ9+:ty4d--');
```

```
define('SECURE_AUTH_SALT','k2mIH|^Mq]hd#Do+hO-evbR!jVD1n!i~&W;HoaNnyEDv:O7H;xj09[_9G2-<$+5A');
define('LOGGED_IN_SALT','Hom>pcK)-y%FR-ce{3&&V52RfeZ[!$1AJ(XodcQcXZRA4L]zTpV`skr/6x?!1-[7');
define('NONCE_SALT', 'Fe]AI5dv.;_zF1d`[}U=X*p?^836`9J%CS]kV-X0(]=x9Gt}(6-E81iZBcS10=g|');
```

[5] Acceder al directorio de WordPress en nuestro servidor. Si el servidor es local, será mediante: http://localhost/libroiaw-wp/. Saltará el asistente de instalación en 5 minutos, que se corresponde con los pasos 11 y 12 de la práctica anterior.

Práctica 8.4: Instalación de Drupal en un servidor Linux

- Realiza una instalación de Core Drupal 7 en un servidor Linux en el que ya funcione MySQL y Apache con PHP

SOLUCIÓN: PRÁCTICA 8.4

[1] Nos ubicamos en el directorio raíz de documentos de Apache (le suponemos en **/var/www**) y descargar la versión deseada de Drupal (descargamos la 7.38):

```
$ cd /var/www
$ wget http://ftp.drupal.org/files/projects/drupal-7.38.tar.gz
```

[2] Extraemos el directorio (y eliminamos el archivo comprimido):

```
$ tar vzxf drupal-7.38.tar.gz
$ rm drupal-7.38.tar.gz
```

[3] Movemos el directorio de Drupal a la ubicación y nombre deseado. En nuestro caso el acceso a Drupal será mediante http://localhost/libroiaw-dp por lo que:

```
$ mv drupal-7.38 libroiaw-dp
```

[4] Preparamos el archivo de configuración:

```
$ cd libroiaw-dp/sites/default
$ cp default.settings.php settings.php
```

[5] Descargamos y preparamos el idioma español:

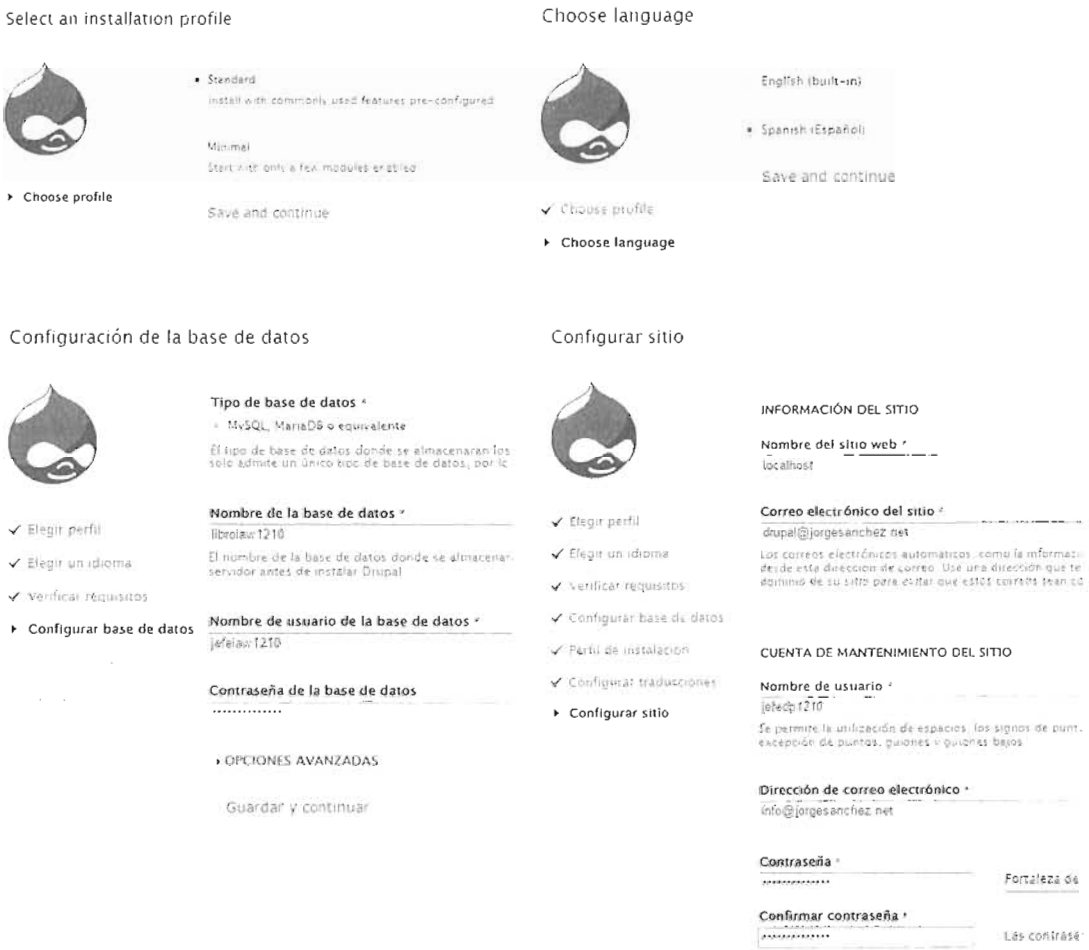
```
$ cd /var/www/libroiaw-dp/profiles/standard/translations
$ wget http://ftp.drupal.org/files/translations/7.x/drupal/drupal-7.38.es.po
```

[6] Preparamos base de datos y usuario de MySQL:

```
$ mysql -u root -p
mysql> CREATE DATABASE libroiaw1210;
```

```
mysql> GRANT ALL PRIVILEGES ON libroiaw210.* TO
    -> "jefeiaw1210"@"localhost" IDENTIFIED BY "A1s2D3f4G5h6_!";
mysql> FLUSH PRIVILEGES
```

[7] Desde el navegador accede a la dirección http://localhost/drupal/install.php y realiza el resto de pasos de la instalación (puede ser necesario tener que instalar algunas librerías):



[8] Modifica los permisos del archivo de configuración

```
$ chmod 444 /var/www/libroiaw-dp/sites/default/settings.php
```

[10] Crea el directorio files (si no existe) y dale permisos de escritura (si es necesario):

```
$ mkdir /var/www/libroiaw-dp/sites/all/files
$ chmod 755 /var/www/libroiaw-dp/sites/all/files
```

8.9 PRÁCTICAS PROPUESTAS

Práctica 8.5: Instalar temas en WordPress

- Teniendo WordPress instalado y en funcionamiento, consigue instalar un tema de la página oficial de WordPres en el que predominen los colores verdes y además sea un diseñocon la barra lateral a la derecha.
- Visita el sitio de http://topwpthemes.com/ y consigue instalar y activar alguno de sus temas.

Práctica 8.6: Instalar temas en Drupal

- Con Drupal instalado y en funcionamiento, consigue instalar y activar algún tema de la página https://www.drupal.org/project/project_theme

Práctica 8.7: Activar Permalink en WordPress

- Consigue que WordPress utilice enlaces a las entradas en esta forma (suponemos que WordPress se sirve en la dirección http://localhost/libroiaw-wp:

  http://localhost/libroiaw-wp/entrada/2015-07-10/nombre-entrada

  Donde *2015-07-10* es la fecha y ***nombre-entrada*** es el nombre del post.
- Escribe uno o dos posts y accede a ellos para comprobar que funcionan correctamente los permalinks.

Práctica 8.8: Activar URL limpias en Drupal

- Consigue que WordPress utilice URL limpias
- Compruébalo escribiendo un artículo y accediendo a él con el nombre.

  http://localhost/libroiaw-dp/node/2

  Suponemos que el número de nodo es el 2, si no, lo cambiamos por su número.
- Consigue acceder ahora a ese artículo con el alias: http://localhost/libroiaw-dp/prueba

Práctica 8.9: Instalar Joomla!

- Instala Joomla en un servidor PHP con MySQL.
- Comprueba la instalación.
- Modifica el tema por defecto y cámbialo por otro de tu gusto.

Práctica 8.10: Instalar Drupal mediante Drush

- Drush es una utilidad auspiciada por el equipo de Drupal que permite automatizar muchas tareas como instalar Drupal, instalar módulos, temas, realizar actualizaciones, etc.
- Documéntate sobre como instalar drush y hazlo en una instalación Ubuntu en la que tengas funcionando Apache + PHP +MySQL correctamente.
- Usa drush para instalar Drupal en esa misma máquina.
- Consigue instalar drush y con él instalar Drupal en una máquina Windows en la que est´´e funcionando Apache y MySQL mediante el paquete XAMPP.

8.10 RESUMEN DE LA UNIDAD

- WordPress es el CMS más popular gracias a: su facilidad para la instalación y configuración, diversidad de opciones de hosting, número y calidad de plugins y temas, estética, documentación y facilidad para la personalización.
- Las entradas o post en WordPress son los contenidos que cambian a menudo y que se organizan por su fecha de publicación. Las páginas son contenidos más estáticos que sirven como información general y que no se organizan en clave temporal.
- Además WordPress maneja términos como: taxonomía, etiquetas, categorías, temas, plugins, widgets, metadatos, permalinks, enlaces, menús, usuarios y roles.
- La instalación de WordPress, en un servidor propio, pasa por crear una base de datos MySQL y un usuario con permisos de modificación sobre ella. Tras esos pasos, un simple asistente finaliza la instalación.
- La configuración de WordPress se realiza a través del archivo **wp-config.php** situado en la raíz de WordPress o en la raíz general del sitio web en el que estamos publicando WordPress.
- La información que WordPress va añadiendo en nuestro sitio web se guarda en las tablas de la base de datos, las cuales tienen una estructura definida, y en el directorio **wp-content**. El resto de directorios de WordPress contienen su código fuente.
- Drupal es un CMS más potente, pero más difícil de gestionar que WordPress, al menos al principio.
- Sus conceptos son parecidos: taxonomía, temas, plugins o módulos, usuarios, etc. Aunque maneja otros conceptos como: nodos, bloques, regiones y vistas. La razón es que permite definir sitios más complejos.
- Drupal almacena los datos en una base de datos de estructura más grande que WordPress. El directorio **sites** (y especialmente su subdirectorio **all**) es donde se almacenan los componentes y elementos que el usuario ha realizado o descargado.
- La arquitectura de Drupal sigue un modelo de 5 capas: nodos, módulos, bloques y menús, permisos y plantillas.
- Como WordPress, Drupal requiere una base de datos y un usuario con acceso a ella. El resto de instalación se hace mediante un asistente sencillo.
- Comprender las bases del funcionamiento e instalación de Drupal y WordPress, serían suficientes para entender la instalación de cualquier otro CMS.

8.11 TEST DE REPASO

1 **Comparado con Drupal ¿Cuál dirías que son los puntos fuertes de WordPress?**

a) Facilidad de uso
b) Número de temas
c) Crear todo tipo de sitios web
d) Potencia para crear aplicaciones web a medida

2 **Comparado con WordPress ¿Cuál dirías que son los puntos fuertes de Drupal?**

a) Facilidad de uso
b) Número de temas
c) Crear todo tipo de sitios web
d) Potencia para crear aplicaciones web a medida

En terminología de WordPress, un post es...

a) Un formulario de introducción de datos
b) Una tarea que tenemos pendiente
c) Un dato oculto
d) Una entrada de blog

En un CMS ¿A qué se refiere el término taxonomía?

a) A la estructura de enlaces del sitio
b) Es un sinónimo de back-end
c) A la jerarquía de términos que permiten clasificar el contenido
d) Solo a las etiquetas y no a las categorías
e) Solo a las categorías

¿Qué es un Widget de WordPress?

a) Un componente visual que se puede añadir a las barras de la web
b) Un plugin, una extensión del sistema
c) Un rol de usuario
d) Un elemento de la taxonomía

¿Dónde se almacena por defecto el archivo de configuración de WordPress?

a) raízWordPress/wp-config.php
raízServidorWeb/wp-config.php
raízWordPress/wp-settings.php
raízServidorWeb/wp-settings.php

¿Qué requieren tanto WordPress como Drupal para poder lanzar su instalador?

Un servidor web con PHP y un servidor MySQL con una base de datos creada y un usuario con acceso a la base de datos

Un servidor web con PHP y un servidor MySQL. El usuario y la base de datos los pueden crear ellos

Un servidor web con PHP y un servidor MySQL de la que sabemos la contraseña

Un servidor web con PHP y un servidor de bases de datos con una base de datos creada y un usuario con acceso a la base de datos. WordPress requiere MySQL, Drupal puede utilizar otros servidores de base de datos

¿Dónde se almacena por defecto el archivo de configuración de Drupal?

raízDrupal/**settings.php**
raízServidorWeb/**settings.php**
raízDrupal/**sites/default/wp-settings.php**
raízDrupal/**sites/all/wp-settings.php**

¿Dónde se almacenan los temas que descarguemos en WordPress?

raízWordPress/**themes**
raízWordPress/**wp-includes/themes**
raízWordPress/**wp-content/themes**
raízWordPress/**wp-content/uploads/themes**

¿Dónde se almacenan los temas que descarguemos en Drupal?

raízDrupal/**themes**
raízWordPress/**includes/themes**
raízWordPress/**sites/default/themes**
raízWordPress/**sites/all/themes**

UNIDAD 9

GESTIÓN DE LOS COMPONENTES, CONTENIDO Y APARIENCIA DEL CMS

OBJETIVOS

- Reconocer las características fundamentales de la publicación de contenidos en los CMS
- Publicar contenidos de diferentes tipos
- Personalizar la forma de mostrar el contenido
- Utilizar elementos que permitan categorizar el contenido para facilitar su búsqueda
- Crear y gestionar comentarios de usuario
- Administrar la estructura de enlaces y menús del sitio web
- Insertar y gestionar plugins y componentes que mejoren la funcionalidad del sitio web

CONTENIDOS

9.1 BASES DE LA PUBLICACIÓN DE CONTENIDO

En todos los CMS la base de su funcionamiento consiste en la publicación de entradas en el sitio web y que esas entradas (o post en la terminología de WordPress) las pueden crear usuarios que no son administradores. Los administradores tienen la posibilidad de gestionar todo el contenido, pero la idea es que el contenido lo creen usuarios que no son ni administradores ni grandes conocedores de los lenguajes y tecnologías de creación de sitios web.

Al final todo lo que se publica mediante un CMS se acabará convirtiendo en código HTML traducible por un navegador. Esto implica que cualquier publicación tiene que tener en cuenta el formato y su ubicación final.

En general, de cara al contenido podemos distinguir, independientemente del CMS estos elementos:

- El contenido en sí que publicamos. Al cual se le puede dar formato HTML para distinguir el texto.
- Los elementos multimedia que se incorporen en el contenido. Las imágenes son los elementos, sin contar el texto, que más se utilizan. Su almacenamiento es un tema preocupante, ya que ocupan espacio. Se entiende que es parte del propio contenido de usuario y, por lo tanto, la mayoría de CMS incorporan herramientas para su gestión.
- La apariencia general del contenido. El tipo de letra del contenido, la zona en la que aparece, dónde aparece el título, cómo aparecen las imágenes, que barras y menús están alrededor de las páginas, qué formato general tienen las páginas. Todo ello lo suelen gestionar las plantillas o temas cuya gestión es otra de las bases del CMS.
- La navegación hacia los contenidos. Es muy deseable que cada contenido tenga una URL asignada y, más aún, que esa URL sea lógica y amigable. Además es fundamental que se pueda buscar contenido fácilmente a través de palabras clave, temas, autores, etc.
- Los permisos de creación de contenido. No todos los usuarios pueden generar contenido, incluso podemos desear que unos usuarios editen un tipo de contenido y otros no. Por ejemplo un usuario podría poner comentarios, pero no añadir nuevas entradas.
- La navegación general de la página. Una vez más para facilitar encontrar contenidos que deseemos y hacer el sitio lo más intuitivo posible.

9.2 AÑADIR ENTRADAS EN WORDPRESS

9.2.1 CREAR NUEVOS POST

La forma más habitual en WordPress de crear una nueva entrada o post, es desde el escritorio (***dashboard***), disponible si nos hemos conectado como usuarios, añadiendo en el navegador la

ruta **wp-admin** a la raíz URL del sitio WordPress. En ese panel basta con hacer clic en **Nuevo** (*New*) - **Entrada** (*Post*).

Figura 9.1: Menú de creación de nuevos post desde el escritorio administrativo de WordPress.

Figura 9.2: Panel de creación de nuevas entradas

Al crear una nueva entrada hay que realizar estas acciones:

- Indicar el título, usando el primer cuadro de entrada de texto del panel de creación de entradas. El título debe indicar de forma clara el contenido que tendrá la entrada. De esa forma se podrá clarificar la temática del contenido de esa entrada. Además, un buen título es clave para mejorar el posicionamiento en los buscadores.
- El contenido del post es, lógicamente, HTML. El panel de creación de post nos permite dar formato a través de botones, pero podemos hacerlo viendo el código HTML si hacemos clic en el botón **Texto** (***Text***). No se debe poner el autor en el contenido; tampoco la fecha ni las palabras clave. Todos los metadatos del contenido se gestionan aparte.
- Indicar categorías y etiquetas (tags) relacionadas con la entrada que estamos escribiendo.
- Añadir los elementos multimedia (imágenes, vídeos, etc) relacionados con la entrada que estamos escribiendo.
- Modificar opciones avanzadas: autor de la entrada, permalink de la entrada, privacidad, metadatos, tipo de entrada, etc.

- Previsualizar el aspecto que tendrá el post, mediante el botón **Ver entrada** (*View post*) que aparecerá cuando hayamos escrito contenido en el post.

Cuando tengamos el post a nuestro gusto, basta con hacer clic en el botón **Publicar** (*Publish*) para que la entrada sea publicada.

9.2.2 OPCIONES DE PUBLICACIÓN

El botón de publicar se encuentra en un panel desde el que se pueden modificar opciones de publicación. Las opciones disponibles en este panel son:

- **Estado** (*Status*). Indica si los visitantes pueden ver el post en la página. Sus posibilidades son:
  - **Publicado** (*Published*). El post está ya disponible en la página. Una entrada alcanza este estado al pulsar el botón de **Publicar** (*Publish*).
  - **Borrador** (*Draft*). El post queda almacenado en el panel de administración. En el escritorio de la zona de administración estará disponible con la lista de post, pero no se publicará hasta que se pulse el botón de publicar.
  - **Pendiente de revisión** (*Pending Review*). Es un tipo especial de modo borrador, en el que la entrada queda pendiente de que un revisor o administrador la publique. Hasta que eso no ocurra, el post no es visible en el sitio web. Es el estado en el que se dejan las entradas realizadas por los usuarios **Colaboradores**, los cuales no pueden publicar directamente sin la revisión de un usuario de mayor nivel.
- **Visibilidad** (*Visibility*). Por defecto es pública, pero puede ser privada (solo los miembros podrán ver el post) o protegido por contraseña.
- **Tiempo de la publicación**. Lo normal es que aparezca elegida la opción **Publicar inmediatamente**; lo que significa que el post aparece en cuanto se pulse el botón publicar. Pero podemos publicar en una fecha concreta, como por ejemplo, a las 8:00 del día siguiente.

9.2.3 PAPELERA

Cuando se borra una entrada, ésta se deja en la **Papelera** (*Trash*). La papelera está accesible desde la lista de entradas en el panel de administración. Sobre cada entrada podemos elegir **Restaurar** (*Restore*) o **Borrar permanentemente** (*Delete permanently*).

No hay solo papelera para los posts, también hay papelera para los comentarios, páginas y otros elementos. El funcionamiento es el mismo.

Conviene revisar la papelera de vez en cuando y eliminar los elementos permanentemente. Si queremos que la papelera borre automáticamente los elementos borrados cada, por ejemplo, 10 días. Entonces en el archivo **wp-config.php** se añadiría esta línea:

```
define('EMPTY_TRASH_DAYS', 10);
```

9.2.4 ESTABLECIENDO CATEGORÍAS Y ETIQUETAS

Tanto las categorías como las etiquetas nos permiten organizar las entradas que hayamos creado. Eso permitirá buscar entrada en base a estos conceptos, lo que facilita mucho las búsquedas.

9.2.4.1 CATEGORÍAS

Las **categorías** son textos (cortos) que describen el tema sobre el que trata el post. Si estamos escribiendo un post sobre el resultado del último partido de la ***final de liga NBA de baloncesto***, las categorías pueden ser ***deportes***, ***baloncesto***, ***ocio y NBA***. Además, se pueden jerarquizar; es decir, ***baloncesto*** puede ser una ***categoría hija*** de ***deportes***. Con lo que bastaría con asignar la categoría NBA para saber que estamos hablando de baloncesto que, a su vez, es un deporte. Si no asignamos categoría, se asigna a nuestra entrada la categoría **Sin categoría** (***Uncategorized***). No es buena idea ignorar las categorías, cada post al menos debería tener asignada una.

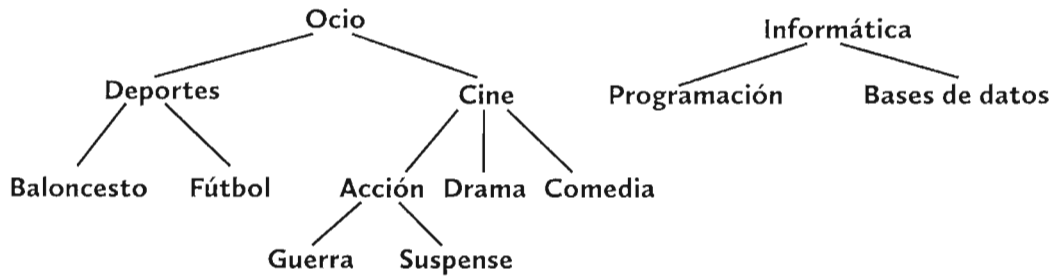


Figura 9.3: Ejemplo de categorías jerarquizadas.

Esta idea jerárquica nos permite realizar organizaciones de categorías complejas, como se puede observar en la Figura 9.3.

Para establecer categorías a una entrada nueva, basta observar el panel de categorías, en la misma ventana de creación de nuevos post, y marcar aquellas categorías que tienen relación con el post que hemos escrito.

Si la categoría que queremos indicar no existe, entonces pulsamos el botón **Añadir nueva categoría** (***Add new category***) y escribimos el nombre de esa nueva categoría, señalando la categoría de la que colgará esa nueva; es decir, señalando la categoría padre de la que estamos añadiendo, si es el caso.

Gestionar categorías de forma más avanzada requiere ir al apartado **Entradas** (***Post***) - Categorías (***Categories***) del panel de administración.

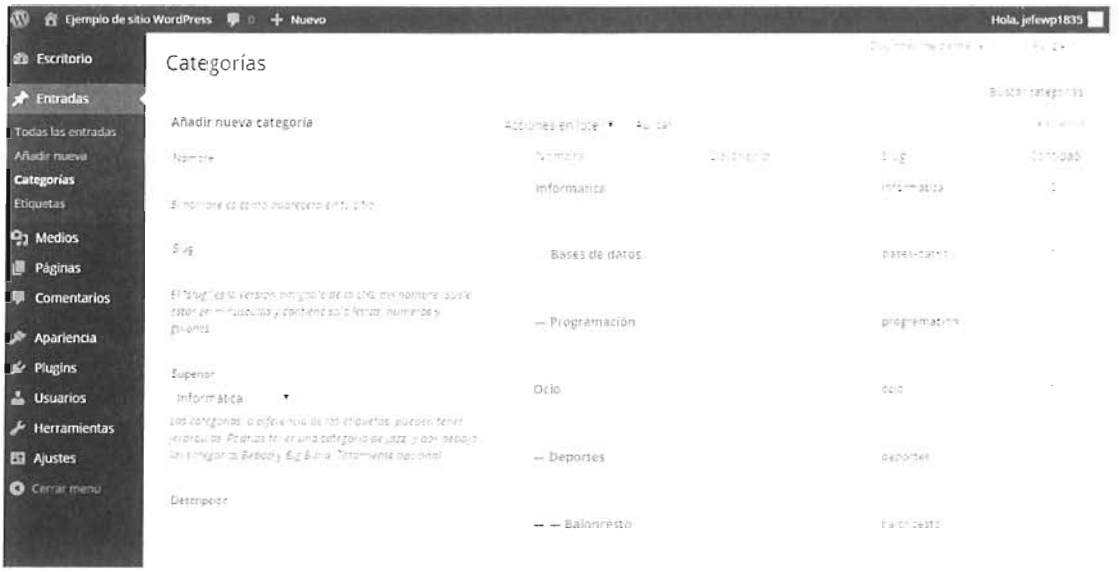


Figura 9.4: Panel de categorías

Desde ese panel podremos:

- Crear nuevas categorías indicando, además, a qué categoría superior pertenece para establecer la jerarquía correcta de nuestras categorías.
- Borrar o modificar las categorías existentes. Gracias al panel en el que aparece la lista de todas las categorías.
- Establecer el ***slug*** de cada categoría. El ***slug*** es el nombre amigable que damos a un elemento de WordPress (como el título, por ejemplo, de un post). Este nombre es con el que aparece la categoría en un permalink. Por ejemplo, si tenemos una categoría que se llama ***bases de datos*** y hemos indicado que para acceder a los post de nuestro sitio, se usen permalinks en los que aparece la categoría, ésta aparece con el slug, por ejemplo, ***bases-de-datos*** (observese Figura 9.4).
- Mostrar las entradas (post) pertenecientes a una determinada categoría. Para lo cual señalamos a dicha categoría y hacemos clic en el botón **Ver** (***View***).

La recomendación habitual es no tener más de 15 o 20 categorías; pero eso dependerá del tipo y complejidad de los contenidos que publiquemos.

9.2.4.2 TAGS O ETIQUETAS

La idea es similar a la de las categorías. Son palabras claves que se asignan a un determinado post. La diferencia es que su pretensión es menos jerárquica. Aunque no se recomienda hacer excesivas categorías, sí es recomendable utilizar numerosas etiquetas; aunque en cada post es mejor elegir 5 o 6 como mucho.

Volviendo al post que habla de la final de la NBA de baloncesto, en este caso podríamos asignar las palabras claves: ***baloncesto***, ***final***, ***NBA*** y ***partidazo***.

Para añadir etiquetas (tags) a un determinado post, basta ir al panel de etiquetas (***tags***) en la propia ventana de creación de nuevas entradas y escribir la lista de etiquetas separadas por comas. Tras pulsar el botón **Añadir** (***Add***), se asignarán las etiquetas.

Tras añadir, aparecerán las asignadas en ese panel. Podemos eliminar, si lo deseamos, algunas de las etiquetas añadidas desde el propio panel.

Lo ideal es que reutilicemos, si es posible, las mismas etiquetas en diferentes post. La finalidad es buscar post que encajen con un determinado tema o palabra clave. En cualquier caso es una muy mala idea denominar a la misma idea con dos nombres de etiqueta distintos; por ejemplo no debemos utilizar el tag ***Smartphone*** en algunos post y ***Teléfono inteligente*** en otros, se dificultaría la búsqueda de entradas sobre ese mismo concepto.

9.2.4.3 NAVEGAR POR EL SITIO UTILIZANDO CATEGORÍAS Y ETIQUETAS

Suponiendo que nuestro sitio WordPress tiene la URL http://misitio.com/wordpress, si añadimos la ruta:

http://misitio.com/wordpress/category/?cat=5

Se nos mostrarán las entradas relacionadas con la categoría número 5. Evidentemente esta forma de examinar entradas por categorías no es muy intuitiva, por ello, es mejor si hemos utilizado permalinks en las categorías asegurando que cada una de ellas dispone de un slug (nombre amigable). En ese caso este enlace: http://misitio.com/wordpress/category/baloncesto, nos mostrará todas las entradas asociadas a la categoría ***baloncesto***.

Lo mismo ocurre con las etiquetas (***tags***). Podemos mostrar todas las entradas asociadas a la etiqueta ***baloncesto***, de esta forma: http://misitio.com/wordpress/tag/baloncesto

9.2.5 OPCIONES AVANZADAS EN LA PUBLICACIÓN DE ENTRADAS

9.2.5.1 PANEL DE OPCIONES DE PANTALLA

El aspecto del panel de creación de post se puede modificar haciendo clic en el botón **Opciones de Pantalla** (***Screen Options***) disponible en la parte superior de la pantalla. Al hacerlo aparece un panel (Figura 9.6) que nos permite mostrar u ocultar algunos apartados en el panel de edición de entradas. Conviene siempre tener en cuenta este panel si observamos que no tenemos a la vista ninguna característica en la edición de entradas.

9.2.5.2 EXTRACTOS

Algunos temas de WordPress ofrecen la opción de no mostrar de golpe el contenido de cada entrada, sino mostrar un primer resumen al que se llama **extracto** (***excerpt***). El extracto es un contenido que se muestra inicialmente en el artículo, pero que permite hacer clic en un botón con el texto **más**, mediante el cual ya sí se muestra el post completo.

Sin definir extracto alguno WordPress muestra los 55 primeros caracteres. Pero podemos definir un extracto personal con el contenido que deseemos. Eso se hace a través del panel **Extracto** (*Excerpt*) disponible al añadir o editar una entrada.

9.2.5.3 COMENTARIOS

Hay dos paneles relacionados con los comentarios. En el primero indicamos si permitimos comentarios en cada entrada, de ser así, sobre cada entrada se podrán realizar comentarios por parte de los usuarios del sitio web.

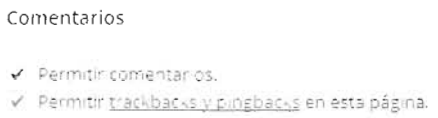


Figura 9.5: Panel de comentarios

Además, en ese panel podemos indicar si permitimos el uso de **trackbacks** y **pingbacks**, sobre la entrada que estamos editando. Los **trackbacks** y **pingbacks** permiten notificar si alguien está mencionando nuestra entrada. Normalmente interesa esta opción, salvo que detectemos que se nos está sometiendo a un ataque de ***pingback spam***.

9.2.5.4 CAMPOS PERSONALIZADOS

Se trata de otro panel (en inglés ***Custom Fields***) que nos permite establecer metadatos personales sobre una determinada entrada. Cada metadato se compone de un nombre y un valor. Estos metadatos no se muestran cuando se visita la página. Realmente se usan para establecer informaciones que pueden ser interesantes para los programadores de aplicaciones.

Figura 9.6: Panel de opciones de pantalla

9.2.5.5 MEJORA DE LOS PERMALINKS

En el apartado 8.3.3 *"Ajuste de Permalinks"*, en la página 332 , se explicó el funcionamiento de los Permalinks. Cuando escribimos un nuevo post, WordPress le asigna el permalink correspondiente y lo hará basándose en el título del mismo. Eso provoca que el permalink no sea muy amigable.

Sin embargo, podemos modificar la parte de la dirección correspondiente al título del post, para hacerla más amigable. Hay dos maneras de hacerlo:

- Desde el panel del título del post, haciendo clic en la URL que se asigna al post y modificando la parte final (Figura 9.7)

Figura 9.7: Modificación del permalink de una entrada a través del panel del título. Se observa el formato de antes y después de la modificación

- Desde el panel **slug**. Este panel inicialmente no está visible, pero se puede mostrar a través del panel de opciones de la pantalla. El slug, palabra intraducible en este contexto, que en inglés significa ***babosa***) es la parte amigable de una dirección URL. Al asignar un slug, estamos modificando la parte del permalink correspondiente al título del post.

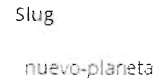


Figura 9.8: Panel mostrando el slug correspondiente al post anterior.

9.3 PUBLICAR ELEMENTOS MULTIMEDIA EN WORDPRESS

9.3.1 IMÁGENES

9.3.1.1 AÑADIR IMÁGENES

Figura 9.9: Panel de inserción de elementos multimedia

Las imágenes son una de las claves fundamentales en el éxito de un sitio web. Lógicamente WordPress incluye interesantes opciones para añadir imágenes a nuestras entradas y páginas.

Para añadir una imagen a una entrada que estamos editando, basta con pulsar el botón **Añadir Objeto** (*Add Media*) en la zona de edición. Los pasos a partir de la pulsación de ese botón son:

[1] Si es una imagen nueva, debemos colocarnos en la pestaña **Subir archivos** (*Upload files*). A continuación podemos arrastrar desde nuestro ordenador al panel de inserción o bien hacer clic en **Seleccionar Archivos** (*Select Files*) y buscar la imagen en nuestro ordenador.

[2] Una vez la imagen esté subida, hay que tener en cuenta que el límite de tamaño de la imagen es de 2MB y que solo se admiten los formatos **GIF**, **JPEG** y **PNG**. Una vez estemos en la pestaña de la **Biblioteca Multimedia** (*Media Library*) debemos asegurarnos de que está seleccionada la imagen que queremos subir.

[3] Hay un panel a la derecha llamado **Detalles de Adjuntos** (*Attachment Details*) que nos permite establecer opciones sobre la imagen que queremos insertar. Esas opciones son:

- **Título de la imagen** (*Title*). Es el texto que aparece en un pequeño cuadrito de texto cuando un visitante arrime el ratón sobre la imagen. Se asocia, en definitiva, al atributo **title** de la etiqueta **img** de HTML correspondiente a esa imagen.
- **Leyenda** (*Legend*). Un texto que aparecerá alineado bajo la imagen y que funciona como pie de la misma.
- **Texto alternativo** (*Alt text*). Se trata del texto que se asociará al atributo **alt** del elemento **img** de HTML asociado a la imagen. Es interesante rellenarlo con una breve descripción de la imagen, ya que ese texto es el que permite a los buscadores como Google hacer que la imagen pueda ser buscada.
- **Descripción** (*Description*). Espacio dedicado a escribir una descripción más extensa sobre la imagen. Esa descripción solo aparece si elegimos ciertos temas como plantilla de nuestro sitio.
- **Alineación** (*Alignment*). Posición de la imagen en la entrada. Si elegimos izquierda, la imagen dejará que el texto siguiente fluya a su derecha. Si elegimos derecha, el texto fluirá a su izquierda. Tenemos también la opción de no alinear y entonces la imagen aparecerá en el punto en el que la insertemos sin dejar que el texto la circunde.
- **Enlazado a** (*Link to*). Determina qué ocurrirá cuando hagamos clic en la imagen. Por defecto, lo que ocurre es que se mostrará la imagen original. Pero podemos poner un enlace a cualquier URL que deseemos o hacer que no ocurra nada al hacer clic.
- **Tamaño** (*Size*). Especifica el tamaño en píxeles que tendrá la imagen cuando la insertemos.

9.3.1.2 MODIFICAR IMÁGENES INSERTADAS

Tras insertar una imagen podremos modificarla haciendo clic sobre ella. Aparecerá una pequeña barra de herramientas desde la que directamente podremos cambiar la alineación, editar la imagen (botón con forma de lapicero) o eliminarla del post (botón en forma de equis).

Al pulsar el botón de editar, disponemos de un panel que permite modificar las opciones de la imagen. En ese panel aparecen las mismas opciones que al añadir la imagen y algunas opciones avanzadas nuevas como indicar una clase CSS para la imagen y otra para el enlace.

También podemos hacer clic en un botón llamado **Editar original**, que se explica más adelante y que permite modificar la imagen en la biblioteca de medios, y otro llamado **Reemplazar** que nos permite cambiar la imagen por otra.

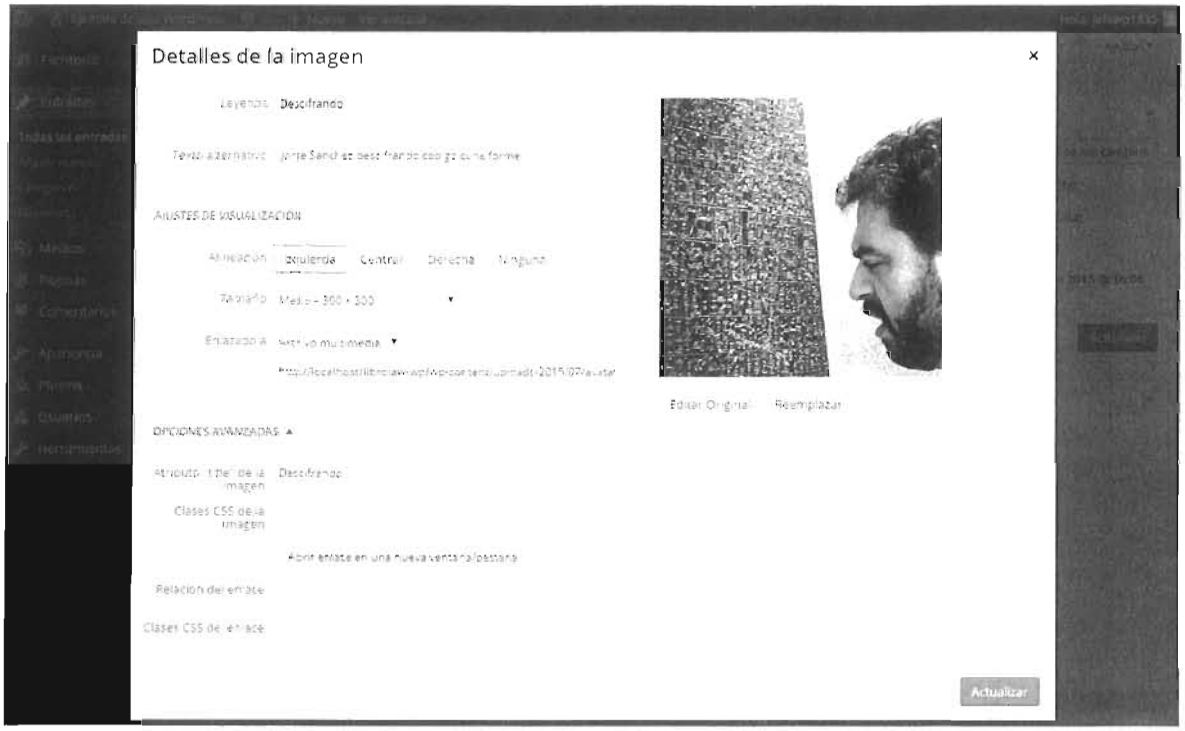


Figura 9.10: Panel de edición de imagen - Detalles de la imagen

9.3.1.3 MODIFICAR IMÁGENES EN LA BIBLIOTECA MULTIMEDIA

En el panel de administración de WordPress, disponemos en el apartado **Medios**, de un panel que nos muestra todos los medios multimedia insertados. Desde ahí podemos elegir una imagen y, tras hacerla clic, modificar sus propiedades.

Si hacemos clic sobre **Editar imagen** (botón que tiene el mismo efecto que el botón **Editar Original** cuando estamos modificando una imagen añadida a un post) llegamos a un cuadro llamado **Detalles de adjuntos** (***Attachment Details***) con el que podremos hacer modificaciones importantes a la imagen original dentro de la biblioteca de medios. Estas modificaciones afectan a cualquier inserción de la imagen en cualquier post. Concretamente podremos:

- **Recortar la imagen**. Para lo cual dibujamos con el ratón un rectángulo sobre la misma. Al estilo de los programas de edición de imagen, podremos modificar mediante sus tiradores este rectángulo. Si al final hacemos clic en el botón , entonces recortaremos la imagen en base al rectángulo dibujado.

- **Rotar la imagen a la izquierda o a la derecha**. Se utilizan para pasar la imagen de orientación horizontal a vertical y viceversa. Son los botones y .
- **Voltear la imagen horizontal y voltear en vertical**. Botones y respectivamente.
- **Modificar el tamaño original de la imagen**. WordPress realizará un escalado de la misma.
- **Deshacer y rehacer operaciones**. Con los botones habitualmente presentes en cualquier software.

En cualquier caso, son opciones que podemos no necesitar de WordPress, ya que los programas especializados de edición de imágenes son más apropiados para estas tareas. Tienen la ventaja de la rapidez, al no tener que abandonar WordPress, y de que podemos realizar cambios sobre imágenes ya insertadas.

9.3.1.4 CAMBIAR LOS TAMAÑOS POR DEFECTO DE LAS IMÁGENES

WordPress define por defecto tres tamaños para las imágenes de la biblioteca multimedia:

- **Para miniaturas** (***Thumbnails***). 150x150 píxeles.
- **Medio** (***Medium***). 300x200 píxeles.
- **Grande** (***Large***). 1024x1024 píxeles.

Estas medidas pueden ser modificadas desde el cuadro de administración si vamos a **Ajustes-Medios** (***Settings-Media***).

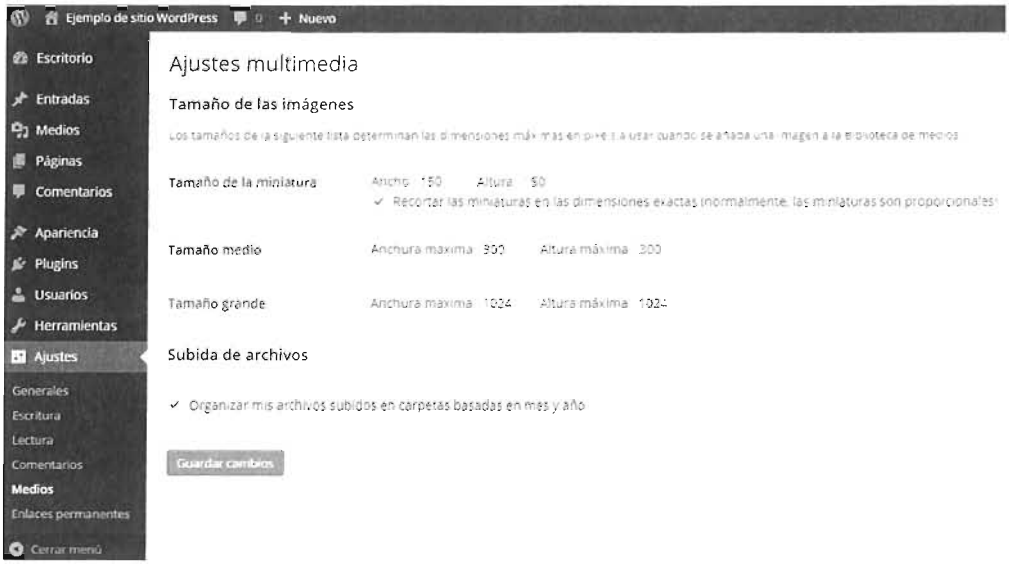


Figura 9.11: Panel de ajustes multimedia

9.3.1.5 USO DE IMÁGENES DESTACADAS

Muchos temas de WordPress permiten la posibilidad de utilizar lo que se conoce como **Imagen Destacada** (***Featured Image***). Esta es una imagen que no sirve para documentar el texto, sino que se utiliza como elemento estético de un post o una página. El enlace **Asignar imagen destacada** (***Set featured image***) permite subir una imagen o elegirla de la biblioteca de medios.

En la mayoría de temas de WordPress, las imágenes destacadas aparecen como cabecera de los post.

Añadir una imagen de este tipo no es muy diferente respecto a añadir una imagen normal. Desde el panel de edición de entradas, disponemos de un pequeño panel dedicado a las imágenes destacadas.

9.3.2 AÑADIR OTROS ELEMENTOS MULTIMEDIA

9.3.2.1 VÍDEO

Los vídeos se añaden de forma análoga. Sin embargo, en general, lo habitual al incorporar vídeos es hacer referencia a un vídeo almacenado en un servicio de streaming de vídeos, como por ejemplo YouTube. En cuanto coloquemos un enlace a un vídeo de YouTube, WordPress detecta el destino del enlace, entendiendo que es un destino multimedia, con lo que el vídeo aparece directamente en nuestra entrada del blog.

ACTIVIDAD 9.1:

- Crea una nueva entrada en tu instalación de WordPress.
- Colocaremos en la entrada el vídeo oficial del día del Niño Hospitalizado. Para lo cual basta pegar el enlace: https://wqww.youtube.com/watch?v=ijoX9mngvkw .
- Observa el resultado final en el sitio web.

9.3.2.2 AUDIO

Todo lo comentado sobre el vídeo vale para el audio. Podemos subir un archivo de audio a la biblioteca multimedia o enlazar hacia un audio en Internet. WordPress se encarga de mostrar de forma apropiada los controles del audio (siempre que le sea posible).

ACTIVIDAD 9.2:

- Crea una nueva entrada en tu instalación de WordPress.
- Colocaremos en la entrada el enlace a un audio procedente de Sound Cloud cuya URL es https://soundcloud.com/beachhouse/master-of-none. Es la canción ***Master of None*** de ***Beach House***.
- Observa el resultado final en el sitio web.

9.4 CREACIÓN Y GESTIÓN DE COMENTARIOS EN WORDPRESS

9.4.1 INTRODUCCIÓN A LOS COMENTARIOS

En la actualidad, la mayoría de sitios web proporcionan a los usuarios algún mecanismo de participación en los contenidos. En el caso de los blogs, es prácticamente obligado que los usuarios del sitio puedan publicar sus opiniones. En el caso de WordPress se hace a través de comentarios.

Los comentarios permiten conocer las opiniones de los usuarios y establecer discusiones. El riesgo es que se utilice está posibilidad por terceros para introducir publicidad de tipo ***spam*** o comentarios que intenten rebajar la credibilidad de nuestros contenidos. Es fundamental tener en cuenta este efecto secundario en el momento que nuestro sitio se abre a comentarios.

9.4.2 PUBLICAR COMENTARIOS

Cualquier persona que acceda a un sitio web en el que se permite publicar comentarios, puede hacerlo a través de un enlace que suele tener el texto ***Publicar un comentario***. La ubicación de este enlace dependerá del tema que hayamos elegido, aunque habitualmente suele estar localizado bajo el título del post.

Figura 9.12: Post en el que se ha resaltado el enlace que permite crear comentarios usando el tema *twenty fifteen*

En el caso de que nos hubiéramos dado de alta en el panel de administración de WordPress, no se nos preguntará nada y se nos permitirá directamente publicar el comentario (ya que WordPress sabe ***quiénes*** somos). Para las personas que no son usuarios registrados en WordPress, se les pedirá un correo electrónico, un nombre y la redacción del comentario. El texto del comentario puede incluir algunos elementos HTML, WordPress indica en la ventana de redacción del comentario, cuáles son los permitidos.

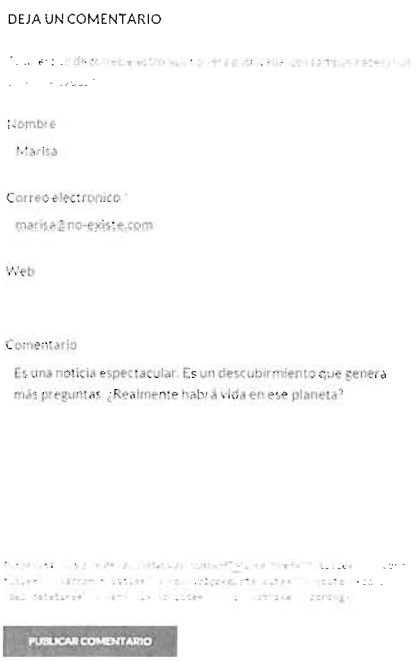


Figura 9.13: Ejemplo de redacción de un nuevo comentario

El hecho de pedir nombre y email no garantiza evitar un comentario de tipo ***spam*** generado automáticamente por un robot.

Un usuario que sea usuario de WordPress o que disponga de una cuenta **Gravatar**, servicio que proporciona perfiles de usuario en la nube, no necesita indicar nombre ni email, ya que está validado.

Una opción para evitar spam es solo permitir que solo los usuarios registrados puedan escribir comentarios. En este caso no correremos el riesgo de que se den de alta usuarios no deseados, ya que para que ese servicio sea eficaz, deberemos permitir que los usuarios se den de alta a sí mismos.

9.4.3 ADMINISTRAR COMENTARIOS

Todas las cuestiones y problemas con los comentarios se pueden gestionar desde la zona de Ajustes de Comentarios (***Settings-Comments***).

Las opciones que permite ajustar el panel son:

- **Permitir que se publiquen comentarios en nuevos artículos.** Si eliminamos esta casilla, no se permitirá publicar comentarios en nuevos posts.
- **El autor del comentario debe rellenar nombre y correo electrónico.** Es una primera medida de seguridad contra el spam, pero ya hemos comentado que hoy en día hay robots capaces de publicar spam porque saben cómo indicar un nombre y un email inexistentes.
- **Los usuarios deben registrarse e identificar para comentar.** Esta es la casilla que hace que solo los usuarios registrados puedan escribir comentarios.
- **Cerrar automáticamente los comentarios con más de *x* días.** Hace que los comentarios viejos no se lean.
- **Activar los comentarios anidados hasta *x* niveles.** Hace los comentarios más legibles cuando tienen respuestas de otros usuarios.
- **Separar los comentarios en páginas de *x* comentarios.** Cuando el número de comentarios de nuestro sitio es muy alto, esta casilla permite paginar dichos comentarios para una mejor legibilidad.
- **Enviarme correo cuando....** Permite que recibamos un correo cuando hay alguien que envía un comentario y/o cuando se requiere moderar un comentario. Es un aviso de

que debemos administrar esos comentarios, si el número de comentarios es muy alto, entonces puede generar un gran número de emails.

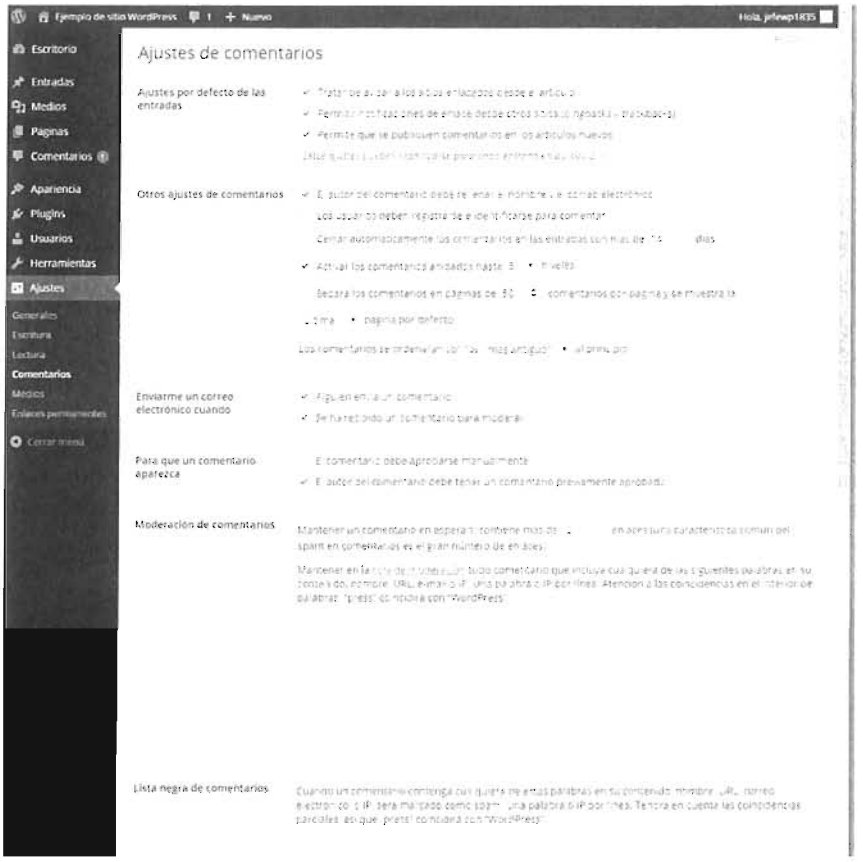


Figura 9.14: Panel de ajustes para la administración de comentarios

- **Para que un comentario aparezca...** Es, sin duda, la opción que permite una gestión más hábil de los comentarios. La primera casilla de este apartado, **El comentario debe aprobarse manualmente**, hace que cada comentario deba ser aprobado por un administrador para que se pueda ver publicado. Es una opción que requiere mucho trabajo de administración y, si tenemos muchos comentarios, es prácticamente inviable.

  La segunda casilla, **El autor del comentario debe tener un comentario previamente aprobado**, es muy interesante. Hace que la primera vez que una persona escriba un comentario tenga que ser aprobado para que se publique. Pero si esa persona hace más comentarios, se publicarán automáticamente. Digamos que validamos a los usuarios por el primer comentario, lo cual en un gran porcentaje es una estrategia suficiente de control de los comentarios.

- **Mantener un comentario en espera si tiene más de *x* enlaces.** Los comentarios de tipo spam se caracterizan por incluir numerosos enlaces, de ahí el interés de esta casilla.

- **Mantener en cola comentarios con ciertas palabras o contenidos.** Un gran cuadro nos permite indicar los términos prohibidos. Intenta evitar el spam o los comentarios poco apropiados, en función de su contenido.
- **Lista negra de comentarios.** Permite marcar como spam comentarios que contengan las palabras o términos que indiquemos. La diferencia respecto al ajuste anterior, es que estos comentarios no se pueden marcar para publicar, son directamente considerados spam y no se publicarán en ninguna circunstancia.
- **Calificación máxima.** Determina el nivel apropiado, en cuanto a la edad de las personas, de comentarios que permitimos en nuestro sitio.

9.4.4 MODERACIÓN DE COMENTARIOS

El apartado **Comentarios** del panel de administración nos permite ver y gestionar los comentarios de los usuarios. En ese apartado aparecen los comentarios publicados o pendientes de publicar.

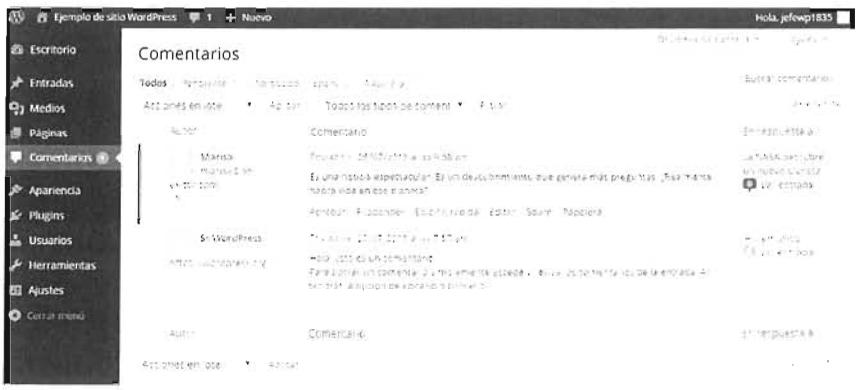


Figura 9.15: Panel de gestión de comentarios

Al arrimar el ratón sobre un comentario concreto podremos:

- **Aprobar** (*Approbe*). Aparece en los comentarios pendientes de aprobación. Esos comentarios no se publican hasta no hacer clic en este enlace.
- **Rechazar** (*Unapprobe*). Opción contrario a la anterior. Hace que un comentario previamente aprobado, pase a estado de espera para su aprobación. El comentario, por tanto, dejará de estar visible.
- **Responder** (*Replay*). Permite directamente responde al comentario.
- **Edición rápida** (*Quick Edit*). Permite modificar el contenido del comentario. Es una opción que tiene la sombra de la falta de ética, al poder ser utilizado como labor de censura.
- **Editar** (*Edit*). Hace la misma labor, pero a través de un panel de edición más completo.
- **Spam.** El comentario se marca como spam.
- **Papelera.** Mueve el comentario a la papelera.

ACTIVIDAD 9.3: PROBAR LA GESTIÓN DE COMENTARIOS

- Para probar los comentarios y entender sus posibilidades de gestión, es interesante conocer cómo funcionan los usuarios. Lo cual se explica en la siguiente unidad.
- En dicha unidad hay una práctica completa de publicación de entradas y comentarios con diferentes usuarios, se trata de la Práctica 10.1.

9.4.5 HERRAMIENTAS ANTISPAM DE COMENTARIOS

Como se ha explicado anteriormente el spam es el gran riesgo de los sitios que permiten comentar las entradas.

Para evitarlo, inicialmente WordPress aporta este funcionamiento de los comentarios:

[1] Un usuario publica un comentario.

[2] Si el usuario es nuevo, el comentario se debe aprobar para ser visto.

[3] Cualquier comentario de usuarios a los que se aprobaron comentarios será publicado automáticamente.

Se puede ser más restrictivo haciendo que cada comentario, sea del tipo que sea, deba de ser aprobado para ser publicado. Pero es inviable esta configuración en sitios con gran cantidad de comentarios.

Para ello disponemos de herramientas antispam, que se pueden incorporar a WordPress como plugins. De hecho WordPress incluye por defecto una: **Akismet**. Se trata de una herramienta personal, pero es de pago para sitios comerciales.

Akismet requiere simplemente darse de alta y obtener una clave API. Esa clave se indica en la zona de plugins de WordPress y directamente el plugin detectará y eliminará los comentarios de tipo spam.

En un sitio con gran interactividad, se hace indispensable el uso de estas herramientas, sea Akismet u otras, en sitios pequeños puede bastar la gestión de comentarios a través del panel de control de WordPress.

9.5 CREACIÓN DE PÁGINAS ESTÁTICAS EN WORDPRESS

9.5.1 PÁGINAS Y POSTS

Hasta este momento hemos visto cómo gestionar todos los aspectos de la publicación de entradas, lo que comúnmente se conoce como post. Los posts están pensados para contenido

que cambia a menudo, artículos, noticias, novedades, etc. El ejemplo claro de sitio donde todo el contenido se hace mediante post es un blog.

Ya se ha comentado anteriormente que WordPress es conocido como CMS de gestión de blogs. De ahí las enormes capacidades que tiene de administrar este tipo de contenido.

Pero es rara la web que no tiene otras secciones que muestran contenido que no cambia continuamente. La típica sección **Acerca de...**, presente en numerosos sitios web, en la que se presenta la información sobre el autor o propietario del sitio, es un ejemplo de contenido estático.

A este tipo de contenido WordPress lo llama **página** (***Page***). Así que, desde este momento los posts o entradas serán una cosa y las páginas otras (aunque el resultado de ambos elementos sea una página HTML).

9.5.2 CREAR PÁGINAS

Una vez más, se hace desde el panel de control de administración (ruta **/wp-admin** desde la raíz de nuestro sitio). Un apartado en el menú principal llamado **Páginas** (***Pages***) nos permite examinar las páginas existentes y crear nuevas.

Crear una página consiste en lo siguiente:

[1] Desde el menú **Páginas** (***Page***) elegir **Añadir nueva** (***Add New***). Aparece el panel de creación de páginas.

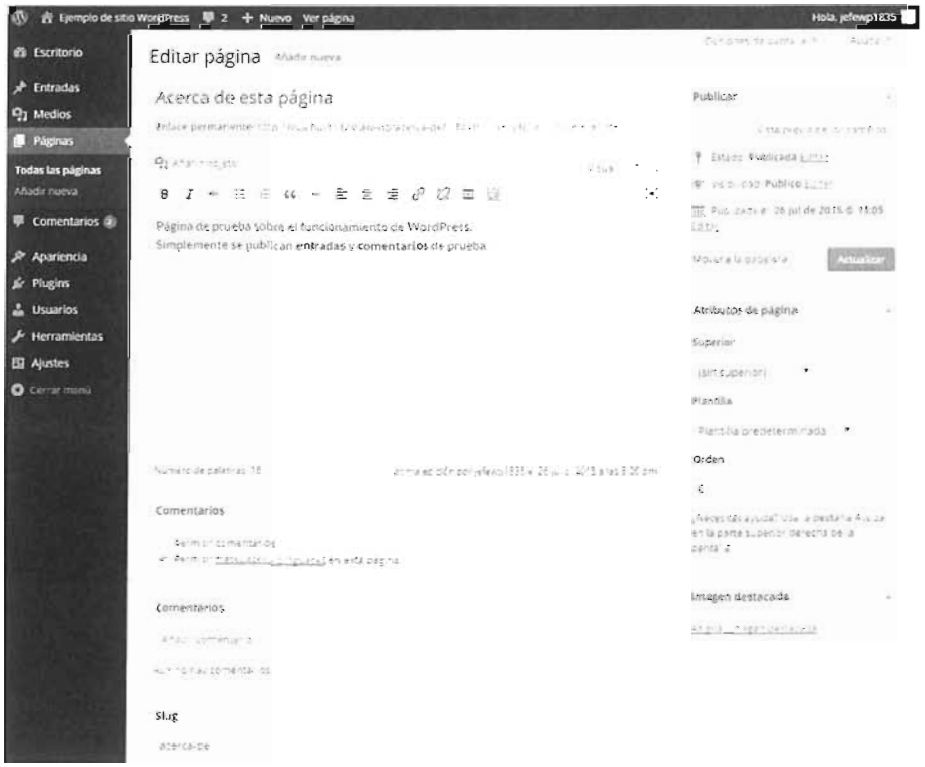


Figura 9.16: Panel de edición de páginas

[2] Establecer las propiedades de la página. La pantalla es muy similar a la de creación de post, los elementos son casi idénticos. Los elementos presentes son:

- **Atributos de página.**
- **Publicar.** Permite determinar la visibilidad, estado de la página (borrador, publicada o pendiente de revisión) y el momento en el que deseamos publicar la página.
- **Imagen destacada.** Como se comentó en el Apartado 9.3.1.5 en la página 373, es una imagen que aparece como cabecera de la página. Su posición e incluso aparición depende del tema elegido actualmente para el sitio.
- **Campos personalizados**. Con la misma función que en los post. Permite indicar metadatos personales para la página.
- **Autor**. Para modificar el autor de la página.
- **Slug**. Permite indicar la parte final del permalink (que normalmente es el título de la página) de acceso a la página.
- **Comentarios**. Permite indicar si permitimos o no comentarios en la página. Es bastante habitual no permitir comentarios en los contenidos estáticos.

Todos esos paneles se pueden mostrar (u ocultar) gracias al apartado **Opciones de pantalla**.

[3] Escribir el **título** y el **contenido**, el cual puede contener etiquetas HTML.

[4] Publicar la página.

Al publicar la página, normalmente aparecerá un nuevo enlace a ella en el menú principal (dependerá del tema en uso).

9.5.3 EDITAR PÁGINAS

En el caso de necesitar modificar páginas existentes, disponemos de dos posibilidades:

- Desde el panel de administración, yendo a **Páginas - Todas las páginas** (***Pages - All pages***). En él aparecen la lista de páginas actuales. Podremos realizar sobre ellas estas modificaciones, las cuales aparecen cuando arrimamos el ratón sobre una página concreta:
  - **Editar** (***Edit***). Pasaríamos a la ventana de edición de páginas, donde podríamos hacer las modificaciones que queramos.
  - **Edición rápida** (***Quick Edit***) . Pasamos a una ventana más sencilla de edición.
  - **Papelera** (***Trash***). Envía la página a la papelera.
  - **Ver** (***View***). Nos muestra la página en sí.

Figura 9.17: Panel de edición de páginas

- Acceder a la página utilizando su URL o un enlace a ella. Si nos hemos dado de alta previamente como usuarios del sitio y tenemos permiso de edición, aparecerá un enlace con el texto Editar que nos llevará al panel de edición de esta página.

9.6 PERSONALIZAR LA APARIENCIA DE UN SITIO WORDPRESS

9.6.1 TEMAS

Como se ha comentado anteriormente, una de las claves del éxito de WordPress es la gestión de los llamados **Temas**. Los Temas son plantillas estéticas que se aplican al sitio y que determinan su apariencia y funcionalidad. Mediante los temas se determinan los colores, fondos y tipos de letra, tanto de las entradas como de las páginas, pero también la ubicación y contenido de los menús de navegación del sitio.

El equipo de WordPress desarrolla temas propios cuyo nombre es ***Twenty***, seguido de otro número en inglés. Así el último tema desarrollado por el equipo, en el momento de escribir estás líneas, es el ***Twenty Fifteen*** y el siguiente se llamará ***Twenty Sixteen***. Los temas de WordPress están muy bien documentados y organizados y permiten, con habilidad, personalizar absolutamente todo su comportamiento.

También podemos elegir entre alguno de los miles de temas a nuestra disposición a través del menú **Apariencia-Temas** del panel de administración de WordPress (véase Apartado 8.3.4 *"Elección de temas"*, en la página 334). Aunque busquemos temas que nos parezcan muy originales, el problema es que en Internet hay millones de sitios que utilizan WordPress, por lo que habrá, casi seguro, otro sitio que utilice exactamente nuestro tema.

9.6.1.1 PERSONALIZAR LA APARIENCIA

Para que nuestro sitio aporte personalidad propia, es interesante dedicar tiempo a seleccionar un tema que se parezca a lo que deseamos. Después, lo lógico es personalizar la apariencia del tema.

La personalización del tema se puede realizar desde el menú **Apariencia** (*Appearance*)-**Personalizar** (*Customize*). En ese apartado, y de manera sencilla, podremos modificar la mayoría de aspectos del sitio. Hay que tener en cuenta que el panel de personalización varía según el tema elegido.

Figura 9.18: Aspecto del panel de personalización de temas para el tema *Twenty Fifteen*

Modificar el título y descripción

Bastará indicar el nuevo título y descripción. Una casilla nos permite indicar (en la mayoría de temas) si se muestra el título y/o la descripción en las páginas.

Colores

Muchos temas nos ofrecen elegir uno de entre varios esquemas de color básicos. Por ejemplo, el tema ***Twenty Fifteen*** por defecto aparece con colores muy claros, podemos cambiar a colores oscuros, morados, amarillos o rosas.

Además, dispondremos de varios selectores para modificar a nuestro gusto los colores del fondo, cabeceras, textos, etc.

Imagen de cabecera

Dependiendo del tema se coloca en unas zonas u otras. Se puede elegir de la biblioteca o subir otra imagen (en todo caso pasará a formar parte de la biblioteca).

Imagen de fondo

En este caso se utiliza como mosaico de relleno del fondo de las páginas del sitio. Se puede indicar la forma de repetirse.

Navegación

Permite indicar, sobre nuestros menús, en qué parte de la página aparecen. Más adelante se explica la creación de menús.

Widgets

Permite elegir qué widgets aparecen. Más adelante se explican los widgets.

Página de inicio

Permite indicar otra página de inicio.

9.6.1.2 PERSONALIZAR Y CREAR TEMAS

Todos los temas que hayamos descargado se encuentran en la ruta **wp-content/themes**, dentro del directorio raíz de WordPress. Realmente, un tema es un conjunto de archivos que tienen una estructura concreta y que se pueden, por supuesto con mucho cuidado, modificar. Es más, nosotros mismos, con las pautas apropiadas podemos crear nuestros temas.

Puesto que los temas están disponibles en la ruta mencionada en el párrafo anterior, existen muchas páginas que permiten descargar temas. Incluso hay temas ***premium***, de pago, que aportan funcionalidades avanzadas. Eso sí, siempre hay que tener muy en cuenta la fuente de la que descargamos el tema, ya que es perfectamente posible crear temas con código malicioso.

Una opción sencilla para personalizar es copiar un tema que nos guste, y que hayamos descargado previamente en el directorio **wp-content/themes**, y dotar a esa copia de las modificaciones deseadas.

Estructura de los temas

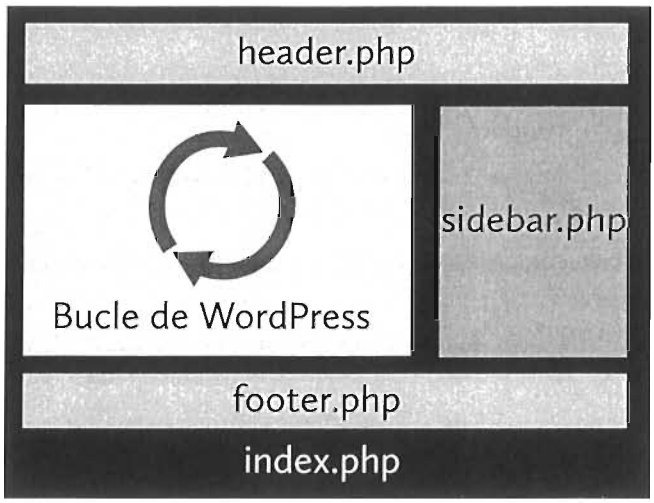


Figura 9.19: Estructura básica habitual de los temas de WordPress

Dentro de la carpeta **themes** hay un directorio por cada tema. Un directorio de tema contiene estos elementos:

- Directorio **css**. Hojas de estilo auxiliares.
- Directorio **fonts**. Tipos de letra utilizados por el tema.
- Directorio **images**. Imágenes utilizadas por el tema.
- Directorio **js**. Código JavaScript.
- **style.css**. Directorio obligatorio que contiene la hoja de estilos principal.
- **index.php**. Página de inicio por defecto. Archivo obligatorio que determina la forma de la página principal. Invocará a otros archivos php para formar el resultado final de la página.
- **home.php**. Plantilla para la página de inicio cuando es una página estática.
- **page.php**. Define la forma de las páginas estáticas.
- **header.php**. Define la zona de cabecera.
- **footer.php**. Define la zona de pie de página.
- **sidebar.php**. Define la barra lateral de tema, en la que generalmente se muestran los widgets y menús.
- **single.php**. Define cómo se muestra un post.
- **single-*tipo*.php**. Permite difinir la forma en la que se muestra un post de un tipo concreto. Por ejemplo, **single-astronomia.php** define la apariencia de los post de astronomía.
- **comments.php**. Define como se muestran los comentarios.
- **search.php**. Define la apariencia de la página de resultados de una búsqueda.
- **category.php**. Define la apariencia de la página de resultados de búsqueda de páginas por categorías.
- **tag.php**. Define la apariencia de la página de resultados de búsqueda de páginas por etiquetas.
- **taxonomy.php**. Define la apariencia de la página de resultados de búsqueda de páginas por taxonomía.
- **author.php**. Define la apariencia de la página de resultados de búsqueda de páginas por autor.
- **date.php**. Define la apariencia de la página de resultados de búsqueda de páginas por fecha.
- **attachment.php.** Determina la forma de mostrar componentes externos.
- **image.php.** Determina la forma de mostrar imágenes individuales.
- **functions.php.** Actúa como si fuera un plugin y se carga al iniciar WordPress. Define funciones que se utilizan en el tema y características para que el usuario pueda personalizar el tema desde los paneles de WordPress.

- **screenshot.png**. Imagen de muestra del tema que es la que utiliza WordPress en los paneles de selección de tema.
- **404.php**. Aspecto de la página de error de tipo *"Recurso no encontrado"* correspondiente al código de error http 404.
- **readme.txt**. Archivo de texto que sirve para documentar el tema.

Por supuesto, luego cada tema puede añadir más archivos si lo considera necesario. Podemos modificar los archivos que deseemos. Hay que tener en cuenta que los archivos php principales, especialmente **index.php**, manejan el llamado bucle de WordPress, que lo que hace es programar una o más estructuras que recorren los objetos del sitio web. Por ejemplo, el típico bucle es el que muestra cada post en la zona de contenidos de la página, cuyo código suele ser:

```
<?php
if (have_posts()):
   while(have_posts()):
      the_post(); //preparación del siguiente post
      //aquí va el código que se aplicará a cada post
   endwhile;
endif;
?>
```

Funciones de uso habitual en la creación de plantillas

Dentro del código de los archivos PHP se pueden utilizar numerosas funciones aportadas por el API de WordPress. Algunas de ellas son:

- **get_header()**. Incluye en el código el archivo de cabecera, ***header.php.***
- **get_footer()**. Incluye en el código el archivo de pie de página, ***footer.php.***
- **get_sidebar()**. Incluye en el código la barra lateral, ***sidebar.php.***
- **is_home()**. Verdadero si está mostrando la página de inicio.
- **have_posts()**. Devuelve verdadero si aún quedan entradas por recorrer.
- **the_post()**. Prepara el siguiente post en el bucle de lectura de entradas
- **the_content()**. Muestra el contenido del post actual.
- **the_title()**. Muestra el título del post.
- **the_ID()**. Identificador del post actual.
- **the_permalink()**. Muestra el permalink del post actual.
- **the_time()**. Muestra la fecha y hora del post.
- **the_author()**. Muestra el autor del post.
- **the_tags()**. Muestra las etiquetas relacionadas con el post.
- **the_category()**. Muestra las categorías relacionadas con el post.

- **edit_post_link()**. Enlace para editar el post.
- **comment_form()**. Formulario para escribir un comentario.
- **get_template()**. Nombre del tema actual.
- **get_template_directory_uri()**. Devuelve la URL al directorio del tema actual de WordPress.
- **get_stylesheet_uri()**. URL a la hoja de estilos (archivo **style.css**).
- **get_stylesheet_directory_uri()**. Devuelve la ruta al directorio de hojas de estilo.
- **get_site_url()**. URL a la raíz de WordPress.
- **get_posts()**. Obtiene un array de posts. Admite parámetros indicados en un array que permiten definir qué posts recogemos. Aplicando la función **setup_postdata()** a cada elemento del array resultado de gest_posts, podemos aplicar las funciones anteriores para obtener información de cada post.a la que se pasa un elemento de ese array, permite utilizar las funciones. Ejemplo de uso sencillo:

```
<?php
    $lista=get_posts();
    foreach ($lista as $post):
        setup_postdata($post);
    ?>
    <h1><?php the_title();?></h1>
<?php endforeach; ?>
```

El código del ejemplo muestra (en el archivo de plantilla en el que coloquemos ese código), el título de todos los posts. Podemos cambiar el código por ejemplo a:

```
<?php
$args=array(
    "orderby"=>"rand"
);
$lista=get_posts($args);
foreach ($lista as $post):
   setup_postdata($post);
   ?>
   <h1><?php the_title();?></h1>
<?php endforeach; ?>
```

Ahora las entradas se muestran de forma aleatoria. En la dirección:
https://codex.wordpress.org/es:Etiquetas_de_plantilla/get_posts

Está la información completa de esta interesante función, incluido el funcionamiento del array de parámetros.

9.6.2 MENÚS

9.6.2.1 INTRODUCCIÓN A LOS MENÚS

En WordPress los menús se han incorporado recientemente. Como CMS orientado a blogs, se entendía que personalizar los menús no era muy necesario. Pero WordPress, hoy en día, es un CMS que administra todo tipo de sitios y los menús son un elemento fundamental de organización.

Los menús establecen una jerarquía de enlaces que permiten recorrer los contenidos del sitio o bien enlazar a contenidos externos. Los menús, en definitiva, contienen enlaces.

Cada menú enlaza a una página, una entrada particular, una categoría, una etiqueta, una URL externa, etc. Inicialmente Wordpress crea un menú automático que va enlazando con cada página estática que hagamos. Si el sitio sirve mucho contenido, el menú automático será caótico; de ahí, que sea más aconsejable crear nuestra propia estructura de menús.

Siempre hay que tener en cuenta que los menús son uno de los elementos que aportan mayor usabilidad a un sitio web.

9.6.2.2 CREAR MENÚS PERSONALES

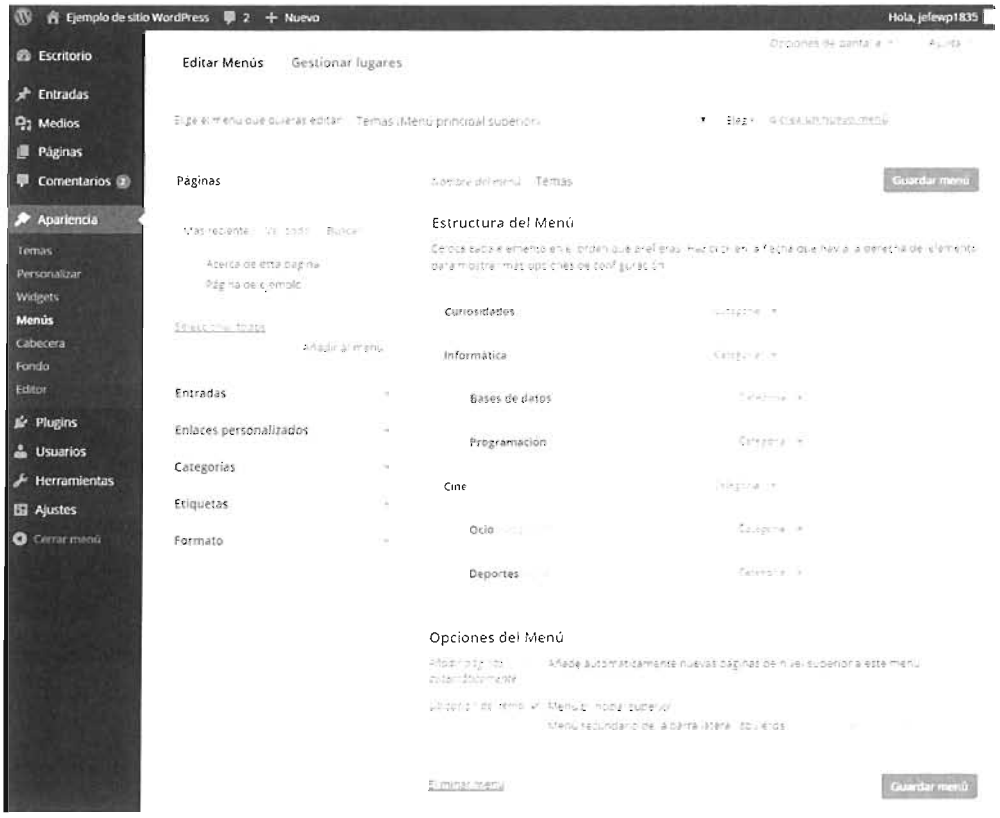


Figura 9.20: Panel de creación de menús. En este caso permite navegar por las categorías del sitio de forma jerárquica.

Los pasos son los siguientes:

[1] Desde el panel de control de administración ir a **Apariencia** (*Appearance*) - **Menús**.

[2] Escribir un nombre para el menú.

[3] Hacer clic en **Crear Menú**.

[4] Marcar los elementos a los que daremos enlace desde el menú. Los cuales pueden ser:

- **Páginas** (*Pages*), podemos marcar una serie de páginas a las que el menú permitirá acceder.
- **Categorías**. El enlace permitirá mostrar todas las entradas relacionadas con las categorías que marquemos.
- **Etiquetas**. Similar al anterior, pero utilizando nombres de etiqueta.
- **Enlaces personalizados**. Podemos indicar enlaces a URL que nosotros indiquemos y que pueden ser externas.

[5] Organizar jerárquicamente los contenidos del menú. Hay elementos que se pueden poner dentro de otros, para ello basta arrastrarles de forma adecuada (véase Figura 9.20). Lo mismo vale para determinar qué enlaces aparecen primero y qué enlaces aparecen al final; bastará arrastrarles a la posición adecuada.

[6] Elegir las opciones del menú

- Determinar si las nuevas páginas se añaden automáticamente a este menú. Puede ser una comodidad pero, en caso de tener muchas páginas, puede dar lugar a un número enorme de enlaces.
- Elegir en qué posición se coloca el menú. Dependiendo del tema habrá varias. En la Figura 9.20 se observa que el tema permite dos posiciones para el menú.

9.6.3 WIDGETS

Los widgets son componentes que se colocan en las barras laterales a fin de que los usuarios puedan acceder rápidamente a funcionalidades avanzadas del sitio como: buscar entradas por categorías, ver entradas recientes, listar entradas por fechas, etc. La barra lateral, en prácticamente todos los temas de WordPress, se rellena a través de Widgets.

La gestión de los Widgets se hace desde el apartado **Apariencia** (*Appearance*) - **Widgets** o desde el menú de personalización del tema actual (aunque es más cómoda la primera opción ya que el panel es más grande).

En el panel de configuración de Widgets aparecen todos los Widgets disponibles y cómo se ubican en la barra lateral u otras barras. Cada tema puede asignar una serie distinta de barras, por lo que las posiciones posibles para colocar widgets dependen del tema en uso.

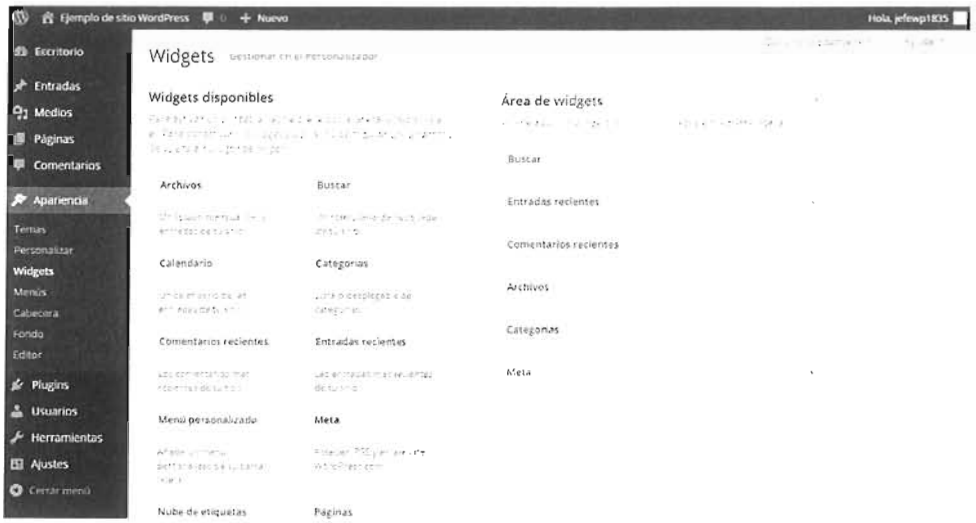


Figura 9.21: Panel de configuración de Widgets

Para añadir un Widget basta con seleccionarlo y, a continuación hacer clic en **Añadir widget** (***Add widget***). También se puede directamente arrastrar a la barra deseada y así, además, ubicar el widget en la posición deseada.

Sobre cada widget que hayamos colocado podemos: quitarlo haciendo clic en él y eligiendo **Borrar** (***Delete***), modificar su título en la barra y cambiar su orden en la barra simplemente arrastrando a otra posición.

Es fundamental tomar buenas decisiones con las opciones que aparecen en las barras, ya que son una de las claves de una navegación intuitiva por nuestro sitio web.

9.6.4 CAMBIAR LA PÁGINA DE INICIO

WordPress es muy conocido porque la página de inicio muestra la lista de entradas realizadas en orden inverso cronológico. Sin duda, es lo ideal cuando el sitio es un blog, sin apenas más contenido que los post que se hayan escrito.

Pero nada nos prohíbe que la página inicial sea diferente, en la que se muestra el sitio de la forma personal que nosotros deseemos. Con lo cual podemos crear una página de inicio que contenga, por ejemplo, menús a otras páginas como se explicó en el apartado anterior.

Para cambiar la página de inicio basta ir al apartado **Ajustes** (***Settings***)-**Lectura** (***Readings***) del panel de administración. Allí podemos seleccionar la página de inicio en el apartado ***Página frontal muestra*** eligiendo la página estática que queramos mostrar como página de inicio.

9.6.5 CAMBIAR LA PÁGINA DE ENTRADAS

La página de entradas es la que muestra los posts que hayamos publicamos. Por defecto es la portada del sitio. Pero, en el caso de que hayamos cambiado la página de inicio, entonces podremos indicar otra página para las entradas.

Para ello hay que seguir estos pasos:

[1] Crear una página con el nombre que deseemos para acceder a las entradas (por ejemplo, el propio nombre de ***entradas***).

[2] No se debe introducir ningún contenido, realmente se utilizará como plantilla para ver en ella las entradas.

[3] En el **Ajustes** (***Settings***)-**Lectura** (***Readings***) del panel de administración indicar esa página como página para mostrar las entradas. Lógicamente esa página debe estar enlazada desde algún menú para poder ver las entradas.

9.7 AÑADIR CONTENIDO EN DRUPAL

9.7.1 CREAR NUEVO CONTENIDO

Para añadir contenido en el caso de Drupal, desde el panel de administración disponemos del menú **Contenido** (***Content***). Desde ese panel podemos revisar el contenido publicado. Además un enlace llamado **Añadir contenido** es el encargado de generar nuevo contenido.

Figura 9.22: Panel de contenido de Drupal

Al igual que WordPress, la versión Core de Drupal tiene solo dos tipos de contenidos o nodos (se pueden añadir más): **artículos** (equivalentes a los post o entradas de WordPress) y **páginas básicas** (equivalentes a las páginas de WordPress). Realmente en Drupal apenas hay diferencia, pero las primeras se organizan, por defecto, cronológicamente al estilo de WordPress.

Tras elegir el tipo de contenido, un panel nos permite definir el mismo. En dicho panel podemos:

- Indicar el título. Es obligatorio hacerlo.
- Indicar etiquetas (tags) que faciliten posteriormente su búsqueda.

- Indicar el cuerpo (body). Es el texto del artículo o página. Es mucho más parco que el panel de WordPress. Permite incorporar etiquetas HTML, pero no dispone de modo visual de trabajo. Si tiene en cuenta los saltos de párrafo que hagamos en el panel para dividir nuestro texto en párrafos en el formato final.

  Una lista permite elegir el texto que podemos introducir, puede ser **HTML filtrado** (solo acepta algunas etiquetas HTML), **HTML completo** (***full***) que acepta todo tipo de HTML y **texto sin formato** que no permite introducir HTML. En los tres modos, convierte automáticamente las URL que introduzcamos a formato HTML (mediante el elemento <a>).

  Se pueden añadir más tipos de texto posible para escribir desde **Configuración-Formatos de texto**, panel que permite añadir o modificar estas tres posibilidades de escribir texto.

- Añadir una imagen (tamaño máximo de 2 MB).

- Indicar opciones:
  - Texto de revisión. Un texto que explique la modificación que hemos hecho en el artículo o página para otros editores.
  - Indicar un texto para la URL. Al estilo de los permalinks de WordPress.
  - Indicar si permitimos comentarios
  - Modificar el autor
  - Indicar dónde publicamos el artículo. **Publicado** significa que se publica con normalidad. **Promovido a portada** le sube a la portada.

9.7.2 MODIFICAR EL CONTENIDO

Para modificar el contenido basta, desde el panel de administración, ir al apartado **Contenido**, seleccionar el contenido y luego editarlo.

Pero podemos también acceder directamente al contenido y hacer clic en **Editar**, enlace que aparece si nos hemos autenticado en el sitio.

9.7.3 ALIAS DE URL

Los alias permiten acceder a un contenido con una URL más sencilla. Se pueden indicar cuando se añade o edita contenido. Pero hay un panel que centraliza los alias en uso, se accede a él desde **Configuración-Alias de URL** en el menú de administración.

9.7.4 CREAR NUEVOS TIPOS DE CONTENIDO

En Drupal se pueden modificar los tipos de contenido existentes y crear nuevos. Esto es posible desde el panel de administración eligiendo el menú **Estructura** y luego **Tipos de contenido**.

Figura 9.23: Panel de tipos de contenido en Drupal

Desde ese panel podemos añadir o editar tipos de contenido existente. En ambos casos un nuevo panel nos permite perfilar el contenido añadido. El panel es muy claro pero tiene muchísimas opciones. Se puede especificar exactamente qué elementos forman parte de cada tipo de contenido, cómo funcionan y, con paciencia, hacer un tipo de contenido totalmente personalizado.

La edición de los tipos de contenido consta de 5 elementos:

- **Panel Editar**. Con la información básica del tipo de contenido:
  - Nombre del tipo de contenido (por ejemplo, ***artículo***)
  - Descripción
  - Opciones generales: Donde se publicará, como se envía el contenido (si se previsualiza antes o no), si se muestra la información del autor, si se admiten comentarios, en qué menú aparece
- **Gestión de campos**. Son los tipos de datos que se pueden almacenar tipo de contenido. Por ejemplo, los artículos tienen los campos ***título***, ***tag***, ***body*** (cuerpo del artículo) e ***imagen***. Podemos añadir más o eliminar los que queramos para cada tipo.
- **Gestionar presentación**. Permite indicar cómo se muestra cada uno de los campos. Podemos personalizar el orden en el que se muestran (se pueden arrastrar los campos para indicar cuál sale primero), la forma en la que lo hacen (si se muestra sin un texto de etiqueta, o si se muestra dicho texto por encima o alineado), el formato que utilizan (texto completo, resumen, etc.) y determinar opciones avanzadas (por ejemplo, en las imágenes, podemos decidir el tamaño que se utilizará).

  Lo mejor es probar y comprobar lo que ocurre al cambiar cada elemento, con mucha paciencia.
- **Campos de comentarios**. Podemos admitir comentarios y al igual que en el contenido en sí, indicar qué campos recogemos y cómo los presentamos.

9.7.5 ESTILOS DE IMAGEN

A la hora de indicar (en los tipos de contenido) qué formato usa la imagen, inicialmente Drupal permite tres opciones:

- **Thumbnail**. Imagen diminuta (por defecto 100x100 píxeles).

- **Medium**. Imagen media (320x200).
- **Large**. Grande (480x480).

Podemos cambiar estos tres estilos por defecto con ayuda del panel de estilos de imagen. disponible en **Configuración-Estilos de imagen**. Podemos añadir nuevos estilos, a los que además podemos aplicar efectos en la imagen como redimensionar, recortar, giros, efectos de color, escalado, etc. Que son los que permiten dejar la imagen como indiquemos. El habitual es escalar, para dejar la imagen al tamaño que indiquemos.

9.7.6 TAXONOMÍAS

Las taxonomías en cualquier CMS permiten la clasificación del contenido. Drupal posee también gestión de taxonomías, porque también suele tener que gestionar sitios con miles de entradas de contenido.

En Drupal las taxonomías se componen de **vocabularios** y **términos**. Los vocabularios sirven para agrupar términos relacionados. Así ***astronomía*** puede ser un vocabulario al que pertenecen los términos ***planetas, satélites, estrellas, espacio*** y ***galaxia.***

La taxonomía en Drupal se gestiona desde el enlace **Estructura** del menú de administración

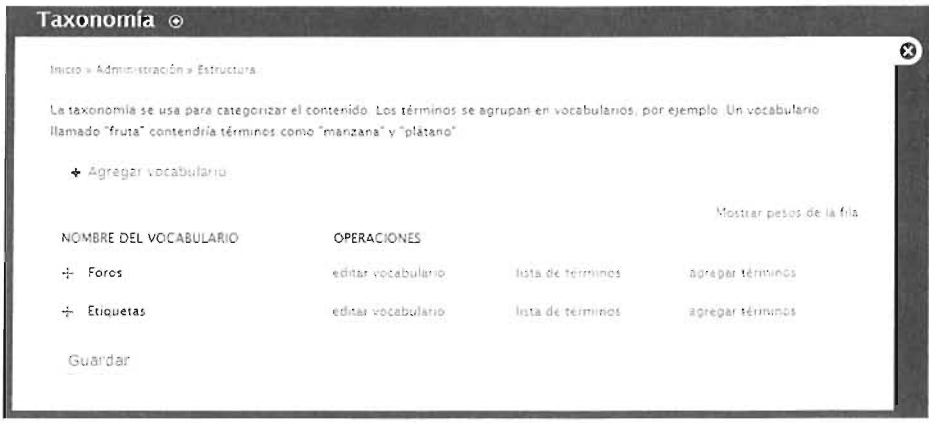


Figura 9.24: Panel de gestión de la taxonomía de Drupal

9.7.6.1 AÑADIR VOCABULARIOS

Desde el panel de taxonomía simplemente se hace clic en añadir vocabulario. Bastará con indicar el nombre y la descripción para crearlo. Una vez creado, se puede editar. Lo cual nos permite:

- Ver y modificar la lista de términos asociados al vocabulario.
- Modificar el nombre y/o la descripción.
- Modificar los campos que ese vocabulario mostrará cuando deseemos añadir un término (normalmente aparece solo el nombre y la descripción del término).

- Modificar la presentación de los términos de este vocabulario cuando se muestren en la página final.

9.7.6.2 AÑADIR TÉRMINOS

Para añadir términos a un vocabulario podemos hacerlo si estamos editando el vocabulario desde la pestaña **Lista** o desde el panel de Vocabularios, ya que cada vocabulario dispone directamente de un enlace para **añadir términos**.

En ambos casos aparece un panel con el que podemos añadir el nombre del término (es obligatorio rellenarlo) y su descripción. La descripción puede incluso incorporar HTML. Podemos indicar un alias para poder establecer una URL más clara para llegar a ese término desde un navegador.

Sobre cada término podemos indicar que es hijo de otro a través del apartado ***relaciones***. Así, podemos indicar que **Plutón** es hijo de **planetas**. Eso permite establecer una jerarquía que enriquecerá la semántica del sitio y facilitará la realización de búsquedas de contenido.

9.7.6.3 ASIGNAR UNA TAXONOMÍA A UN TIPO DE CONTENIDO

En el Apartado 9.7.4 *"Crear nuevos tipos de contenido"*, en la página 391 vimos cómo se pueden crear tipos de contenido a medida. Lo interesante de las taxonomías es que se pueden asignar a un tipo de contenido y hacer que al añadir contenido de ese tipo, aparezca un campo en el que se pueden indicar términos de esta taxonomía.

Para ello hay que editar el tipo de contenido y en el apartado de gestión de campos agregar un nuevo campo al que se le indicará:

- El nombre que deseemos (ese nombre será lo que vea el usuario como título del control que tiene que rellenar).
- Nombre de sistema. El nombre único que le asignar Drupal a ese control.
- En tipo de campo hay que elegir **Referencia a término** que es el adecuado para que al añadir contenido indiquemos términos taxonómicos. Más tarde deberemos editar este apartado y hacer clic en el enlace para indicar a qué vocabulario nos estamos refiriendo.
- Finalmente podemos elegir como tipo de control **Autocompletar término,** que permite escribir directamente el término. También podemos indicar un **Cuadro desplegable** y entonces, al añadir contenido, solo se podrá elegir uno de entre los términos existentes.

Conviene editar el campo (desde el propio panel de tipo de contenido, en el apartado de **gestión de campos**) para modificar las propiedades.

Al editar el campo podemos indicar opciones avanzadas. Por ejemplo, si admitimos más de un término separado por comas o si es obligatorio indicar al menos un término.

9.7.7 CONCLUSIONES SOBRE LA CREACIÓN CONTENIDOS EN DRUPAL

Indiscutiblemente Drupal permite personalizar de forma espectacular la forma de crear contenidos. Lo malo es la edición del contenido en sí, que no es nada cómodo ni versátil. Aunque podemos instalar módulos que nos faciliten la trea. Así el **CK Editor** es un módulo casi obligado para escribir contenidos más cómodamente.

WordPress admite solo dos tipos de contenido (posts y páginas), Drupal infinitos: dos en principio, pero podemos crear los que queramos y personalizarlos para que hagan exactamente lo que queremos. A muchos usuarios no les gusta esta versatilidad, pero los desarrolladores (eso sí conociendo más a fondo los entresijos de Drupal) sacan mucho jugo de ella.

9.8 PERSONALIZACIÓN DE LA APARIENCIA EN DRUPAL

9.8.1 MENÚS

La navegación por las páginas de un sitio se hace a través de menús. Los menús se componen de enlaces y también pueden contener otros tipos de elementos (como los bloques). La gestión de los menús en Drupal se hace a través del panel del mismo nombre, disponible al hacer clic en **Estructura**, dentro del menú de administración.

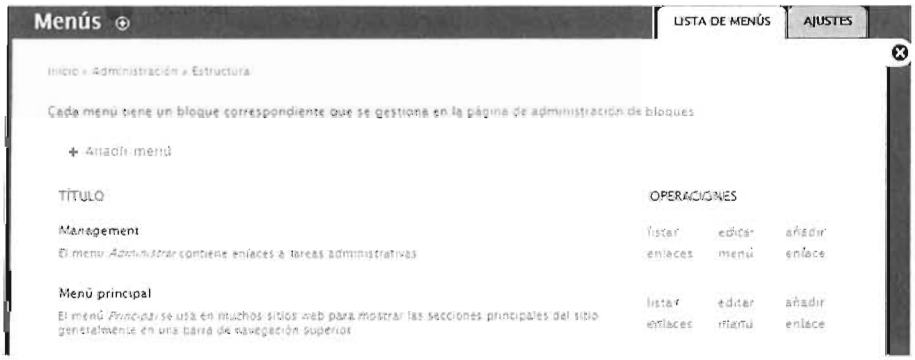


Figura 9.25: Panel de gestión de menús de Drupal

9.8.1.1 MODIFICAR Y AÑADIR ENLACES A LOS MENÚS

Estando en el panel de gestión de menús. Podremos ver y modificar sus enlaces a través de **listar enlaces**. Aparece la lista jerárquica de enlaces. Podemos cambiar el orden y su jerarquía simplemente arrastrando desde el icono ✣.

Además, podemos editar cada enlace para modificar su título, la ruta a la que nos dirige el enlace, si tiene enlace padre (lo que le convertiría en parte de un submenú) y el peso, mediante el cual se establece el orden en el que aparece el enlace en el menú.

Al añadir un enlace tendremos el mismo panel, que nos permitirá establecer las mismas opciones que al editar un enlace.

Figura 9.26: Enlaces del menú principal típico de las instalaciones de Drupal Core.

9.8.1.2 CREAR MENÚS PROPIOS

Desde el panel de gestión tenemos la opción de crear un menú al cual le podemos añadir los enlaces que deseemos.

9.8.2 BLOQUES Y REGIONES

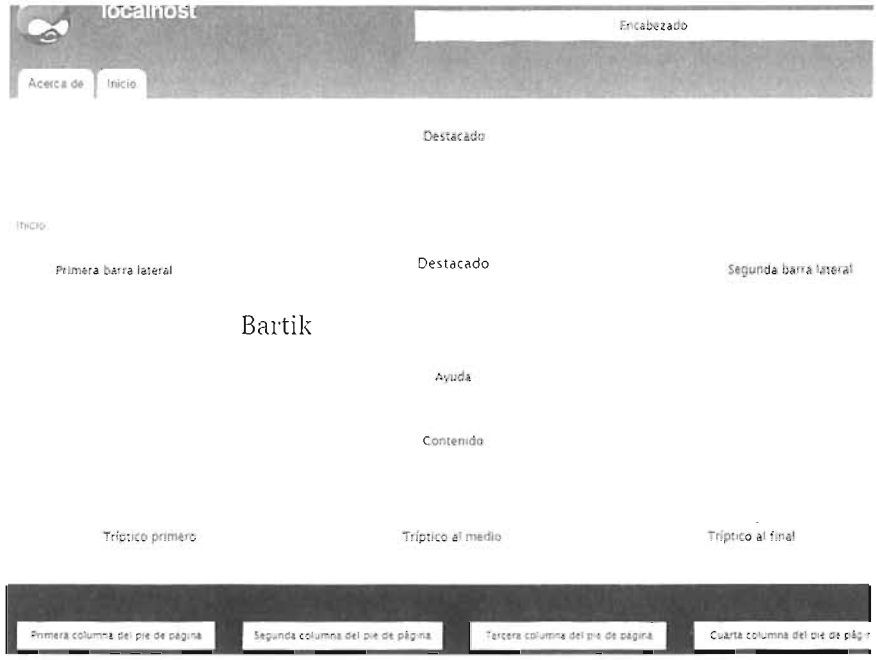


Figura 9.27: Demostración de los bloques que forman parte del tema Bartik del núcleo de Drupal

Los temas definen **regiones** en el tema, áreas diferenciadas en la plantilla estética del sitio. Cada una de esas regiones puede mostrar uno o más elementos. Esos elementos se conocen como **bloques**.

Los bloques son un término de Drupal que se refiere a un objeto de la página capaz de mostrar contenido, enlaces, menú, código... En definitiva, es un elemento que se verá en la página y que se puede colocar en la región que deseemos. De hecho, los menús también se consideran bloques.

La gestión de los bloques se realiza desde **Estructura-Bloques** (menús del panel de administración) ahí aparecen los bloques disponibles y la región a la que se aplican.

El botón **Demostrar regiones de bloques** muestra una imagen (Figura 9.27) con todas las regiones en el tema elegido. La imagen indica el nombre y posición de la región. Es muy conveniente hacer este paso para conocer la ubicación de cada región.

9.8.2.1 AÑADIR BLOQUES

Desde el panel de administración de bloques podemos **Añadir bloques** haciendo clic al enlace homónimo. Al añadir debemos indicar lo siguiente:

- Nombre y descripción del bloque.
- Cuerpo del bloque. Que puede contener código HTML.
- Región en la que se colocará el bloque (en cada tema).
- Páginas en las que se mostrará el bloque (en principio en todas, pero podemos elegir en cuales).
- Tipos de contenido en los que se muestra el bloque (si no se indica, se muestra en todos).
- Roles de usuario a los que se muestra el bloque.
- Finalmente si permitimos a los usuarios personalizar este bloque.

9.8.3 TEMAS EN DRUPAL

En el Apartado 8.7.2 *"Instalar temas"*, en la página 344 , se explicó como instalar nuevos temas en Drupal. Ahora se explicará como personalizar el tema elegido.

9.8.3.1 AJUSTAR EL TEMA

En el panel **Apariencia-Temas** podemos ver la lista de temas disponibles para ser el tema actual. Todos ellos son temas ya instalados. En **Ajustes** (pestaña de la parte superior), podemos tomar las siguientes decisiones:

- Elementos que aparecerán en el tema: logos, imágenes, menús, etc. Ajustes que nfluyen a todos los temas.
- Podemos elegir el nombre del tema que queremos modificar y tendremos más opciones (éstas ya dependerán del tema concreto). Muchos temas admiten modificar sus colores.

9.8.3.2 PERSONALIZAR EL TEMA

Como en WordPress, podemos personalizar completamente el tema. En Drupal, lo aconsejable es copiar un tema existente y modificar su código, ya que la organización es más jerárquica.

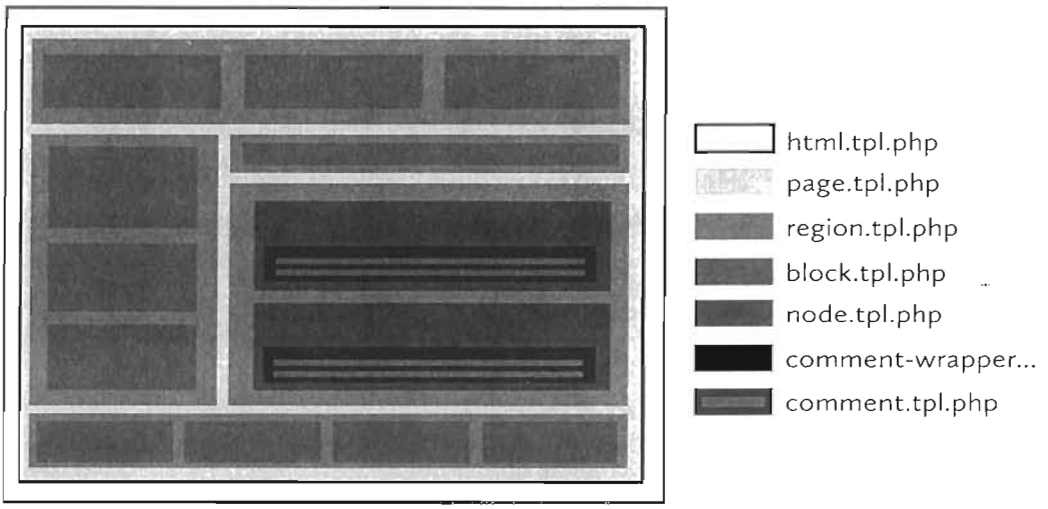


Figura 9.28: Arquitectura de las plantillas para los temas de Drupal

Los temas se sitúan, como ya se ha comentado en la ruta **sites/all/themes** dentro del directorio de Drupal. Hay una serie de archivos que conviene comentar:

- Archivo con el nombre del tema y extensión ***info,*** por ejemplo ***bartik.info***. Contiene la lista de elementos y detalles que tienen que ver con el tema: nombre, localización de los archivos CSS y JavaScript, regiones que utiliza el tema, etc.

  Ejemplo de contenido:

```
name = Tema propio
description = Ejemplo de tema
core = 7.x
stylesheets[all][] = style.css
; stylesheets[print][] = print.css

regions[header] = Cabecera
regions[menu] = Menu
regions[content] = Contenido
regions[right_a] = Barra lateral A
regions[right_b] = Barra lateral B
version = "0.1"
project = "drupal"
; 12/7/2015
datestamp = "1436611165"
```

Solo con un archivo de este tipo, el tema estará disponible en el administrador de temas de Drupal. Si activamos ese tema y vamos a **Estructura-Bloques**, veremos que ya se reconocen las regiones indicadas.

El contenido de las mismas se especifica en los archivos que explicamos a continuación; pero si no los especificamos, el menú de **Estructura-Bloques** nos permitiría ya, directamente lo que va en cada apartado (se basará en el tema por defecto de Drupal).

- **html.tpl.php**. Plantilla que se encarga de la cabecera de las páginas y los ajustes generales del cuerpo de la pagina. Muchos temas no la incluyen y en ese caso funcionarán el archivo (con el mismo nombre) existente en la ruta (partiendo de la raíz) **modules/system.** Utiliza instrucciones PHP del tipo **<?php print**, seguido del nombre de una variable. Por ejemplo, `<?php print $styles; ?>`, hace que en esa posición se escriba el código HTML que carga las hojas de estilo.
- **page.tpl.php**. Plantilla general de las páginas del sitio. Se encarga de generar el código de todas las regiones de la página. Por ejemplo, la instrucción:

  ```
  <?php print render($page['rightbar_B']) ?>
  ```

  se encarga de generar el contenido de la región correspondiente a la segunda barra lateral.
- **node.tpl.php.** Código base para mostrar un nodo en el sitio Drupal.
- **region.tpl.php**. Código base para cada región del tema.
- **style.css**. Hoja de estilos general. Al menos tiene que estar este archivo. Pero suele haber más, normalmente en el directorio llamado CSS.

Los archivos de plantillas utilizan diversas variables, algunas de ellas son:

- **$base_path**. Ruta al directorio raíz de Drupal.
- **$front_page**. URL a la página de portada del sitio.
- **$directory**. Ruta al directorio del tema actual.
- **$logged_in**. Devuelve verdadero si el usuario está conectado.
- **$is_admin**. Devuelve verdadero si el usuario tiene acceso a las páginas de administración.
- **$logo**. Ruta a la imagen que se utiliza como logotipo del sitio, según se definió en la página de ajustes del tema.
- **$site_name**. Nombre del sitio.
- **$title**. Título del nodo.
- **$content**. Texto que representa a un nodo. Es un array con todos los elementos del nodo, pero puede ser escrito de golpe con `render($content)`. Si queremos renderizar solo un campo concreto del nodo se hace con `render($content["campo"])`. Así tendremos por ejemplo que `$content['links']` escribe los enlaces del nodo
- **$uid.** Identificador del autor.

- **$user_picture**. Imagen del autor.
- **$name.** Nombre del autor.
- **$date**. Fecha de creación del contenido.
- **$page**. Array que representa la página actual. A través del él podemos acceder a las diferentes regiones, por ejemplo: `page["right_a"]` accede a la región identificada como `right_a` (barra lateral A).
- **$node.** Objeto que representa el nodo completo, no es recomendable su uso. Aunque a través de esta variable (que es un array) podemos acceder a todos los elementos del nodo.
- **$type**. Tipo de nodo.
- **$main_menu.** Array que representa cada enlace del menú principal.
- **$comment_count**. Número de comentarios que tiene el nodo.
- **$submitted**. Información de nombre y fecha del elemento.
- **$id**. Posición del nodo, se incrementa con cada nodo que se muestre.

En cualquier caso, conviene observar un tema ya existente y examinar cómo funcionan las páginas de plantilla y variables. Al principio conviene copiar un tema existente y modificar sus archivos. Al final el funcionamiento de los temas personales es tan fácil (o más incluso) que personalizar temas con WordPress.

9.9 PRÁCTICAS RESUELTAS

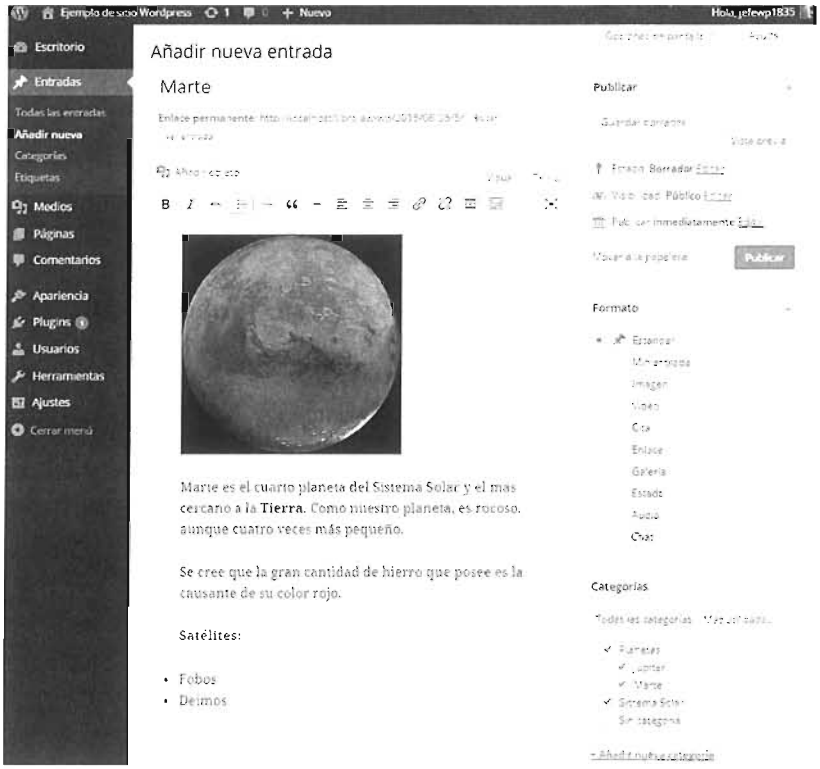
Práctica 9.1: Crear contenidos en WordPress

- Crearemos dos entradas nuevas de contenido. Una hablará del planeta **Júpiter** y otra de **Marte**.
- Conseguiremos utilizar como categorías los términos: **Sistema Solar**, **Planetas**, **Venus** y **Marte**. Haremos que Venus y Marte sean términos hijos de Planetas.
- Pon una foto de Júpiter y otra de Marte y escribe lo que quieras de cada planeta.
- En ambos casos, usando una lista encabezada por puntos, escribe la lista de sus satélites (para el caso de Júpiter escribe solo 4).

SOLUCIÓN: PRÁCTICA 9.1

[1] Elimina las entradas de WordPress de ejemplo iniciales. Para ello entra al apartado entradas del menú de administración, selecciona las que tengas (al menos dos escribe al principio WordPress) y en acciones en lote elige **Mover a la papeler**. Después pulsa **Aplicar**.

[2] Elige Añadir nueva entrada en el apartado de entradas del menú de administración.



[3] Aparece el panel de nuevas entradas, en él escribe:

- Como título: **Marte**.

- Pulsa añadir objeto y añade una foto de Marte que, previamente, habrás descargado de Internet. Cuando la añadas pon como Título y texto alternativo: Planeta Marte.
- Antes de publicar, consigue que aparezcan las categorías tal cual se pide en el enunciado.

[4] Haz lo mismo con Júpiter escribiendo el texto, imagen, propiedades de la imagen y marcando la categoría a la que pertenece.

[5] Comprueba el resultado

Práctica 9.2: Crear contenidos personalizados en Drupal

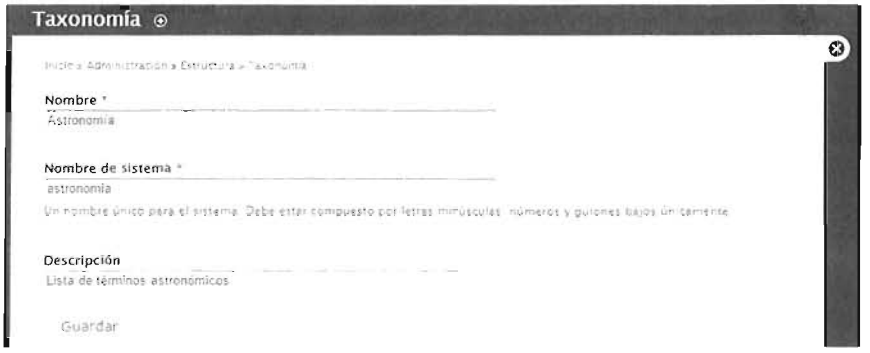
- Crearemos dos entradas nuevas de contenido. Una hablará del planeta **Júpiter** y otra de **Marte**.
- Como vamos a tener numerosas entradas de Astronomía, hemos decidido crear un **tipo de contenido** que se llamará **Artículo Astronómico** en el que podremos indicar el título del artículo, el texto del mismo que puede llevar solo HTML filtrado (no se permite HTML completo), una posible imagen y podremos indicar palabras clave que previamente habremos almacenado en el vocabulario Astronomía que se explica en el punto siguiente.
- Las palabras clave pertenecerán a un vocabulario llamado Astronomía en el que insertaremos las palabras. Por ahora introduciremos los términos: **Sistema Solar**, **Planetas**, **Venus** y **Marte**. Haremos que Venus y Marte sean hijas de Planetas.
- Para facilitar la escritura de los artículos, instala el módulo **CKEditor.**
- Escribe los dos artículos asegurándote que eliges el tipo de contenido creado para este tipo de artículos. Pon una foto de Júpiter y otra de Marte y escribe lo que quieras de cada planeta.
- En ambos casos, usando una lista encabezada por puntos, escribe la lista de sus satélites (para el caso de Júpiter escribe solo 4).

SOLUCIÓN: PRÁCTICA 9.2

[1] Empezamos creando la nueva taxonomía. Desde el menú de administración elegimos **Estructura-Taxonomía**.

[2] Hacemos clic en **Agregar vocabulario**.

[3] Indicamos como nombre **Astronomía**. En descripción: ***lista de términos astronómicos***. Además, hacemos que el nombre en el sistema sea **astronomia** (no puede llevar tilde)



[4] Hacemos clic en guardar e inmediatamente en **Agregar términos**.

[5] Sin usar el campo de descripción, añadimos el primer término: **Sistema Solar**. Pondremos como alias de URL: **sistema-solar**. Hacemos clic en **Guardar**.

[6]

Se guardará el término y podremos añadir el siguiente término. Haremos lo mismo para el resto, a todos daremos alias y a Marte y a Júpiter les haremos hijos de planetas

[7] Nos dirigimos a **Estructura-Tipos de Contenido**. En el panel siguiente hacemos clic en **Añadir tipo de contenido**.

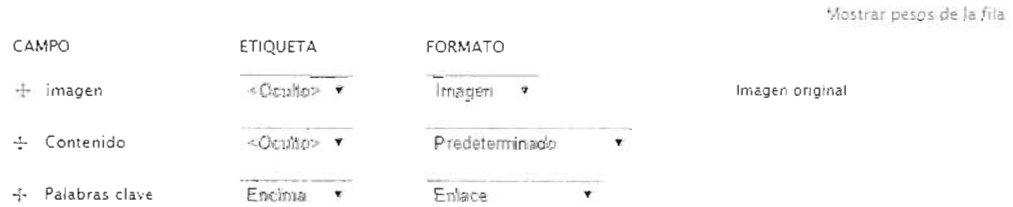
[8] En el panel de tipo de contenido indicamos:

- **Nombre**: Artículo Astronómico.
- **Nombre de sistema**: articulo_astronomico.
- **Descripción**: Tipo de contenido para artículos astronómicos.
- **Opciones de formulario de envío**:
  - Etiqueta de campo de título: **Título**
  - **Resto de opciones: Tal cual están**

[9] Hacemos clic en **Guardar y añadir campos.**

[10] Aparece el panel de campos. En él hacemos:

- Editamos el campo Body, y lo llamamos Contenido. Lo marcamos como campo obligatorio
- Añadimos un campo llamado **Palabras clave**, que es una referencia a término. Elegiremos como tipo de control, una lista desplegable. Tras guardar, elegimos el vocabulario **Astronomía**. Permitiremos un número ilimitado de palabras.
- Añadimos un campo de imagen. Marcamos el tipo de contenido como imagen. En sus propiedades marcamos activar campos **Alt** y **Titulo** (para que la imagen pueda ser buscada y etiquetada) y dejamos un a anchura máxima de 640x480.



- En el apartado de **Presentación** (imagen anterior), haremos que la imagen se pida antes que los demás datos (simplemente la arrastramos encima de los otros campos). La etiqueta de la imagen será oculta.

[11] Instalamos el módulo CKEditor, desde la URL https://www.drupal.org/project/ckeditor

- Podemos ir a **Módulos-Instalar nuevo módulo** y elegir el archivo comprimido del CKEditor que previamente habremos descargado
- En Linux otra opción, incluso más rápida es situarnos en el directorio de los módulos (lo suponemos colocado en **/var/www/libroiaw-dp/sites/all/modules**, coherentemente con la Práctica 8.4 *"Instalación de Drupal en un servidor Linux"*, en la página 354) y desde ahí descargar y descomprimir el archivo del CK Editor:

```
$ cd /var/www/libroiaw-dp/sites/all/modules
$ wget http://ftp.drupal.org/files/projects/ckeditor-7.x-1.16.tar.gz
$ tar vzxf ckeditor-7.x-1.16.tar.gz
$ rm ckeditor-7.x-1.16.tar.gz
```

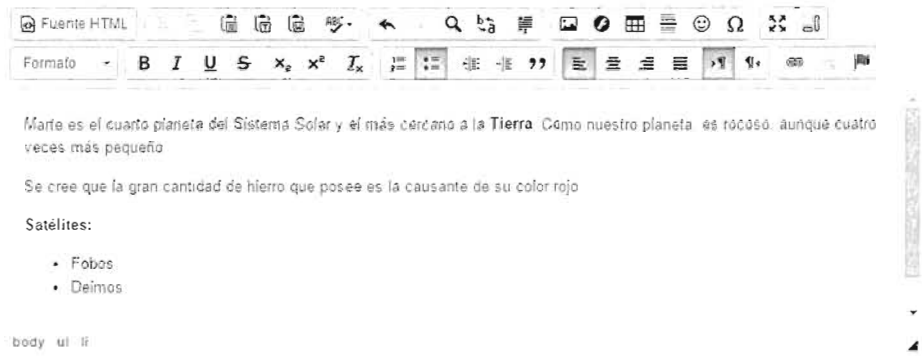
- Finalmente en Drupal, vamos a Módulos, buscamos el módulo **CKEditor** (estará al final) y lo activamos.

[12] Busca una imagen de Marte y de Júpiter (que no sean excesivamente grandes y guárdalas en tu ordenador.

[13] Vamos a **Contenido-Agregar Contenido-Artículo Astronómico**

[14] Rellenamos de esta forma:

- **Título:** El planeta Marte.
- **Imagen:** Elegimos la imagen de Marte descargada.
- **Contenido:**



[15] Haz los mismos pasos para Júpiter. Invéntate el contenido y coloca la imagen de Júpiter

[16] Comprueba el resultado.

Práctica 9.3: Personalizar tema en WordPress

- Utilizando el tema Twenty Fifteen de WordPress, personalízalo para que presente la apariencia de la siguiente imagen:

- Como detalles ten en cuenta el color negro de la barra lateral y el gris del fondo del cuerpo.
- Se muestran las entradas en la zona de contenido
- La barra lateral solo muestra el contenido del menú de información con los enlaces a sus páginas estáticas (Práctica 9.3), el cuadro de búsqueda y un cuadro de selección de categorías en forma de cuadro desplegable.

SOLUCIÓN: PRÁCTICA 9.3

[1] Lo primero es ir al apartado temas y comprobar que el tema en uso es **Twenty Fifteen**, de no ser así, hay que activar ese tema.

[2] Ir a **Apariencia-Personalizar** desde el panel de administración, abrir el apartado **Colores** y elegir negro como color de fondo de cabecera y barra lateral y gris como color de fondo. Hacer que no aparezca ni imagen de fondo ni imagen de cabecera.

[3] En el apartado **Navegación** hacer que no se muestre ningún menú (los menús les manejaremos desde el apartado Widgets)

[4] Elimina todos los Widgets y deja el Widget de **Buscar**. Añade (botón **Añadir un widget**) el widget de Categorías haciendo que se muestre como desplegable y que se muestren la can-

tidad de entradas. Añade también el widget **Menú personalizado** y añade el menú creado en la Práctica 9.3 para mostrar las páginas estáticas de esa práctica.

[5] Guardar y comprobar

Práctica 9.4: Crear páginas estáticas en WordPress

- Elimina todas las páginas estáticas que tengas
- Crea una página estática que se llame **Sobre mi** en la que hables de ti y otra que se llame **Sobre mi localidad** en la que hables de la localidad donde vives. No admitas comentarios.
- Consigue que el slug hacia ellas sea **/sobre-mi** en el caso de la primera página y **/mi-ciudad** en la segunda.
- Haz un menú que se llame información personal y que permita acceder a ambas páginas. Consigue que ese menú esté accesible desde la página principal, en la página principal debes conseguir que aparezca ese menú.

SOLUCIÓN: PRÁCTICA 9.4

[1] Ve al apartado **Páginas.** Selecciona todas las páginas que haya, marca **Mover a la papelera** en la lista de **Acciones en lote** y Aplica la acción.

[2] Haz clic en el botón añadir nueva. Coloca el título ***Sobre mi*** y el texto que te parezca bien escribir.

[3] En el apartado **Opciones de pantalla** (arriba de la pantalla), haz que se muestre el panel de comentarios. Elimina la casilla Permitir comentarios.

[4] Publica la página.

[5] Realiza las mismas acciones con la página *Sobre mi localidad*.

[6] En el menú de administración vete al apartado **Apariencia-Menús** (ver imagen anterior).

[7] Haz clic en **Crea un nuevo menú**, ponle como nombre **Información personal.**

[8] En el panel **Páginas**, elige las dos páginas que has creado y pulsa en Añadir al menú. Observa que aparezcan en el menú.

[9] Marca la casilla **Menú principal** y guarda el menú.

[10] Comprueba el resultado en la portada del sitio.

Práctica 9.5: Crear tema propio en WordPress

- Tomando como referencia la imagen superior crea un tema desde cero, escribiendo tú todo el código.
- La barra superior de color negro mide 60 píxeles de altura, contiene cuatro enlaces:
  - El título ***Mi blog***, es un enlace a la página de inicio.
  - A la página sobre mi, página estática creada en la Práctica 9.3.
  - A la página sobre mi localidad.

- Al panel de administración.

- Después se muestran los post realizados. En la imagen se muestran los post que se suponen escritos tras realizar la Práctica 9.1. En los posts, el título sale centrado dentro de una barra de color gris, la imagen del contenido sale alineada. No se muestran más datos de los post y se mostrarían todos los post escritos.
- Si cambiamos la ruta y ponemos el permalink a un post concreto, entonces se mostrarán solo los datos de ese post
- Cuando el tema funcione, consigue que WordPress en el cuadro de temas muestre una previsualización del mismo.

SOLUCIÓN: PRÁCTICA 9.5

[1] Crea una nueva carpeta dentro de **wp-content/themes** en la raíz de WordPress. La llamaremos **Mi tema**.

[2] En un editor de texto escribe el siguiente código CSS y guardarle bajo el nombre **style.css**.

```
#cabecera{
    position:fixed;
    left:0;
    top:0;
    width:100%;
    height:60px;
    background-color:black;
    color:white;
}
#titulo{
    position:absolute;
    top:0;
    left:0;
}
#menu{
    position:absolute;
    top:30px;
    right:0;
}
a{
    color:white;
    text-decoration: none;
}
a:hover{
    text-decoration: underline;
}
```

```
#cuerpo{
    position:fixed;
    left:0;
    top:60px;
    width:100%;
    bottom:0;
    background-color:white;
    color:black;
    overflow-y: scroll;
}
#cuerpo h1{
    margin: 0;
    text-align: center;
    background-color:gray;
    color:white;
}

img{
    float:left;
    margin-right:5px;
}
```

Con esto hemos decidido la apariencia de las páginas. La barra superior será una capa con identificador cabecera, dentro se coloca el título y el menú, después otra capa mostrará el contenido en el que ya hemos indicado cómo queremos mostrar los títulos.

[3] Lo mínimos ahora sería hacer la página **index.php** que contiene el bucle de WordPress de recogida de entradas. Sin embargo, para mayor claridad dividimos el código entre los archivos de cabecera, pie y de bucle (ver Figura 9.19 *"Estructura básica habitual de los temas de WordPress"*, en la página 383). El código del archivo **header.php** es:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="<?=get_stylesheet_uri()?>">
</head>
<body>
<div id="cabecera">
    <div id="titulo">
        <h1><a href="<?=get_site_url()?>">
                Mi blog</a></h1>
    </div>
```

```
    <div id="menu">
          <a href="<?= get_site_url() ?>/sobre-mi">
              Sobre mi</a>
          | <a href="<?= get_site_url() ?>/mi-localidad">
              Sobre mi localidad</a>
          | <a href="<?= get_site_url() ?>/wp-admin">
              Administración</a>
      </div>
</div>
```

[4] El código del archivo **footer.php** es más sencillo. La única razón para hacerlo es que sea más legible el código de **index.php**. Este es el código del archivo **footer.php**:

```
</body>
</html>
```

[5] Y, finalmente, el código del archivo **index.php**. Éste invoca a los anteriores archivos y lee cada post:

```
<?php get_header(); ?>
<div id="cuerpo">
    <?php
    if (have_posts()) {
        while (have_posts()) {
            the_post();
    ?>
        <h1><?= the_title() ?></h1>
    <?php
            the_content();
        }
    }
    ?>
</div>
<?php get_footer(); ?>
```

[6] Ahora desde el panel de administración de WordPress tenemos que ir al apartado Temas y activar el nuevo tema que aparece con el nombre de Mi sitio, pero no muestra previsualización.

[7] Probamos la página ahora y, si todo va bien, capturamos la pantalla y, con ayuda de un programa de imágenes, guardamos la imagen en el directorio del tema con el nombre **screenshot.png**. Si volvemos al panel de temas en WordPress ahora sí deberíamos ver la imagen en miniatura del tema.

Práctica 9.6: Crear páginas estáticas en Drupal

- Haz lo mismo que en la Práctica 9.4, pero utilizando Drupal.

SOLUCIÓN: PRÁCTICA 9.6

[1] Desde el menú de administración ve a **Contenido-Agregar contenido-Página básica**.

[2] Escribe la primera página, por el título Sobre mi y el contenido que quieras.

[3] En el apartado **Opciones de menú** indica **Proporciona un enlace de menú** y marca ecomo elemento padre del enlace el **Menu principal.**

[4] En opciones de ruta URL escribe **sobre-mi** para disponer de un permalink a la página.

[5] Asegúrate de que el apartado **Opciones de comentarios** aparece cerrado.

[6] Guarda la página y repite los pasos para la página sobre mi localidad.

[7] Comprueba el resultado.

9.10 PRÁCTICAS PROPUESTAS

Práctica 9.7: Crear tema propio en Drupal

- Crea un tema propio semejante al realizado para WordPress en la Práctica 9.5.
- Al aspecto debe ser lo más parecido posible y debe conseguir los mismos hitos con los enlaces y el recorrido mediante permalinks.

Práctica 9.8: Tipo de contenido personalizado en Drupal

- Crea un nuevo tipo de contenido y llámalo **Noticia**.
- Consigue que en ese tipo de contenido solo se pueda indicar el nombre y el cuerpo.
- Permite en esas noticias añadir términos de una nueva taxonomía (llámala como desees), de modo que se autorrellenen y se puedan introducir tantos como deseemos.
- Haz que el cuerpo de la noticia solo pueda tener como HTML, las etiquetas de negrita (***strong***) y cursiva (***em***).

Práctica 9.9: Página personalizada de búsqueda en Drupal

- Esta práctica requiere haber realizado la anterior.
- Consigue crear una nueva página llamada ***buscar noticias***, en la que un cuadro de texto nos pida un término y pulsando un botón vayamos a otra página que nos muestre las noticias relacionadas con el término.
- Haz que un enlace en los menús de Drupal nos permita ir hasta la página de búsqueda.

- Integra tanto la página de búsqueda como la de resultados para que tengan apariencia coherente con el tema actual.

Práctica 9.10: Consultas a la base de datos de WordPress

- Crea una página en la que aparezca el nombre de cada categoría y el total de páginas que tiene dicha categoría
- La página se realizará directamente en PHP (sin usar realmente WordPress) y accediendo a las tablas de la base de datos en las que guarda WordPress la información.
- Haz que esta página tenga un enlace en el menú lateral (usa un tema cualquier de WordPress).

Práctica 9.11: Consultas a la base de datos de Drupal

- Crea una página en la que aparezca el nombre de cada categoría y el total de páginas que tiene dicha categoría
- La página se realizará directamente en PHP (sin usar realmente Drupal) y accediendo a las tablas de la base de datos en se guarda la información.
- Haz que esta página tenga un enlace en la página de inicio

Práctica 9.12: Página usando la base de datos de WordPress

- Usando solo la base de datos de WordPress, consigue sacar una página que muestre los títulos de los post (solo entradas, no páginas) que se están publicando ahora.
- Cada título será un enlace mediante el cual veremos el contenido de ese post.
- El contenido se muestra con un enlace que nos indicará cuántos comentarios tiene esa entrada.
- La cuenta de los comentarios es también un enlace, al hacerle clic, veremos todos los comentarios de ese contenido

Práctica 9.13: Publicar contenido en Joomla!

- Basándote en la Práctica 9.1 y la Práctica 9.2, publica dos artículos uno sobre Marte y otro sobre Júpiter categorizando términos de la misma forma que en esas prácticas y dando el mismo formato.
- Crear también dos páginas estáticas que sean enlazables desde la portada (al estilo de la Práctica 9.3 y la Práctica 9.5) una de ellas que hable sobre ti y la otra sobre tu localidad.

9.11 RESUMEN DE LA UNIDAD

- La publicación de contenidos es la actividad fundamental en un sitio administrado mediante CMS.
- La publicación implica no solo el texto e imágenes que se desean publicar, sino otros aspectos como: la apariencia general, navegación hacia contenidos, permisos de usuario y la navegación hacia esos contenidos desde los menús principales.
- En WordPress hay dos tipos de contenidos:
  - Entradas o post. Que cambian a menudo y se organizan, por defecto, de modo que aparezcan antes las entradas más recientes.
  - Páginas estáticas. Contenido del sitio que cambia muy poco a menudo.
- En Drupal existen los mismos tipos de contenido, pero se pueden crear más.
- Las taxonomías permiten asignar términos, que luego se utilizan para organizar y hacer búsquedas en el sitio.
- Las entradas y páginas pueden incorporar elementos multimedia que se organizan en una biblioteca dentro del CMS.
- La publicación de comentarios es una de las bases de los sitios administrados por CMS. Se tiene que establecer una política sobre como publicar y ver comentarios, de otro modo nos arriesgamos a tener pocos comentarios o a tener mucho spam en ellos.
- Los temas, tanto en WordPress como en Drupal, se pueden personalizar. Para ello hay que conocer la estructura y arquitectura completa de los temas en cada CMS.

9.12 TEST DE REPASO

En WordPress cuando se borra una entrada o post:

a) Se elimina definitivamente
b) Siempre debe de ser autorizado el borrado por un administrador
c) Se envía a una papelera
d) Queda marcado el post como **Borrador**

2.- **Una entrada en WordPress...**

a) Puede tener asignadas tanto categorías como etiquetas (tags)
b) Puede tener asignadas tanto categorías pero no etiquetas (tags)
c) Puede tener asignadas etiquetas (tags) pero no categorías
d) No se pueden asignar ni categorías ni etiquetas (tags) a las entradas

3.- **Un slug es...**

a) Un nombre amigable que se incrusta a la URL para acceder más fácilmente a un recurso de la web
b) Un elemento de las taxonomías diferentes de las categorías y las etiquetas
c) El nombre de una entrada o una página
d) Un plugin elaborado por terceros

Por defecto como funcionan los comentarios en WordPress

a) Cualquier persona puede publicar comentarios sin restricciones
b) Cualquier persona puede publicar comentarios, pero éstos deben de ser aprobados por un administrador
c) Cualquier persona puede publicar comentarios, pero el primero debe de ser aprobado por un administrador
d) Solo usuarios registrados en el sistema pueden hacer comentarios

En WordPress los comentarios a una entrada los puede escribir...

a) Cualquier persona que visite la página
b) Solo usuarios registrados
c) Solo editores
d) WordPress no admite comentarios

En una entrada como elementos multimedia se admite...

a) Solo imágenes
b) Solo imágenes, vídeo y audio
c) Cualquier elemento multimedia del tipo que sea
d) WordPress no admite elementos multimedia

¿Qué diferencia hay entre páginas y post en WordPress?

a) Los post son contenidos que se ordenan cronológicamente, las páginas no.
b) Los post pueden contener comentarios las páginas no
c) Los post pueden ser escritos por usuarios no administradores, las páginas solo por administradores
d) No hay diferencia entre post y página

8.- **En Drupal se admiten como tipos de contenido...**

a) Los mismos que en WordPress, pero Drupal los llama artículos y páginas básicas
b) Drupal admite solo artículos como contenido que puedan rellenar los usuarios
c) Artículos, páginas básicas, entradas de foros y cuestionarios
d) Drupal admite cualquier tipo de contenido

9.- **En Drupal los términos se agrupan en...**

a) Taxonomías
b) Vocabularios
c) Categorías
d) Tags

10.- **¿Qué diferencia hay entre bloque y región en Drupal?**

a) Los bloques contienen regiones
b) Las regiones contienen bloques
c) Son términos independientes, no tienen relación
d) Son sinónimos

UNIDAD 10

ADMINISTRACIÓN DE SITIOS GESTIONADOS POR CMS

OBJETIVOS

- Realizar copias de seguridad de un sitio gestionado por CMS
- Realizar tareas de exportación e importación de datos entre diferentes sitios gestionados por un CMS
- Realizar actualizaciones del sistema CMS
- Reconocer la importancia de las políticas de administración de usuarios
- Identificar los diferentes tipos de usuarios de un CMS
- Crear usuarios estableciendo los permisos que poseen dentro de un sitio web administrado por un CMS
- Establecer medidas que incrementen la seguridad del sitio web

CONTENIDOS

10.1 GESTIÓN DE USUARIOS EN WORDPRESS

10.1.1 INTRODUCCIÓN A LOS USUARIOS DE WORDPRESS

Durante la instalación de WordPress solo se crea un usuario: el usuario administrador del sitio. Las capacidades de ese usuario son absolutas, por lo que si siempre utilizamos a ese usuario, el riesgo de que un tercero capture la contraseña, crece.

Además, utilizar un único usuario tiene sentido en sitios web personales, pero en sitios corporativos lo habitual es que tengamos muchos usuarios. En WordPress existen cinco tipos de usuarios. Es fundamental reconocer sus capacidades antes de empezar a gestionar usuarios:

- **Administradores** (***Administrators***). Pueden realizar cualquier tarea sobre la instalación de WordPress. Por razones de seguridad debería haber una sola cuenta de usuario que funcione como administrador.
- **Editores** (***Editors***). Controlan todos los aspectos que tienen que ver con entradas (posts) y páginas. Pueden crear, eliminar y modificar cualquier post o página, además pueden realizar cualquier tarea administrativa sobre etiquetas (***tags***) y categorías, pueden subir o eliminar archivos multimedia y pueden moderar los comentarios.

  Las únicas tareas que quedan fuera de sus capacidades son las que tienen que ver con el ***back-end*** del sitio web: apariencia, menús, plugins, temas, gestión de usuarios, etc.
- **Autores** (***Authors***). Pueden publicar entradas de tipo post, pero no pueden crear páginas ni administrar las categorías o etiquetas; aunque sí pueden indicar a qué categorías y/o etiquetas pertenecen sus posts. De hecho solo pueden editar sus propios post, no pueden modificar de ninguna manera las entradas de otros usuarios.
- **Colaboradores** (***Contributors***). Pueden publicar entradas, pero no se verán en el sitio hasta que un editor o administrador las apruebe. Además, no pueden añadir elementos multimedia al post ni modificarlo, una vez lo hayan enviado para publicar. Se considera que generan entradas de tipo borrador.
- **Seguidores** (***Followers***) o **Suscriptores** (***Subscriber***). Seguidor y suscriptor significa lo mismo. El matiz es que ***seguidores*** es el nombre que tienen en los sitios implementados online en la nube de WordPress.com. En las instalaciones en un servidor propio, este mismo tipo de usuario se llama **suscriptor**. Pueden leer los post del sitio (aunque fuera privado) y escribir comentarios. En ningún caso pueden publicar sus propias entradas.

10.1.2 AÑADIR USUARIOS

Para añadir un nuevo usuario debemos ir al apartado **Usuarios** (***Users***)-**Añadir nuevo** (***Add new***) que se encuentra en el menú del panel de administración (de hecho, solo un administrador

puede realizar esta tarea). Aparece el panel de creación de usuarios, en él deberemos indicar lo siguiente:

- **Nombre de usuario**. Ese nombre será el que tendrá que indicar para acceder al panel de administración de WordPress.
- **Correo electrónico**. Es un dato obligatorio.
- **Nombre y apellidos**. Es conveniente indicarlo, pero opcional.
- **Contraseña**. Que debería ser lo más compleja posible.
- **Perfil**. En este apartado indicamos el tipo de usuario (Editor, Autor, Colaborador,etc) que estamos creando.

Además podemos hacer que se envíe por correo la contraseña del usuario. El propio usuario podrá modificar su contraseña cuando se autentifique en el sitio.

10.1.3 AUTENTIFICACIÓN DE USUARIOS

Para que un usuario pueda realizar las tareas que tiene dentro de su perfil, se tiene que autentificar. La forma más fácil es ir añadir **/login** a la raíz del sitio WordPress. Por ejemplo si la ruta a nuestro sitio WordPress es http://esteesmisitio.com/wordpress entonces para autentificarse debemos ir a http://esteesmisitio.com/wordpress/login. Podemos acceder al panel de administración (usando el ejemplo, la URL sería http://esteesmisitio.com/wordpress/wp-admin), solo que ahora el panel tendrá más o menos capacidades dependiendo del usuario que seamos.

10.1.4 PERFIL DEL USUARIO

Todos los usuarios tienen acceso al enlace **Perfil** (***Profile***) desde el que pueden cambiar aspectos de su cuenta como pueden ser:

- Esquema de color del panel de administración.
- Activar barras de herramientas.
- La información personal: email, nombre, apellidos, alias, correo electrónico o información biográfica.
- Modificar la contraseña. Sin duda el aspecto más importante de gestión de un usuario que no sea administrador.

10.1.5 APROBACIÓN DE POST

Los usuarios colaboradores, como ya se ha comentado, pueden publicar post, pero estos quedan a la espera de la aprobación de un editor o un administrador.

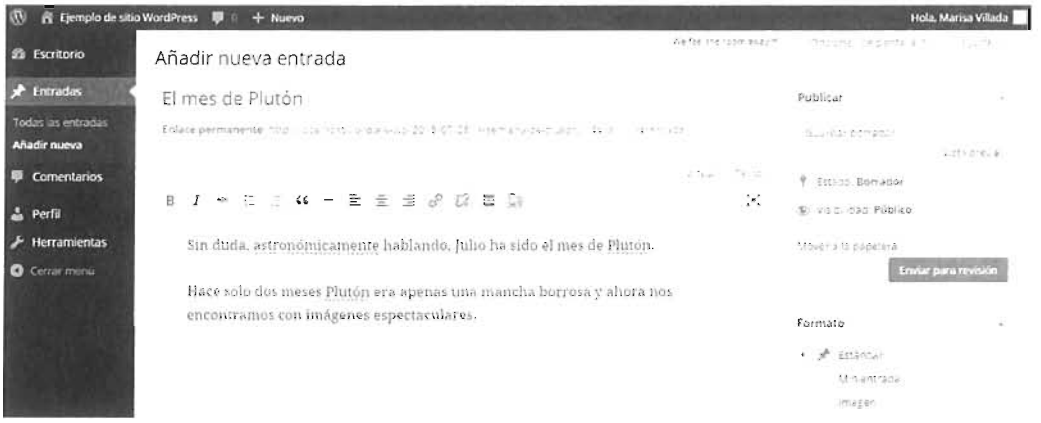


Figura 10.1: Nuevo post realizado por una usuaria de tipo *Colaborador*.

Como se observa en la Figura 10.1, la funcionalidad del panel de un usuario colaborador está disminuida y el botón de Publicar se sustituye por **Enviar para revisión**.

Los editores y administradores cuando tienen entradas pendientes, aparecen así marcadas en la lista de entradas. En el menú de entradas hay un apartado dedicado a las entradas pendientes. Para aprobar estas entradas hay que hacer clic en el título y en el panel de edición, que aparece a continuación, hacer clic en **Publicar**.

10.1.6 BLOQUEO DE POST

Figura 10.2: Lista de entradas. Se observa un post pendiente y otro bloqueado.

Puesto que podemos tener diferentes usuarios con capacidad de editar posts, es posible que ocurran conflictos. Si un usuario editor está modificando un artículo y otro lo hace también, la cuestión es ¿Cuál de las modificaciones actúa? En WordPress **solo un usuario tiene permitido en cada momento editar el artículo**.

Así si un usuario está editando una entrada, el resto de editores conectados verán un icono en forma de candado sobre esa entrada, además, se indica quién está realizando la notificación

(véase Figura 10.2). Cuando ese usuario deja de editar, porque abandona la página o cierra el navegador, el resto verán que el post está liberado. Hay retraso en estas notificaciones porque WordPress hace la comprobación de bloqueo cada 15 segundos.

Lo que sí puede hacer otro editor es tomar posesión de la entrada para editar el artículo. Pero lo que ocurre entonces, es que las modificaciones del otro editor quedan anuladas. Esto puede iniciar una guerra sin fin, solo cabe apelar a la responsabilidad de los editores. Desgraciadamente WordPress no tiene mecanismos para dar poder a uno de los editores en caso de tener que decidir qué versión final queda. Lo que sí puede hacer el administrador es rebajar el nivel de los usuarios (de editores a autores por ejemplo).

10.1.7 REVISIONES

10.1.7.1 MANEJO DE REVISIONES

Se trata de una característica muy potente. WordPress permite que un mismo post pueda ser editado por varios usuarios, siempre y cuando tengan privilegios para ello. No hay que olvidar las cuestiones de bloqueo comentadas en el apartado anterior: solo un usuario puede editar una entrada al mismo tiempo.

Una de las grandes habilidades de WordPress es el hecho de que mantiene todas las revisiones que se han hecho sobre una entrada.

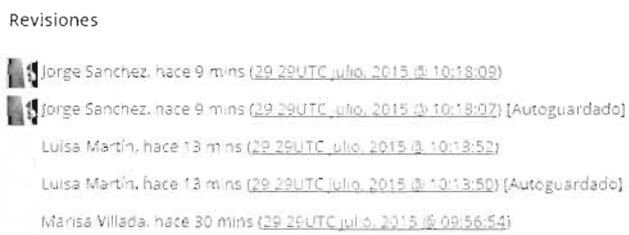


Figura 10.3: Panel de revisiones mostrando los cambios realizados sobre un post

Podemos verlas desde el panel de revisiones al modificar una entrada. Si el panel no está a la vista, que es lo habitual, debemos ir a **Opciones de pantalla** y hacer clic sobre dicho panel. El panel nos muestra cuándo se han realizado cambios y qué usuario los ha hecho.

Figura 10.4: Panel Publicar disponible al editar una entrada. Se ha resaltado el enlace para examinar las revisiones de la entrada+

Además, en el panel **Publicar** podemos ver exactamente qué cambios se han realizado. Para ello en este panel hay que hacer clic en el enlace **Explora** (*Browse*) disponible en el apartado Revisiones.

Tras hacer clic en ese enlace, un nuevo panel de **Comparación de revisiones** nos muestra con detalle los cambios que se han realizado en cada revisión. Podemos avanzar de una a otra con los botones **Anterior** o **Siguiente**.

Figura 10.5: Panel de comparación de revisiones mostrando una revisión sobre una entrada

El panel compara revisiones consecutivas. Pero hay una casilla con el texto **Comparar dos revisiones cualesquiera** (***Compare any two revisions***).

Si no estamos conformes con los cambios ocurridos desde una determinada revisión, el botón **Restaurar esta revisión** (***Restore this revision***) vuelve a dejar la entrada exactamente en el estado que tenía cuando se realizó esa revisión.

10.1.7.2 EL PROBLEMA DE LAS REVISIONES

Las revisiones son una herramienta fantástica, pero tienen un problema: el espacio. Almacenar los cambios que produce cada revisión tiene el problema de que cada entrada ocupa mucho espacio en disco (cuando hay bastantes revisiones por entrada).

Podemos limitar el espacio que gastan las revisiones modificando el archivo **wp-config.php** de configuración de WordPress:

```
define('WP_POST_REVISION', 5);
```

La constante **WP_POST_REVISION** permite indicar el máximo número de revisiones que permitimos. En el código anterior solo almacenaríamos 5 revisiones por cada entrada. Más agresivo es este código:

```
define('WP_POST_REVISION', false);
```

No se almacenarán los cambios anteriores al último: no hay revisiones. No es muy conveniente esta última posibilidad, especialmente si hay varios editores.

10.2 GESTIÓN DE LOS DATOS DE WORDPRESS

10.2.1 COPIAS DE SEGURIDAD EN WORDPRESS

Indiscutiblemente una de las labores fundamentales del administrador de un sitio con gran cantidad de contenido es la gestión de las copias de seguridad de los datos. Perder los contenidos de un sitio que mueve gran cantidad de datos es, simplemente, catastrófico. De ahí que tengamos que tener una política seria y meticulosa a este respecto.

Tener una buena administración en materia de copias de seguridad supone que en caso de desastre podamos reinstalar absolutamente todo el contenido en el menor tiempo posible. Realmente las cuestiones sobre las copias de seguridad, en el caso un sitio WordPress, deben incluir:

- Toda la información en la base de datos. Teniendo en cuenta que se trata de una base de datos MySQL y que esta base de datos dispone de muchas opciones al respecto.
- Temas, plugins, archivos subidos, código modificado manualmente, etc. Todo ello se encuentra en el directorio **wp-content** en la raíz de WordPress.

No es difícil realizar una copia de seguridad, de hecho con hacer copia de la base de datos MySQL y de todo el sitio WordPress, ya tenemos una copia en condiciones de ser restablecida sin ningún problema. Eso sí, debemos hacer copia a menudo para perder la mínima información posible. Teniendo en cuenta que los posts y comentarios pueden cambiar constantemente, lo ideal es tener una solución automatizada al respecto.

Las dos estrategias fundamentales de copia son:

- Hacer la copia manualmente cada cierto tiempo. Eso sí, el destino de la copia debe ser una computadora propia. Aún mejor, si la copia la realizamos hacia un dispositivo de almacenamiento externo que, posteriormente, guardemos en sitio seguro.
- Hacer copia diaria hacia un servidor diferente del nuestro a través de un servicio automatizado.

10.2.1.1 COPIAS DE SEGURIDAD AUTOMATIZADAS

Sin duda son la mejor opción. Delega en un servicio externo esta gestión y nos quita esta preocupación de nuestra labor como administrativos. La pega: el precio, la mayoría de los servicios, y sin duda los mejores, son de pago.

El más famoso es **Vaultpress** (https://vaultpress.com/plans/) que por 100$ al año realiza copia diaria y mantiene un archivo de todas las versiones de nuestro sitio en los últimos 30 días. También se utilizan mucho soluciones de backup de sitios web completos, como **CodeGuard** que serviría no solo para copias de seguridad de WordPress sino de cualquier otro tipo incluidos sitios gestionados por otros motores CMS como Joomla o Drupal.

Numerosas copias de seguridad automatizadas se realizan mediante la instalación de un plugin, lo que añadirá un nuevo elemento al panel de administración que nos permitirá la administración del sitio. Algunas son de pago como **Backup Buddy**, y otras gratuitas como **UpdraftPlus Backup and Restoration** que permite hacer copia en la nube de Amazon, Google, Microsoft o DropBox entre otras.

10.2.2 EXPORTAR LOS DATOS DE WORDPRESS

La exportación de datos en WordPress es muy sencilla. Basta ir al apartado **Herramientas** (*Tools*)-**Exportar** (*Export*). En esa sección del panel de control de administración podremos elegir si deseamos descargar todos los datos o solo los post o las páginas.

Finalmente, tras hacer clic en el botón **Descargar el archivo de exportación** (*Download Export File*) se descarga un archivo XML con los datos.

En principio ese archivo se puede utilizar para importar datos a otros CMS. Pero hay pocos sistemas que lo permitan directamente. En muchos casos tendremos que utilizar plugins o utilidades de terceros.

10.2.3 IMPORTAR DATOS EN WORDPRESS

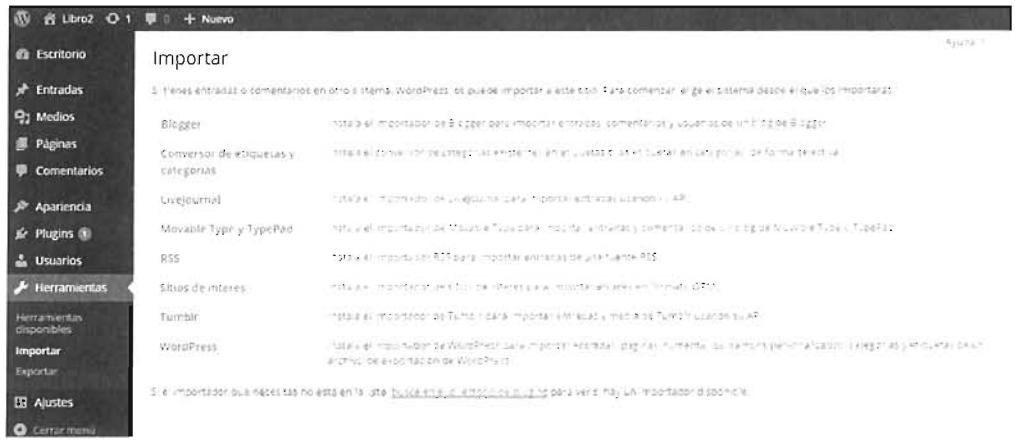


Figura 10.1: Panel de importación

El proceso es un poco más complicado, ya que los datos pueden proceder de otra instalación de WordPress o de otro CMS. Los pasos son:

[1] Ir a **Herramientas** (*Tools*)-**Importar** (*Import*) menú desde el cual se instala un plugin encargado de realizar la tarea. Este plugin es distinto dependiendo de donde procedan los datos. Podemos importar datos de otra instalación de WordPress, de una fuente **RSS**, de **Blogger**, de **Tumblr**, etc.

[2] Elegimos el sistema desde el que hemos exportado los datos. Podremos instalar el plugin correspondiente si no lo habíamos instalado. Y tras la instalación podremos elegir el archivo con los datos exportados. Si el plugin ya está instalado, podremos directamente elegir el archivo de exportación.

[3] Tras elegir el archivo podremos hacer clic en **Subir archivo e importar**.

[4] Después se nos harán preguntas sobre los datos elegidos. Lo malo es que esas preguntas no están traducidas. Lo principal que nos pregunta es si se deben crear en nuestra base de datos, usuarios que tengan el mismo nombre de usuario que en la otra instalación.

El problema viene de que los usuarios de una y otra instalación no coincidirán, por lo que tenemos que decidir si los creamos, si asignamos los post de un autor a otro autor ya existente o si creamos usuarios nuevos con el nombre que queramos a los que asignaremos los post de los usuarios originales.

También se nos preguntará si se descargan los archivos adjuntados a los post (como las imágenes por ejemplo). Si dichos archivos no están disponibles, la importación fallará. Si no adjuntamos los archivos, los post se quedarán sin sus imágenes, vídeos, etc.

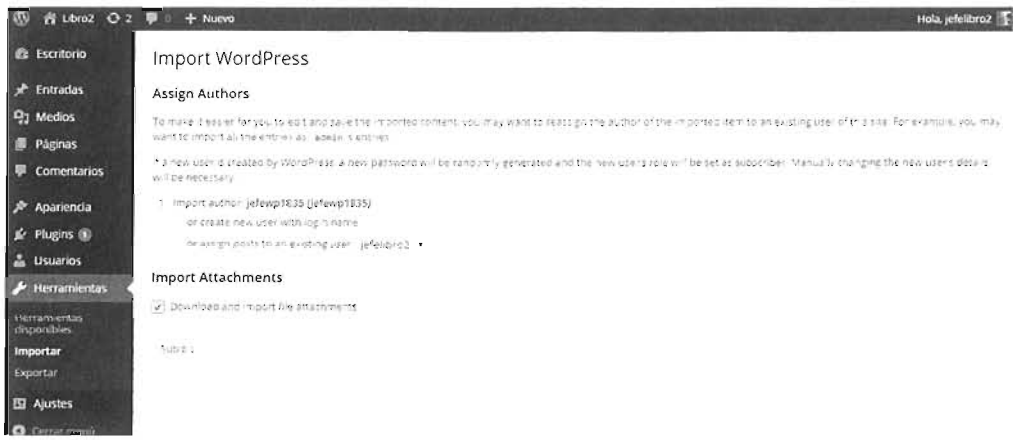


Figura 10.2: Panel de importación de datos procedentes de otra instalación de WordPress. En este caso los post originales eran de un autor y se les vamos a asignar a uno de los autores de la nueva instalación.

[5] Tras responder adecuadamente a las preguntas, hacemos clic en **Submit**. Conviene (como nos recordará el propio WordPress restablecer las contraseñas de los usuarios que se han importado.

10.3 ACTUALIZACIÓN DEL SISTEMA WORDPRESS

10.3.1 INTRODUCCIÓN A LAS ACTUALIZACIONES

El proceso de actualización siempre ha tenido cierta complejidad, pero sobre todo riesgo. Lo primero que hay que tener en cuenta es si realmente necesitamos la nueva versión. En general las nuevas versiones aportan ciertas mejoras y hay que estar especialmente atentos a las que ofrecen para la seguridad del sistema. Pero también hay versiones en muchos CMS que tienen algunos problemas recién se han creado; por ello conviene esperar un poco y examinar si esa versión está funcionando bien.

En todo caso, en WordPress la actualización es sencilla. En el propio escritorio de administración se nos avisa de las nuevas versiones de WordPress y desde ahí mismo podremos realizar la instalación.

10.3.2 ACTUALIZAR EL SISTEMA

Antes de instalar una nueva versión de WordPress conviene, encarecidamente, realizar estas tareas:

- Copia de seguridad de los datos.
- Copiar de seguridad de los archivos WordPress.

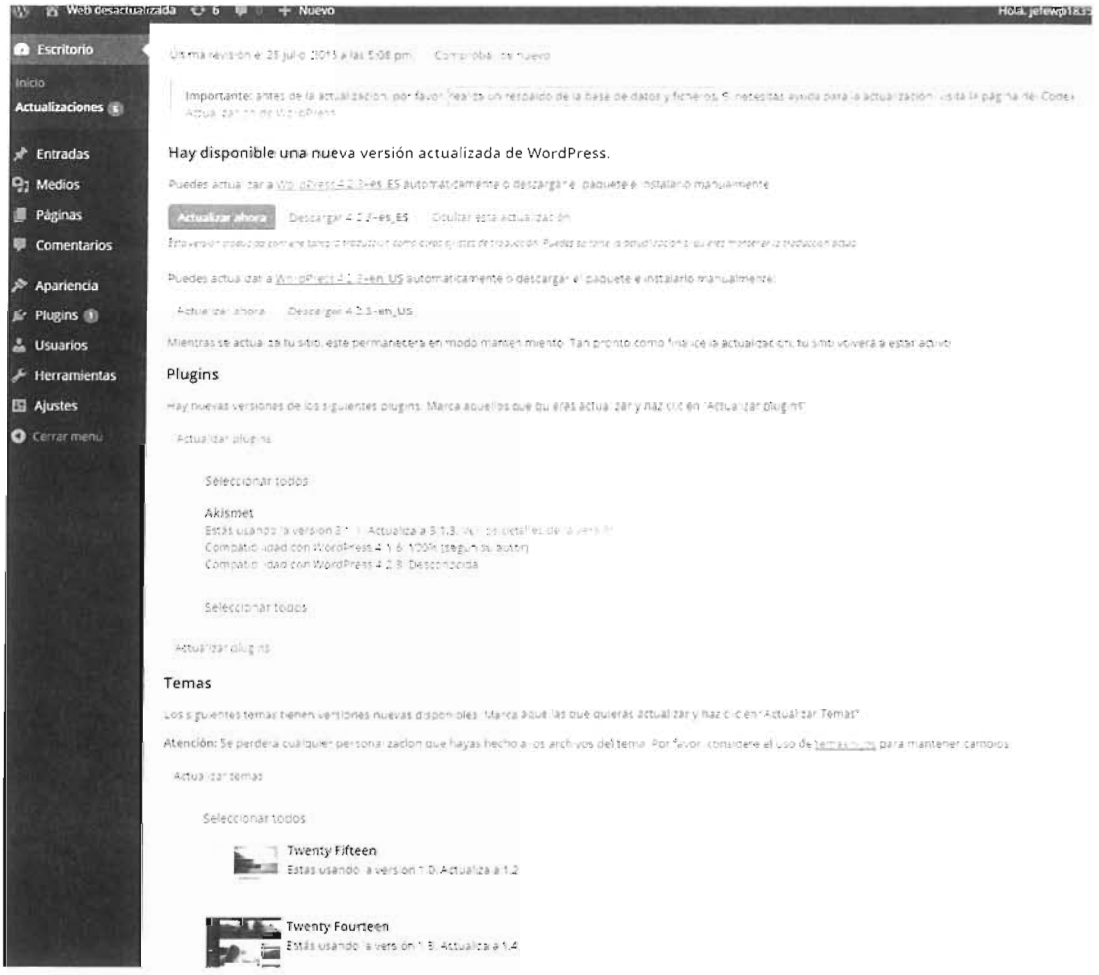


Figura 10.3: Panel de actualizaciones de una versión de WordPress que no está al día

Tras estas tareas lógicas, la actualización de WordPress se puede realizar en el apartado **Actualizaciones** (***Updates***) que se encuentra debajo del enlace al **Escritorio** (***Dashboard***) en el panel de administración.

En ese enlace tenemos la posibilidad de actualizar tanto el sistema WordPress como todos los elementos del sistema sobre los que se haya detectado que no están actualizados. Actualizar el sistema es hacer clic en el botón **Actualizar ahora** (***Update now***). Después se descarga de Internet la nueva versión y se instala. Un panel de información nos avisará de los cambios en la versión; los fundamentales son los relativos a errores.

Tras actualizar WordPress tenemos que realizar estas tareas:

- Revisar los permalinks, comprobando que funcionan como antes. Implica también comprobar el archivo **.htaccess**. Los ajustes de permalinks (**Ajustes-Enlaces permanentes**) estarán como al principio.
- Comprobar los plugins. Puede que necesitemos instalar de nuevo algunos de ellos o bien actualizarlos.
- Comprobar los temas, de la misma forma que los plugins.

10.3.3 ACTUALIZACIÓN DE OTROS ELEMENTOS

Como se puede observar en la Figura 10.3, el mismo panel de actualizaciones comentado en el apartado anterior nos permite conocer qué componentes (plugins, temas, traducciones, etc) están desactualizados. Desde ese mismo panel podremos actualizarlos.

Puede ocurrir también que algún componente no funcione con la versión nueva de WordPress, y si tenemos dependencia de él, encontrarnos con un verdadero problema. El consejo es intentar buscar opciones al componente; si es un tema, simplemente no utilizarle. La razón es que, siempre que sea posible, conviene tener al día el sistema WordPress.

Las mejoras operativas que aportan las nuevas versiones de WordPress y que suelen traer nuevas funciones y usabilidad al sistema, pueden no ser razón suficiente para querer actualizar una aplicación web que está funcionando correctamente. Pero también están las mejoras de seguridad que son las que eliminan errores en las versiones antiguas y que continuamente se renuevan en las nuevas versiones. Tener una versión muy desfasada, puede traer como consecuencia que tengamos un agujero de seguridad en nuestro sistema.

Resumiendo: un simple componente, o unos pocos, no pueden determinar la versión de WordPress a instalar en nuestro sistema. Es labor del administrador determinar qué versión es la idónea tras un análisis de ventas y desventajas al respecto y qué componentes nos conviene utilizar con ella.

10.4 SEGURIDAD EN WORDPRESS

10.4.1 ¿POR QUÉ HAY QUE PRESTAR ATENCIÓN A LA SEGURIDAD?

En el mundo de hoy la frase anterior se responde rápidamente a sí misma. Internet posee ya miles de millones de usuarios, con millones y millones de webs. Hay que recordar, además, que

la cuarta parte están creadas con WordPress, lo que le convierte indudablemente en la tecnología más popular para desarrollar sitios y aplicaciones web.

Esto tiene una clara contrapartida. Las instalaciones con WordPress son uno de los objetivos claros para las personas que se dedican al ataque de servidores en Internet. La facilidad de uso de WordPress para rápidamente crear un sitio web de una apariencia bastante profesional, hace que todo tipo de personas actúen como administradores de sitios web WordPress; pero, sin saberlo, dejan puertas abiertas al acceso indebido de terceros.

Es irresponsable no dedicar tiempo a la seguridad. Cuanto más contenido y usuarios participen en él, con más razón. El ataque a un sitio web, deja una muy mala imagen para el propietario del mismo; en muchos casos irrecuperable.

Desgraciadamente la garantía de tener un sitio realmente blindado ante cualquier ataque no existe, pero hay que, al menos, hacer todo lo que esté de nuestra parte para minimizar los riesgos.

10.4.2 PRINCIPALES MEDIDAS DE SEGURIDAD

- **Trabajar con la última versión** tanto de WordPress como de sus temas y plugins (véase 10.3 *"Actualización del Sistema WordPress"*, en la página 423)
- **No utilizar admin como nombre para el usuario MySQL.** WordPress utiliza un usuario que tiene privilegios suficientes para escribir en la base de datos en la que se guarda la información de WordPress (véase 8.2.2 *"Instalación manual de WordPress"*, en la página 323). No conviene utilizar ***admin*** como nombre de ese usuario. El ataque típico hacia la base de datos de WordPress es siempre intentar obtener la contraseña de ese usuario. Si el usuario no tiene un nombre predecible, es mucho menos probable el éxito del ataque.
- **Cambiar el prefijo de las tablas en MySQL.** Durante la instalación, WordPress nos permite indicar un prefijo para las tablas que se crean. Lo clásico es dejar el prefijo **wp_**, pero eso hace que el nombre de las tablas sea perfectamente reconocible. Podemos indicar otro de nuestra cosecha para hacerle menos previsible y, por lo tanto, menos atacable.
- **Utilizar contraseñas muy complejas.** El ataque por fuerza bruta, consiste en intentar acceder a un sistema probando con todas las contraseñas posibles. Cuanto más complejas sean las contraseñas, más tardan esos programas en obtener éxito. Una buena contraseña debe ser larga, incorporar números, letras mayúsculas, minúsculas y símbolos.
- **Limitar el número de intentos al conectar.** Es la medida que mejor funciona ante ataques de fuerza bruta para dar con la contraseña de un usuario. Cuando un usuario se conecta a WordPress, en principio, tiene un número infinito de intentos. Podemos hacer que, tras el número de intentos que indiquemos, se bloquee la IP del usuario durante un tiempo. Tras ese tiempo lo podría volver a intentar (no debemos bloquearla del todo, porque puede ser un usuario válido que se ha equivocado).
- **Ocultar la versión de WordPress.** Si estamos utilizando una versión que tenga un agujero de seguridad, los atacantes que conozcan el problema sabrán como sobrepasar la seguridad del sitio. Hay atacantes que buscan por Internet versiones concretas de WordPress que tienen agujeros de seguridad.

La versión de WordPress se conoce porque hay una etiqueta meta que lo indica, concretamente (ejemplo para una instalación de WordPress versión 4.1.2):

```
<meta name="generator" content="WordPress 4.1.2" />
```

Una simple búsqueda en Internet de este código con la versión que se quiere atacar nos mostraría miles de entradas, es decir, miles de sitios con esa versión concreta.

Evitar la escritura de esa etiqueta es fácil, simplemente hay que escribir una línea al final del código del archivo **functions.php** del tema que se esté utilizando. Por ejemplo, si utilizamos el tema ***twenty fifteen***, la ruta, desde la raíz de WordPress sería: **wp-content/themes/twentyfifteen/functions.php**

La línea a escribir es:

```
Remove_action('wp_head','wp_generator');
```

- **Limitar el permiso de los archivos.** Es un tema lógico que conviene recordar, se trata de que para los usuarios normales los archivos de WordPress tengan permisos de solo lectura y dejar la escritura solo para el administrador del sistema.
- **Proteger el archivo de configuración.** Un problema típico que puede acarrear un resultado desastroso es: que no funcione PHP durante un tiempo. Si es así, el servidor web mostrará el código fuente literal de los archivos PHP que se soliciten, en lugar de traducirle.

  Un atacante que se dé cuenta de este detalle, puede requerir la entrega del archivo **wp-config-php** y ver el código de este crítico archivo. Normalmente, si requerimos la entrega de este archivo, no se ve nada en el código fuente, pero si el intérprete de PHP no está funcionando, veremos todo el código y, con él, el usuario de la base de datos MySQL y su contraseña.

  La solución más sencilla es mover ese archivo fuera de la raíz de documentos del servidor web, al directorio padre. WordPress puede buscar el contenido de ese archivo cuando está en el directorio padre de la raíz y puesto que está fuera de la raíz, el servidor web no puede entregarlo de ninguna manera.

  En el caso de que no podamos optar por esta solución (por ejemplo, si alojamos nuestra web en un servidor del que no tenemos acceso), podemos editar el archivo .htaccess de la raíz de WordPress y escribir este código, el cual hace que Apache prohíba el acceso al archivo de configuración:

```
<Files wp-config.php>
 Order Deny,Allow
 Deny from all
</Files>
```

- **Usar códigos aleatorios para las claves.** Como se ha explicado en el Apartado 8.2.2.3 *"Configuración de la instalación de WordPress. Archivo wp-config.php"*, en la página 326.

- **Utilizar SSL**. Nuevamente es un consejo lógico para cualquier web, no solo de tipo WordPress, que recoja datos de usuario. Para ello, nuestro servidor debe utilizar una firma digital y eso implica configurar nuestro servidor web. Si esto es posible, entonces podemos indicar a WordPress que las conexiones de usuario conecten por SSL mediante esta línea en el archivo **wp-config.php**:

```
define('FORCE_SSL_LOGIN', true);
```

Si deseamos que todo WordPress utilice SSL, entonces además añadiremos esta línea:

```
define('FORCE_SSL_ADMIN', true);
```

10.5 ADMINISTRACIÓN DE USUARIOS EN DRUPAL

10.5.1 TIPOS DE USUARIOS EN DRUPAL. ROLES

Fundamentalmente Drupal divide a los usuarios en **autentificados** y **anónimos**. Los anónimos son los visitantes, los que no crean contenido. Visto así, parece que la gestión de usuarios de Drupal es muy simple.

Sin embargo, Drupal incorpora roles a fin de indicar a cada usuario lo que puede realizar. Y eso ya especifica mucho más a cada usuario. Podemos diferenciar entre editores de un departamento concreto, revisores de contenido, administradores de segundo nivel... Francamente tendremos una jerarquía a medida. Los roles se pueden incluso asociar a secciones específicas del sitio. Además, podemos asignar más de un rol a cada usuario.

Los roles, en realidad, aglutinan **permisos**. Y los permisos es lo que concreta las acciones que un usuario puede realizar. Los permisos establecen cosas muy concretas, por ejemplo, si un usuario puede añadir un artículo, si puede añadir comentarios, si puede ver la información personal de otros usuarios, etc.

10.5.2 AÑADIR CUENTAS DE USUARIO

Hay una cuenta obligada en Drupal, la del súper administrador. Las demás son todas opcionales. Puede haber sitios que solo necesiten esa cuenta. Pero no es lo normal.

- **Solo los administradores**. Con lo cual la administración de cuentas está centralizada. Es el escenario más habitual, menos flexible pero más controlado para evitar altas de usuarios no deseadas.
- **Los visitantes**. Cualquier visitante se podría dar de alta como usuario directamente. Esto dispararía el número de usuarios y podría dar cabida a usuarios no deseados. Salvo en páginas en las que se requiera un gran dinamismo en la creación de contenidos que impidan poder verificar a cada usuario (páginas wiki, por ejemplo) o en entornos controlados de red, no se suele utilizar.

Cuando un visitante se puede dar de alta, siempre conviene pedir verificación por correo electrónico, lo cual implica que el visitante se da de alta, indica su correo y después recibe en él un correo con un enlace que debe utilizar para verificar la cuenta. Entonces ya podrá usar su cuenta.

- **Los visitantes, pero con aprobación de los administradores.** Se pueden dar de alta los visitantes, pero para que la cuenta se active, debe de ser aprobada. Cuando un visitante se da de alta en esta forma, aparece bloqueado en la lista de usuarios

  Esta forma, funciona igual que la anterior, salvo en que hay un paso previo. Cuando el visitante se da de alta indicando su nombre y email, recibe un email indicando que la cuenta de usuarios se debe aprobar. El usuario aparecerá como usuario bloqueado en la lista de usuarios del panel administrativo. Cuando un administrador le desbloquea, recibe otro email indicando su desbloqueo junto con el enlace que permite autentificar el email.

Para los visitantes que se dan de alta ellos mismos, es imperativo que al menos tengan que verificar su correo electrónico. De otro modo, estamos abriendo la puerta a la creación de cuentas spam y de otro tipo, aparte de que no tendríamos ningún medio verificado para comunicar con los usuarios.

El envío automatizado de los correos a los usuarios implica que el módulo de PHP para el envío de correos (PHP mail) funcione correctamente en el servidor. De no ser así, debemos instalar módulos, el más habitual es el módulo SMTP, a través de los cuales podemos indicar los datos para el envío de una cuenta de la que dispongamos y que asociaremos a Drupal para el envío de correos.

Esta decisión sobre la gestión de los usuarios se realiza desde **Configuración-Configuración de las cuentas** en el menú de administración. En el panel de configuración se pueden realizar todas estas tareas sobre las cuentas:

- Nombre que se da a los usuarios anónimos.
- Decidir como se dan de alta los usuarios, según lo explicado en los párrafos anteriores.
- Decidir qué ocurre con el contenido de los usuarios cuando se dan de baja.
- Automatizar los permisos de administrador cuando se añadan nuevos módulos.
- Indicar qué información personal se pide a los usuarios, el nombre de su directorio de imágenes, como se presentan, qué imagen predeterminada asignamos a los usuarios por defecto.
- Texto del mensaje de bienvenida para los nuevos usuarios. Podemos personalizar el mensaje para cada uno de los tipos de alta.
- Texto del mensaje de activación, bloqueo, cancelación, confirmación de cancelación y recuperación de la cuenta.
- Texto del mensaje de bloqueo de la cuenta.

- Datos que se piden al usuario y controles que se usan para ello.

10.5.3 EDITAR ROLES Y PERMISOS

La gestión de los roles, desde el menú de administración, se encuentra eligiendo **Usuarios** y después en la pestaña **Permisos**, el botón **Roles**. Inicialmente hay tres roles: **Administrador**, **usuario anónimo** y **usuario autenticado** (los dos últimos bloqueados).

Para añadir un rol, basta con, en la parte inferior del panel, escribir el nombre del rol y hacer clic en **Añadir rol**.

Cada rol, dispone del enlace **editar permisos**, que es el encargado de indicar qué permisos asignamos a cada rol.

También podemos asignar a los usuarios permisos a través de botón **Permisos** en ese panel. El botón muestra una tabla con todos los roles y los permisos posibles a asignar, una casilla de verificación permite asignar o no el permiso al rol.

10.5.4 CREAR Y EDITAR USUARIOS

La creación de usuarios se hace a través del enlace **Usuarios** del menú de administración. Ese panel muestra la lista de usuarios (la cual podemos editar). Podemos añadir un usuario a través del botón **Añadir usuario**, el cual muestra un panel en el que podremos:

- Indicar el nombre de usuario
- Indicar su correo electrónico
- Indicar su contraseña
- Hacer que la cuenta esté bloqueada inicialmente
- Asignar rol al usuario
- Indicar si notificamos al usuario sobre su nueva cuenta
- Elegir el idioma del usuario (en el que le aparecerán los menús y paneles de Drupal).

Ese mismo panel de usuarios podemos editar a un usuario, eligiendo editar usuario. Al editar podremos:

- Cambiar el nombre del usuario
- Cambiar el email del usuario
- Cambiar la contraseña (en una pestaña superior)
- Bloquear a desbloquear la cuenta
- Modificar el rol
- Modificar el idioma de la interfaz

- Cambiar o asignar la imagen de la cuenta

ACTIVIDAD 10.1: PROBAR EL FUNCIONAMIENTO DE LOS USUARIOS

- Para realmente asimilar el funcionamiento de los usuarios de WordPress y sus posibilidades, lo mejor es practicar. Realiza la Práctica 10.1, en la que puedes probar la mayoría de acciones con los usuarios

10.6 GESTIÓN DE LOS DATOS EN DRUPAL

10.6.1 COPIAS DE SEGURIDAD Y EXPORTACIÓN DE DATOS

Todo lo comentado sobre copias de seguridad para WordPress (Apartado 10.2.1 *"Copias de seguridad en WordPress"*, en la página 421), valdría para Drupal ya que tiene las mismas particularidades.

Drupal dispone de un módulo, que hay que instalar, llamado **Backup & Migrate** (disponible en https://www.drupal.org/project/backup_migrate) que facilita las tareas de copia de seguridad y exportación/importación de datos. Es un módulo muy poderoso que admite hacer copia de la base de datos MySQL (incluso de más de una), copia de los archivos de Drupal, hacer copia hacia correo electrónico o cuenta FTP o cuenta **NodeSquirrel**, programar automáticamente el proceso de copias para que se realicen automáticamente y encriptar (mediante cifrado AES) la copia de seguridad. En definitiva, es un módulo absolutamente profesional.

Una de las opciones más interesantes es el uso de **NodeSquirrel** que es un servicio de copias de seguridad en la nube (al estilo de **CodeGuard** o **Vaultpress**) creado por la empresa **Pantheon** (creadores también del módulo Backup & Migrate) y que ofrece almacenar los datos de un sitio de hasta 5GB de manera gratuita. A partir de ahí tiene diversos precios.

Una vez instalado el módulo, debemos ir a **Configuración**-apartado **Sistema-Backup and Migrate** para administrar las copias.

10.6.1.1 REALIZACIÓN DE COPIAS DE SEGURIDAD

Directorio de archivos privados

Si realizamos copias de seguridad en modo local, es decir, hacia el propio servidor donde está la base de datos. Entonces, debemos primero indicar la ruta a nuestro sistema privado de archivos. Esto se hace desde **Configuración-Multimedia-Sistema** de archivos y ahí se indicaría la ruta (debe estar fuera de las rutas de Drupal). Esa ruta es necesaria para poder utilizar **Backup and Migrate**.

Figura 10.4: Panel de sistema de archivos estableciendo la ruta al sistema privado de archivos. Será la que se use al almacenar las copias de seguridad

Copia rápida

Figura 10.5: Panel ***Backup and Migrate*** mostrando las opciones rápidas para hacer copias de seguridad

Una vez configurado el directorio de archivos privados, en el apartado **Configuración-Sistema-Backup and Migrate** disponemos de la posibilidad de hacer una copia rápida (***Quick Backup***) en la cual se establecen estas opciones:

- Indicar de qué hacemos copia:
  - **Default Database**. Base de datos MySQL en uso por Drupal.
  - **Public Files Directory**. Directorio de archivos público que contiene los archivos creados por los usuarios.
  - **Entire Site**. Todo lo anterior.
- Indicar el destino de la copia:
  - **Local-Manual Backups Directory**. Se almacena la copia dentro del directorio de archivos privados en la ruta que se indique en la pestaña Ajustes de **Backup and Migrate**. Por defecto dentro del directorio privado se almacena en **backup_migrate/manual**.
  - **Local-Download**. El archivo de copia de seguridad se descargará desde el navegador.
  - **NodeSquirrel**. La copia se realiza en nuestra cuenta de NodeSquirrel.
- Añadir un comentario (***Add a note to the backup***). Lo cual es bastante conveniente.

Tras pulsar el botón **Backup now**, la copia estará hecha.

Copia avanzada

El botón, en el mismo panel, **Advanced Backup**, nos permite ajustar más opciones al hacer la copia. Además de las opciones vistas en la copia rápida podremos:

- Indicar el nombre de los archivos y el formato del archivo comprimido.
- Establecer opciones de compresión AES (requiere instalar el módulo de compresión AES).
- Hacer copia solo de algunas tablas de la base de datos.
- Excluir algunos tipos de archivos y/o directorios al hacer la copia.
- Enviar un email cuando se haga la copia o si falla la misma, dejar el sitio en modo offline mientras se haga la copia.

10.6.1.2 PROGRAMAR COPIAS DE SEGURIDAD

Si deseamos que las copias se realicen automáticamente de forma programada, debemos ir al apartado **Schedule** del panel **Backup and Migrate**. En ese panel podremos indicar:

- De qué hacemos la copia (***Backup Source***), la base de datos, archivos públicos o de todo.
- Qué perfil (***profile***) utilizamos al hacer las copias. Los posibles perfiles se gestionan desde el apartado ajustes.
- Cada cuánto se realiza la copia. Por defecto, aparece cada día (***Backup Every 1 Days***). Además, podemos indicar que se haga con el administrador ***cron*** de tareas de Drupal o con otras opciones (normalmente se elige el cron de Drupal).
- Podemos indicar que al hacer la copia se eliminen las anteriores (casilla ***Automatically delete old backups***). Lo cual es un riesgo porque si dejamos una sola copia, hay más posibilidades de que pueda fallar.
- Destino de la copia, normalmente el directorio de copias programadas (***Scheduled Backups Directory***).
- Podemos guardar la copia en un segundo destino a la vez (casilla ***Save a copy to a second destination***)

10.6.1.3 RESTAURAR COPIAS DE SEGURIDAD

En el mismo panel, yendo al apartado **Restore** podremos restaurar los datos procedentes de una copia de seguridad. Hay que tener en cuenta que esto elimina datos actuales, por lo que solo se debe realizar en caso de pérdida de datos.

Podemos restaurar desde un archivo que indiquemos (***Restore from an uploaded file***), o bien desde una copia realizada en la zona privada del servidor (***Restore from a saved copy***).

10.6.1.4 AJUSTES AVANZADOS

El apartado **Ajustes** (***Settings***) del panel permite establecer más opciones sobre las copias de seguridad. Concretamente:

- Desde el panel vemos las copias programadas o podemos crear nuevas programaciones (***Create new schedule***)
- Podemos indicar más fuentes para hacer copias de seguridad mediante el enlace ***Create new source***. Al hacer clic podemos indicar otros directorios y otras bases de datos (aparte de la que usa Drupal). En ambos casos deberemos indicar los datos necesarios. Para hacer copia de bases de datos MySQL, los habituales de nombre de usuario con permisos de acceso a la base de datos, contraseña, nombre de la base de datos y nombre o IP del servidor de la base de datos.
- Podemos modificar (***override***) o crear nuevos destinos (***Create a new destination***). Es en este apartado donde podremos indicar como destinos de posibles copias de seguridad:
  - Directorio FTP.
  - Cuenta **NodeSquirrel**.
  - Cuenta de almacenamiento **Amazon S3**.
  - Correo electrónico.
  - Un directorio en el servidor que aloja a Drupal.
- Modificar o añadir perfiles.

10.7 OTRAS TAREAS ADMINISTRATIVAS EN DRUPAL

10.7.1 CRON

Todos los administradores de Linux conocen la utilidad ***cron***. Se trata de un automatizador de tareas que funciona en cualquier sistema (no solo en Linux, a pesar del nombre) y que hace que Drupal ejecute automáticamente una serie de comandos, entre los que fundamentalmente está el de buscar actualizaciones del sistema.

Para saber qué acciones realizará ***cron***, tendremos que abrir el archivo **cron.php** que estará en la raíz de nuestro sitio Drupal.

En el menú **Configuración**, dentro del apartado **Sistema** tenemos un enlace para gestionar cron. Desde ese enlace podemos ejecutar cron inmediatamente y programar cada cuanto se ejecuta automáticamente. Además, se nos indica la última vez que cron se ejecutó.

10.7.2 INFORMES DE DRUPAL

Drupal dispone de un apartado llamado **Informes** en el menú de administración que permite examinar diferentes aspectos del funcionamiento de Drupal.

Desde el punto de vista de la administración en Drupal, el más interesante es la revisión de los mensajes recientes en el **registro**, que muestra los logs de Drupal indicando todos los eventos que han ocurrido en Drupal: errores, altas de usuarios, sesiones, tareas de cron, etc.

10.7.3 ACTUALIZACIONES

En el mismo panel de informes visto en el apartado anterior, disponemos de un enlace a las **actualizaciones disponibles**. Si hay actualizaciones, aparecen resaltadas en este panel. Además, se diferencian entre las actualizaciones del núcleo de Drupal, las de cada módulo y las de los temas. Hay además otras actualizaciones de seguridad que pueden aparecer; de hecho, aparecen a menudo para arreglar fallos de seguridad en el sistema. Las actualizaciones de seguridad se deben realizar lo antes posible. Las otras actualizaciones no hace falta hacerlas tan rápido, ya que aportan más funcionalidades, pero no son críticas.

En todo caso cuando aparece una nueva versión, ya sea del núcleo o de componentes de Drupal, conviene leer lo que aporta la actualización para saber si es imperativa su instalación por incorporar parches de seguridad.

Mientras las instalaciones de temas y módulos se hacen directamente desde el panel de actualizaciones sin hacer nada más que clic en **Descargar actualización**, las actualizaciones del núcleo, requieren estos pasos.

[1] Hacer copia de seguridad completa (de la base de datos y de los archivos) de Drupal

[2] Descargar la actualización. Se puede hacer desde el mismo panel de actualizaciones. También podemos descargar versiones concretas desde la página oficial de Drupal.org.

[3] La descarga es un archivo comprimido. La descomprimimos. Tendremos un directorio con la nueva versión.

[4] Borrar todos los directorios de Drupal **excepto el directorio sites**.

[5] Copiar los archivos de la nueva versión de Drupal.

[6] Acceder desde el navegador http://servidor/update.php, donde ***servidor*** es el nombre y ruta a la raíz de Drupal en nuestro servidor.

[7] Comprobar que todo funciona correctamente.

10.8 SEGURIDAD EN DRUPAL

Una vez más, lo dicho en WordPress sobre la seguridad (Apartado 10.4 *"Seguridad en WordPress"*, en la página 425), vale para Drupal por lo que debemos aplicar las mismas acciones en Drupal.

Hay un módulo muy interesante llamado **SecurityReview** que tras ser instalado y activado, nos permite en el apartado **Informes** del menú de administración, ejecutar un informe que nos muestra los riesgos de seguridad que tiene nuestra instalación de Drupal.

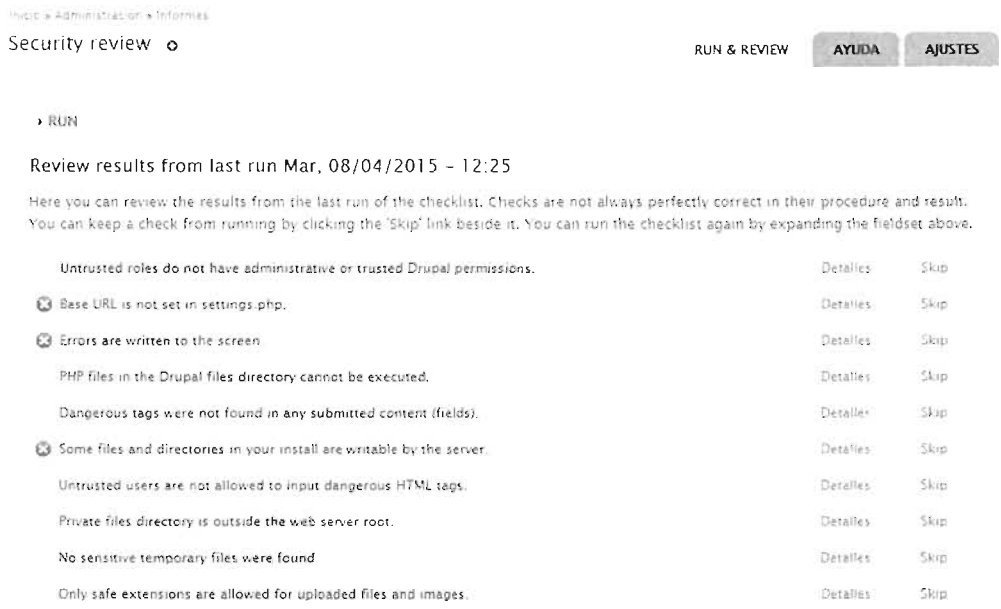


Figura 10.6: Resultado de un informe del módulo Security Review avisando de tres problemas de seguridad

Ante cada ítem del informe disponemos del enlace **Detalles**, que nos permite leer (eso sí, en inglés) cómo solucionar el problema y lo que implica no hacerlo.

En la parte superior, la pestaña de Ajustes permite configurar exactamente lo que este módulo verifica. Es fundamental marcar qué roles de usuario son inseguros y cuáles no.

Los errores típicos son:

- ***Untrusted roles do not have administrative or trusted Drupal permissions.*** Aparece marcado cuando a un usuario no seguro, se le han asignado permisos administrativos.
- ***Private files directory is outside the web server root.*** Indica si el directorio de archivos privados está fuera del servidor. Para arreglarlo hay que cambiar la ruta de los directorios de archivos privados, que se usan, por ejemplo, para copias de seguridad (véase Apartado 10.6.1.1 *"Realización de copias de seguridad"*, en la página 431).
- ***Base URL us not set in settings.php***. Se trata de que no se ha indicado correctamente la URL base del sitio web. Para ello, modifica el archivo **settings.php** que está en la ruta **sites/default** y coloca este código (seguramente baste con descomentar y modificar una línea que ya existe), en el que ***localhost*** se cambia, si es necesario, por la URL a tu servidor:

```
$base_url = 'http://localhost/drupal';
```

- *Errors are written to the screen*. En este caso avisa de que los errores en el código de Drupal salen por pantalla. Para evitar hay que ir a **Registro-Desarrollo-Mensajes** de error y marcar **Ninguno**. Si estamos modificando Drupal no es conveniente marcar esa casilla.
- *Some files and directories in your install are writable by the server*. Avisa de que se pueden modificar archivos de Drupal. Por ello, es mejor cambiar los permisos, por ejemplo, con el comando **chmod**.
- *Untrusted users are allowed to input dangerous HTML tags*. No conviene que los usuarios utilicen el formato **Full HTML**. Para evitarlo, hay que ir al apartado **Configuración-Autoría del contenido-Formatos de texto** y desactivar de ese formato a los usuarios que no son seguros. Solo los administradores deberían poder usar **Full HTML**:
- *Only safe extensions are allowed for uploaded files and images*. Indica que los cuadros de adjuntar imagen solo admiten formatos seguros (GIF, JPEG y PNG). Para arreglarlo hay que revisar los campos de imagen en cada tipo de contenido.

10.9 PRÁCTICAS RESUELTAS

Práctica 10.1: Establecer cuentas de usuario en WordPress

- Crea un usuario editor, dos autores y un colaborador.
- Haz que cada autor y el usuario colaborador publiquen una entrada.
- Intenta que el colaborador modifique su propio artículo.
- Usa el editor para modificar algo en los tres artículos.
- Comprueba que el autor ve las modificaciones.
- Consigue que los usuarios puedan publicar comentarios solo si se dan de alta en el sistema al menos como seguidores.
- Establece una política que prohíba que se publiquen comentarios que contengan la palabra "violencia".
- Comprueba que los autores no pueden añadir categorías a los post, pero sí etiquetas. Usa el usuario editor para establecer categorías en los artículos escritos.

SOLUCIÓN: PRÁCTICA 10.1

[1] Suponemos iniciada la sesión con el usuario administrador. Para crear los usuarios vamos al apartado **Usuarios** del panel de administración y elegimos **Añadir nuevo**

[2] Creamos una primera usuaria de nombre ***luisa***, elegimos un e-mail cualquiera y la contraseña. Dejamos sin rellenar el resto. Marcamos a luisa como **Editora**.

[3] Repetimos el proceso y creamos al usuario **Pedro** y a **Marta** como Autores.

[4] Añade a **Carlos** como colaborador. Comprueba en el panel de usuarios que todos los usuarios aparecen con el perfil apropiado:

Nombre de usuario	Nombre	Correo electrónico	Perfil	Entradas
Carlos		carlos@jorgesanchez.net	Colaborador	0
jefewp1835		info@jorgesanchez.net	Administrador	2
Luisa		luisa@jorgesanchez.net	Editor	0
Marta		marta@jorgesanchez.net	Autor	0
pedro		pedro@jorgesanchez.net	Autor	0

[5] Cierra la sesión con el usuario administrador; basta ir al icono superior derecho (que tiene el nombre de usuario) y elegir **Cerrar sesión**.

[6] Entra con el usuario Pedro y entra en el panel de entradas. Observa que solo puedes ver las entradas anteriores, pero no modificarlas.

[7] Escribe una entrada cualquiera (desde **Entradas-Añadir nueva**). Observa que no puedes establecer categorías (sí utilizar los que ya hay) pero sí palabras clave.

[8] Cierra sesión y publica contenido con **Marta** y **Carlos**.

[9] Cierra sesión y observa la página como usuario visitante. Observa que no se ha publicado la entrada de Carlos.

[10] Conecta como **Luisa** (usando la ruta del servidor más **/login**). Edita el artículo de Carlos (aunque esté bloqueado, toma posesión) y añade nuevas categorías relacionadas con su texto.

[11] Vuelve a cerrar sesión, entra como Carlos. Observa sus entradas (aparecerá la que escribió antes). Intenta modificarla para comprobar que no puede (aunque sea el autor de la misma).

[12] Cierra sesión y entra como administrador. Vete al apartado Ajustes-Comentarios. Activa la casilla ***Los usuarios deben registrarse para escribir comentarios***. Además en el cuadro de moderación de comentarios escribe la palabra ***violencia***. Guarda los cambios.

[13] Cierra sesión y vete a la página principal. Intenta escribir un comentario, observa que no puedes.

[14] Ahora entra conéctate como usuario Carlos y vete a la portada. Deja un comentario en una entrada. Observa que el comentario debe aprobarse.

[15] Ahora conecta como Luisa y aprueba el comentario anterior.

[16] Escribe como Carlos un segundo comentario en otra entrada. Fíjate que ahora puedes perfectamente publicar el comentario.

[17] Finalmente, escribe un tercer comentario que contenga la palabra ***violencia***. Observa que el comentario queda pendiente de moderación (por utilizar una palabra marcada como problemática).

Práctica 10.2: Roles y permisos en Drupal

- Consigue en Drupal la misma estructura de usuarios que tiene WordPress. Es decir consigue que haya usuarios editores, autores, colaboradores y suscriptores (además de los visitantes).
- Haz que un autor y un colaborador publiquen contenido.
- Haz que un editor modifique el contenido.
- Consigue que los comentarios solo los puedan publicar usuarios registrados.
- Entra con un usuario autor y luego con un suscriptor y publica contenido con cada uno de ellos.

SOLUCIÓN: PRÁCTICA 10.2

[1] Con el usuario administrador, ve al apartado **Configuración-Usuarios-Configuración de la cuenta.** Indica que solo los administradores pueden crear usuarios.

[2] Desde el apartado **Usuarios-Permisos,** en la barra de administración, haz clic en el botón Roles.

[3] Añade los roles: **editor, autor, colaborador** y **suscriptor**.

[4] Haz clic en **Permisos**

[5] Asigna los siguientes permisos a los roles:

- **Editor**: Todos los permisos con los comentarios, ver el panel administrativo, usar el formato Full HTML, asignar todos los permisos que tienen que ver con Bloques, Menús, nodos (**Node**), las rutas (**Path**), revisiones, búsquedas (**Search**) y taxonomía.
- **Autor**: Además de lo ya marcado podremos editar nuestros propios comentarios, ver contenido propio sin publicar, ver revisiones del contenido, y crear contenido propio, borrar contenido propio y editar contenido propio de los artículos (ignoraremos foros y contenido estático).
- **Colaborador**: Marcaremos Crear contenido nuevo de Artículos.
- **Suscriptor**: No marcaremos nada más que lo que hay
- A todos los usuarios dales permiso de ver el panel administrativo (no es lo normal, pero nos sirve con fines didácticos).

[6] Guarda los cambios

[7] Añade a la usuaria **Luisa** como editora, activa la cuenta y no hagas ninguna notificación a su correo.

[8] Añade a **Marta** como autora

[9] Añade a **Carlos** como colaborador

[10] Añade a **Susana** como suscriptora

NOMBRE DE USUARIO	ESTADO	ROLES	MIEMBRO DURANTE	ULTIMO ACCESO	OPERACIONES
susana	activo	• suscriptor	4 segs	nunca	editar
carlos	activo	• colaborador	24 segs	nunca	editar
pedro	activo	• autor	1 min 38 segs	nunca	editar
marta	activo	• autor	1 min 54 segs	nunca	editar
luisa	activo	• editor	2 mins 20 segs	nunca	editar
jorge	activo	• astronomo	2 días 3 horas	hace 2 días 3 horas	editar
jefedo1210	activo	• administrator	4 días 49 mins	hace 4 segs	editar

[11] Cierra la sesión. Conecta como **Marta** y publica contenido

[12] Cierra la sesión. Conecta como **Carlos** y publica contenido

[13] Cierra la sesión. Conecta como **Susana** y publica contenido. No te debería dejar. Publica un comentario

[14] Conecta con **Luisa** e intenta modificar los comentarios y entradas anteriores. Para ello desde el menú de administración haz clic en el enlace **Contenidos**.

[15] Conecta con Marta e intenta modificar la entrada de Carlos. No podrá. Intenta modificar sus propios artículos. Sí puede, disponemos de la pestaña **Editar**.

[16] Haz lo mismo con Carlos y observa las diferencias: no puede editar ningún artículo.

Práctica 10.3: Comprobar errores de seguridad en Drupal

- Instala el módulo **Security Review**, comprueba y arregla los errores que indique

SOLUCIÓN: PRÁCTICA 10.3

[1] Ve a la dirección https://www.drupal.org/project/security_review y descarga el archivo comprimido que desees. O bien, coge la dirección y descárgala desde la consola mediante **wget**.

[2] Como administrador ir al apartado **Módulos** y hacer clic en **Instalar nuevo módulo.** Haz clic en **Instalar desde archivo** y elige el archivo descargado.

[3] Activa el módulo (estará hacia el final del panel).

[4] Ve al apartado **Informes-Security Review**. Aparecerá una pantalla de información.

[5] Observa errores y sugerencias.

[6] Corrige los errores

10.10 PRÁCTICAS PROPUESTAS

Práctica 10.4: Crear una estructura compleja de usuarios y contenidos en Drupal

- Consigue crear un sitio web mediante Drupal en el que tengamos tres tipos de contenidos:
  - Electrónica
  - Informática
  - Cultura general
- En todos los tipos contenidos se aceptarán palabras clave (pueden ser de la taxonomía tags que ya existe) e imágenes. En electrónica e informática, además, habrá un campo que permita indicar (opcionalmente) un modelo de artículo al que puede que se refiera la entrada. Ese campo solo aceptará texto plano.
- Crea un usuario para cada tipo de contenido, de modo que solo pueda escribir artículos de un tipo Habrá un autor de electrónica, otro de informática y otro de cultura general
- No habrá más tipos de contenido que esos tres
- La portada muestra todas las entradas (sean del tipo que sean) en orden cronológico
- Además muestra un menú que nos permita acceder a cada tipo de contenido
- Crea un usuario más que tenga permiso de escribir y gestionar cualquier tipo de contenido
- Consigue que solo la cultura general acepte comentarios

Práctica 10.5: Crear una estructura compleja de usuarios y contenidos en Joomla!

- Consigue la misma estructura anterior en Joomla!
- Observa las diferencias de gestión y las implicaciones que tiene

Práctica 10.6: Crear una estructura compleja de usuarios y contenidos en WordPress

- Intenta conseguir la misma estructura en WordPress
- Observa las diferencias de gestión y las implicaciones que tiene

10.11 RESUMEN DE LA UNIDAD

- En un CMS hay diferentes tipos de usuarios. Cada tipo de usuario tiene asignado un rol; es decir, un conjunto de permisos.
- En el caso de Wordpress hay 5 tipos de usuario: administrador, editor, autor, colaborador y seguidor.
- Drupal no tiene límite en el número de tipos de usuario, ya que se pueden crear indefinidamente nuevos roles.
- Puesto que las entradas y páginas las pueden haber editado varios usuarios, se permite realizar seguimiento de las mismas, e incluso anular algunas de esas revisiones.
- Establecer una política de copia de seguridad es vital para todo sitio gestionado por CMS. Se pueden realizar copias a mano, pero se dispone de módulos que automatizan esta tarea e incluso de servicios de alojamiento en la nube, para nuestras copias.
- Es posible exportar datos de una instalación a otra e incluso, de forma muy limitada, exportar datos de un CMS a otro.
- Tener el sistema actualizado es importante para no tener agujeros de seguridad en el mismo. Módulos, plugins y temas deben estar siempre al día. El núcleo del CMS no tan a menudo porque en muchos casos implica una parada del servicio antes de realizar la actualización.
- Siempre hay que tomar medidas de seguridad de alto nivel en los sitios administrados por CMS las cuales incluyen:
  - Nombres crípticos para bases de datos y usuarios.
  - Contraseñas muy complejas.
  - Evitar dar salida a mensajes de error del sistema y minimizar la información sobre el mismo.
  - Ocultar y/o denegar el acceso a los archivos de configuración.
  - Proteger contra escritura el código fuente.
  - Usar códigos de sal aleatorio en el cifrado de las claves.
  - Cifrar la comunicación mediante SSL.
  - Evitar dar posibilidad de escribir HTML de altas capacidades a usuarios no confiables.
  - Permitir subir solo archivos con extensiones controladas.

10.12 TEST DE REPASO

¿Cuáles de estas tareas no pueden ser realizadas por un usuario Editor en WordPress?

a) Escribir entradas
b) Editar entradas propias
c) Editar post de otros usuarios
d) Crear usuarios
e) Hacer comentarios
f) Eliminar comentarios de otros usuarios

¿Cuáles de estas tareas no pueden ser realizadas por un usuario Autor en WordPress?

a) Escribir entradas
b) Editar entradas propias
c) Editar entradas de otros usuarios
d) Crear usuarios
e) Hacer comentarios
f) Eliminar comentarios de otros usuarios

3.- ¿Cuáles de estas tareas no pueden ser realizadas por un usuario Colaborador en WordPress?

a) Escribir entradas
b) Editar entradas propias
c) Editar entradas de otros usuarios
d) Crear usuarios
e) Hacer comentarios
f) Eliminar comentarios de otros usuarios

¿Cuáles de estas tareas no pueden ser realizadas por un usuario Seguidor en WordPress?

a) Escribir entradas
b) Editar entradas propias
c) Editar entradas de otros usuarios
d) Crear usuarios
e) Hacer comentarios
f) Eliminar comentarios de otros usuarios

¿Qué ocurre en WordPress si dos usuarios intentan modificar a la vez la misma entrada?

a) Solo uno puede acceder a editar, bloqueando al otro usuario
b) Pueden editar ambos usuarios a la vez sin problemas
c) Cada entrada en WordPress solo pertenece a un usuario, ninguno más la puede editar
d) Para modificar un post hay que bloquearle en el sitio web. De otro modo, no se permite ni leerlo.

¿Para qué sirve la constante de WordPress WP_POST_REVISION?

a) Para modificar el aspecto del panel de revisiones
b) Para limitar el número máximo de revisiones de cada entrada
c) Para indicar si permitimos revisiones o no
d) Las dos respuestas anteriores son correctas

La importación de datos incorporada a WordPress...

a) Solo permite recoger datos de sistemas WordPress
b) Permite recoger datos de varios servicios y CMS
c) Permite recoger datos de cualquier CMS

8.- Al actualizar el sistema WordPress...

a) Hay que hacer copia de seguridad de archivos y datos
b) Hay que hacer copia de seguridad de los archivos
c) Hay que hacer copia de seguridad de los datos
d) No es necesario hacer copia de seguridad

¿Cuáles de las siguientes medidas de seguridad no son necesarias para un sistema WordPress?

- Utilizar contraseñas muy complejas
- b) Ocultar la versión de WordPress
- c) Quitar el acceso al archivo de configuración
- d) Hacer que el administrador no se llame *admin*
- e) Todas las medidas anteriores son necesarias

10. ¿Qué tipos de usuarios tiene Drupal?
- a) Autentificados y anónimos
- b) Administradores, editores, autores y anónimos
- c) Administradores, editores, autores, autentificados y anónimos
- d) Autentificados, visitantes y anónimos

11. Al actualizar módulos en Drupal...
- a) Hay que hacer copia de seguridad de archivos y datos
- b) Hay que hacer copia de seguridad de los archivos
- c) Hay que hacer copia de seguridad de los datos
- d) No es necesario hacer copia de seguridad¿

12. De cuántos roles dispone Drupal?
- a) De tantos como queramos
- b) De cinco
- c) De tres
- d) De dos

13. ¿Qué es Cron?
- a) El panel de administración de Drupal
- b) El archivo que almacena las tareas que va realizando el sistema
- c) Un módulo de automatización de tareas programadas
- d) El reloj del sistema que utiliza Drupal

¿Qué personas pueden crear cuentas de usuarios en Drupal?

- Solo los usuarios con rol de administrador
- Los usuarios con rol de administrador y los propios visitantes
- Los usuarios con rol de administrador y los propios visitantes, solo con la aprobación del administrador
- Los usuarios con rol de administrador y los propios visitantes, solo si verifican su e-mail

15. Para hacer copias de seguridad en Drupal...
- Hay que utilizar la herramienta Backup and Migrate que es parte del núcleo de Drupal
- Hay que instalar obligatoriamente módulos de terceros, como por ejemplo Backup and Migrate
- c) Hay que utilizar servicios externos ajenos a Drupal
- d) Se pueden hacer manualmente, sin ayuda de módulos ni servicios de terceros

16. Las decisiones de seguridad en Drupal...
- a) Son muy diferentes de las que hay que tomar en WordPress
- b) Son prácticamente las mismas que hay que tomar en WordPress
- c) Se toman a partir del informe que redacta el módulo SecurityReview mediante el cual tenemos la certeza de que no hay ningún problema de seguridad
- d) No son necesarias ya que Drupal es totalmente seguro, a diferencia de WordPress

UNIDAD 11

IMPLANTACIÓN DE APLICACIONES DE OFIMÁTICA WEB

OBJETIVOS

- Reconocer la utilidad de las aplicaciones de ofimática web
- Distinguir las ventajas y desventajas que aportan los servicios de ofimática en la web
- Identificar las principales soluciones de ofimática web
- Agregar usuarios a servicios de ofimática en línea
- Crear documentos a través de un servicio de ofimática web
- Elaborar documentos de forma colaborativa aprovechando las principales opciones de trabajo de los productos de ofimática online

CONTENIDOS

11.1 APLICACIONES DE OFIMÁTICA WEB

La aparición del *Cloud Computing* ha supuesto un cambio de forma de trabajar a casi todos los usuarios. La aparición de herramientas de todo tipo que almacenan los datos del usuario en Internet ha tenido también una gran influencia en la forma de trabajar en el mundo empresarial.

En la informática clásica de oficina, una persona se sentaba en su puesto de trabajo, creaba o modificaba sus documentos y los almacenaba. Si tenía que utilizar otro equipo, debía llevarse los documentos. Una mejora al respecto, fue el hecho de que se utilizaran directorios compartidos en red, de modo que el almacenamiento de los documentos se centraliza y no importa ya en qué equipo estamos sentados. Pero la cuestión es si queremos trabajar fuera de la oficina, incluso en el tren que nos lleva al trabajo, utilizando cualquier tipo de dispositivo, incluidos nuestros dispositivos móviles.

Está cambiando la forma de trabajar y ahora se posibilita que los trabajadores utilicen sus propios dispositivos, en lo que se conoce como estrategia ***BYOD Bring Your Own Device (trae tu propio dispositivo)***. Se intenta que el acceso a Internet de estos dispositivos, garantice el acceso a los documentos de trabajo.

El software de oficina más utilizado indudablemente es **Microsoft Office**. Trabajar con documentos de Office siempre ha tenido la pega de que el dispositivo que utilicemos debe tener instalado este software. Conocedores de este hecho y de las posibilidades de la computación en la nube, aparecieron aplicaciones **SaaS** capaces de crear y modificar documentos online, en el que el único requisito es disponer de un navegador o una aplicación móvil que acceda al servicio.

La influencia que han tenido este tipo de herramientas en estos años ha producido el término Office 2.0 para hablar de las herramientas de productividad online en el mundo de la oficina.

Actualmente estas herramientas se concentran en productos que intentan cubrir todas las necesidades de uso profesional de los usuarios y además de las herramientas ofimáticas clásicas (procesador de textos, hojas de cálculo, software de presentaciones, agenda...) incluyen otras relacionadas con los servicios fundamentales en la nube como el correo electrónico, calendario, centralización de contactos, espacio de almacenamiento de archivos, chat, etc.

11.2 VENTAJAS Y DESVENTAJAS

11.2.1 VENTAJAS DE LAS HERRAMIENTAS DE OFIMÁTICA WEB

- **Mínimos requisitos de instalación.** Basta conexión a Internet y un navegador web, un poco actualizado.
- **Mínimos requisitos de almacenamiento.** Todas las soluciones de ofimática web incluyen espacio de almacenamiento en la nube.
- **Centralización de los documentos.** Podemos acceder desde cualquier dispositivo y siempre veremos el último trabajo que hemos realizado.

- **Acceso multiplataforma**. Podemos trabajar en los documentos independientemente de cuál sea nuestro sistema, sea Windows, Linux, Unix, Mac, Android, etc.
- **Mayor seguridad**. Podemos matizar esta ventaja, pero lo cierto es que no tenemos que preocuparnos por los virus que puede albergar el documento, ya que no estamos almacenando los documentos en nuestros ordenadores. El control de acceso también suele ser más fiable porque las empresas que ofrecen estos servicios, en principio, poseen mecanismos más avanzados de seguridad.
- **Facilidad para la colaboración**. Una de las claves del éxito de este tipo de herramientas. La centralización de los documentos permite que puedan ser utilizados por diferentes personas que pueden aportar sus modificaciones. La mayoría de herramientas, además, ofrece control de revisiones que nos posibilita visualizar las modificaciones al documento realizadas por cada usuario, incluso al instante.

11.2.2 DESVENTAJAS DE LAS HERRAMIENTAS DE OFIMÁTICA WEB

- **Menor potencia**. Las herramientas de oficina clásicas que se instalan en el equipo ofrecen, en principio, una mayor potencia y rapidez. Las herramientas online dependen de la potencia que les ofrecen los navegadores y siempre es menor que trabajar de forma local en el equipo. Las aplicaciones instaladas aportan más funciones, objetivamente hablando, que las que se utilizan online.

  Esta es la razón por la que los equipos de tipo ***thin client***, no terminan de despegar. La mayoría de personas prefieren seguir teniendo instalada su herramienta ofimática favorita; lo que no impide que también utilicen herramientas de ofimática online.
- **Menor tradición**. Cada vez tiene menos peso esta desventaja, pero sigue siendo influyente. La forma de trabajar de los usuarios que llevan años y años utilizando las herramientas clásicas como **Microsoft Office** u **OpenOffice**, son difíciles de cambiar. Este hecho hace que muchas personas que trabajan con gran fluidez con las herramientas clásicas, sean menos productivas utilizando herramientas online. Aunque el cambio parece sencillo, no lo es tanto precisamente para aquellas personas que utilizan fluidamente el software de oficina clásico.
- **Delegación del control de la información**. Los documentos empresariales, fácilmente, contienen información privada que, en ningún caso, se desea exponer a la vista de cualquier persona. En el momento que se utilizan servicios de oficina en la web, estamos cediendo el control de nuestros documentos a la empresa propietaria del software. Hay que tener en cuenta estas cuestiones:
  - **Seguridad del servicio en la web**. Necesitamos tener la absoluta seguridad de que el servicio ofrece mecanismos que no permiten el acceso no autorizado.
  - **Eliminación de la información cuando abandonemos el servicio**. La cuestión es, qué ocurre si no deseamos seguir utilizando el servicio. ¿Se eliminará en los servidores del propietario del software esa información?

- **Legislación del propietario del software**. Es un problema propio de Internet. La empresa que aloja nuestros contenidos puede pertenecer a otro país, por lo que la legislación que se aplica a estos contenidos en cuanto a privacidad, propiedad intelectual, etc, puede ser distinta respecto a la de la empresa que creó los contenidos.

  Ha habido en estos años actuaciones en ciertos gobiernos que han sido criticadas por algunos sectores, al acceder, dichos gobiernos, a contenidos privados gracias a leyes especiales de seguridad, antiespionaje, etc.

- **Copias de seguridad.** Es un problema debido al mismo enfoque del punto anterior. Si todo lo guardamos en un servidor externo a nuestra empresa ¿cómo recuperar la información en el caso de que nos demos de baja del servicio? Es un aspecto a veces olvidado, pero fundamental. Debemos de poder tener en local el documento, de otro modo dependemos en exceso del servicio.

- **Compatibilidad.** En base a esta misma idea, utilizar un formato de documentos privado, que solo es utilizable por el servicio de ofimática web concreto que estamos usando, puede restringir de forma dramática la posibilidad de cambiar a otro servicio, al no poder convertir los documentos.

11.2.3 CONSECUENCIAS DE LAS VENTAJAS Y LAS DESVENTAJAS

Parece claro que las herramientas de ofimática web son una excelente opción, especialmente para el trabajo empresarial colaborativo. Las ventajas son suficientes para considerarlas como una herramienta casi imprescindible en las tareas de oficina. Hay que tener en cuenta las desventajas para paliarlas, en la medida de lo posible.

En el caso de que utilizar herramientas online en Internet suponga abrir los problemas más de lo debido, siempre se pueden utilizar soluciones instalables y distribuibles en la Intranet empresarial, como es el caso de **SharePoint** de Microsoft y otras.

11.3 SOLUCIONES DE OFIMÁTICA WEB

11.3.1 GOOGLE DOCS. GOOGLE APPS

Google Docs fue el servicio que revolucionó este área. No fue el primero (aunque sí de los primeros) pero sí el que tuvo más influencia. Su virtud principal es la sencillez de uso. Por otro lado es totalmente compatible con los documentos procedentes de Microsoft Office, lo que soluciona la conversión de documentos ya existentes.

Actualmente Google Docs ya no es independiente, sino que se trata de un servicio integrado en **Google Drive**. Google Drive es el servicio de almacenamiento online de la empresa Google, el cual ofrece espacio gratuito, ampliable mediante pago mensual, y la creación de documentos de texto con formato, hojas de cálculo, presentaciones y formularios en línea para recabar datos.

Permite trabajar sin conexión, de modo que, cuando creemos documentos en nuestro ordenador, éstos se sincronizan automáticamente en el espacio en la nube. Es accesible desde la web y además a través de aplicaciones móviles. Es capaz de abrir documentos de Microsoft Office y se integra con el resto de servicios de Google. Esto último, junto con su interfaz sencilla y minimalista, es su gran baza.

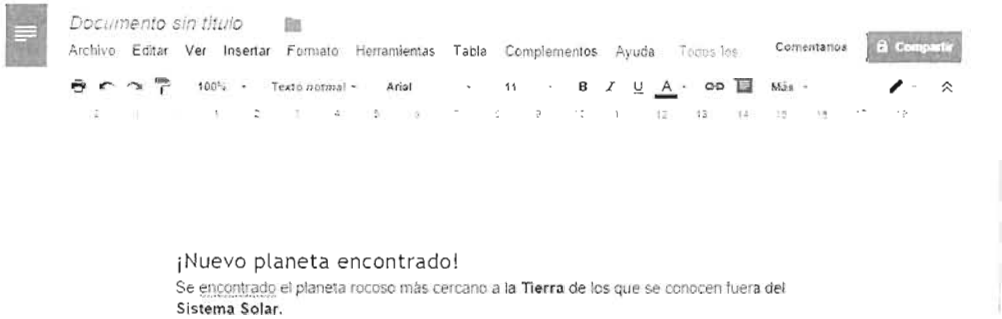


Figura 11.1: Aspecto del procesador de texto de Google Docs.

Google Apps es un servicio online de productividad empresarial que integra a Google Drive (y por lo tanto a Google Docs) junto con el correo electrónico (**Gmail**), calendario (**Google Calendar**), servicio de conferencias (**Hangout**), contactos (**Google Contacts** y **Google+**) y creación de sitios web tanto privados como públicos (**Google Sites**). Es un servicio de pago para empresas y gratuito para entidades educativas.

11.3.2 MICROSOFT OFFICE ONLINE. MICROSOFT OFFICE 365

La versión de Microsoft para su propia versión de servicio de ofimática web se llamó inicialmente **Office Web Apps**. Actualmente se llama **Microsoft Office Online** y es una versión gratuita y solo accesible online de sus clásicos **Word**, **Excel** y **PowerPoint**, a los que se han añadido **OneNote** (software de creación de anotaciones), **Sway** (software de exposición de ideas), **Microsoft Calendar**, control de contactos y el gestor de correo **Microsoft Outlook**.

El espacio que se utiliza para almacenar los documentos en la nube es el que ofrece Microsoft a través de **OneDrive**, con el que mantiene un servicio integral. OneDrive proporciona espacio gratuito (actualmente 16 GB) que, una vez más, se puede ampliar mediante pago.

Microsoft Office 365 es el servicio de suscripción (mediante pago, aunque es gratis para instituciones educativas) a las herramientas de productividad para empresas de Microsoft. Aporta más espacio, comparado con el servicio de Office Online y más servicios: correo, control de cuentas, chat online y calendarios empresariales. Hay opciones (incrementando el precio) de utilizar el clásico Microsoft Office en su versión habitual de escritorio.

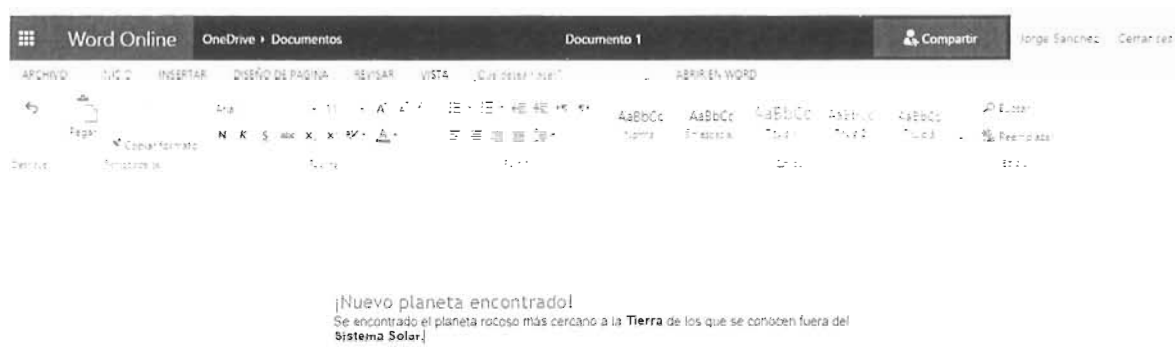


Figura 11.2: Aspecto de Microsoft Office Word Online, prácticamente indistinguible respecto a su versión de escritorio

Office Online permite trabajar sin conexión y su ventaja sobre el resto, es el conocimiento que la mayoría de usuarios poseen del software Microsoft Office. Además, puesto que OneDrive aparece integrado en las últimas versiones de Windows (especialmente en Windows 10), los usuarios de Microsoft pueden directamente utilizar el servicio.

Microsoft posee otro producto de pago muy conocido llamado **SharePoint** que permite crear un servidor de documentos en la Intranet o la nube empresarial privada. Al igual que las herramientas en la nube, podemos crear, examinar y almacenar documentos de Microsoft Office vía navegador web. La diferencia es que solo podemos acceder si estamos conectados en la Intranet de la empresa.

11.3.3 ZOHO DOCS

Solución ya veterana para generar documentos ofimáticos vía web. Ofrece capacidades similares a las soluciones de Google y Microsoft. Además, mantienen la compatibilidad con los documentos de Microsoft Office. Permite trabajar sin conexión y también aporta almacenamiento en línea.

Su oferta gratuita se compone de 1 GB de almacenamiento para cada usuario, sincronización con archivos en el escritorio del usuario, procesador de textos, hojas de cálculo, software de presentación, control de versiones y administración de usuarios. Es gratis hasta 25 usuarios, a partir de ahí, proporciona más espacio de almacenamiento online y prestaciones avanzadas.

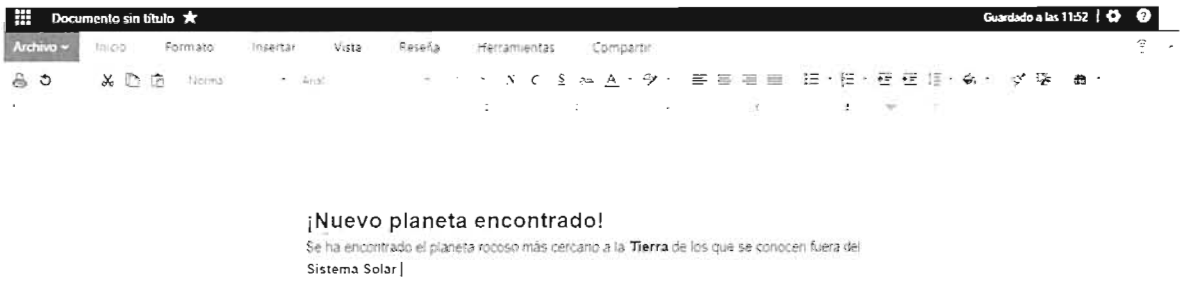


Figura 11.3: Aspecto del procesador de texto de Zoho Online. Recuerda muchísimo a Microsoft Word.

Las aplicaciones mantienen una estética y funciones basadas en Microsoft Office, por lo que la curva de aprendizaje es casi inexistente.

A pesar de su rapidez y funcionamiento correcto, la dificultad para competir con Google y Microsoft que tienen una pléyade espectacular de usuarios, gracias al éxito de sus sistemas operativos, ha hecho que el enfoque de Zoho se dirija al mundo empresarial, aportando soluciones online para aplicaciones empresariales de alto nivel.

11.3.4 THINKFREE ONLINE

Otra de las alternativas como aplicación de ofimática web. Ofrece almacenamiento gratuito, trabajar sin conexión, compatibilidad con Microsoft Office, etc. En definitiva, prestaciones similares a las otras soluciones.

Comparado a los anteriores parece menos ligero y utiliza el plugin de Java que lo hace, hoy en día, bastante incómodo. La apariencia que utiliza es la de las versiones de Microsoft Office anteriores a la 2007.

Como le ocurre a Zoho, a pesar de ser también gratuito no consigue rivalizar con los servicios de Google y Microsoft. Por ello se ha especializado en un servicio de ofimática web en Intranet que se llama **Thinkfree Server** y que compite directamente con Microsoft SharePoint. Permite la creación de un servidor interno capaz de ofrecer un servicio en la red empresarial de ofimática web.

11.4 USO DE LOS SERVICIOS DE OFIMÁTICA ONLINE DE GOOGLE DRIVE

11.4.1 EMPEZAR A UTILIZAR EL SERVICIO

Como se ha comentado anteriormente, la solución de ofimática online de Google está integrada dentro de Google Drive que ofrece soluciones más completas al trabajo diario de oficina.

Utilizar Google Drive, y por lo tanto Google Docs, requiere poseer una cuenta con Google. Es fácil que un usuario actual ya tenga una cuenta con Google ya que la mayoría de usuarios de móviles Android tienen que crear una cuenta en la nube de Google para acceder a sus servicios. También tendremos una cuenta de Google si nos hemos dado de alta en la red social Google+ o en cualquier otro servicio de Google.

En todo caso, si no tenemos una cuenta de Google, el proceso para el alta es el siguiente:

[1] **Acceder a la URL:** https://accounts.google.com

Figura 11.4: Página de acceso a los servicios de Google. https://accounts.google.com

[2] Hacer clic en el enlace **Crear cuenta**. Si tuviéramos cuenta ya, bastaría con indicar el correo y la contraseña.

[3] A continuación se nos pedirán los datos para la cuenta, los cuales son:

- **Nombre y apellidos**
- **E-mail.** En principio Google nos crea un nuevo buzón de correo en su servicio **Gmail**. También podemos crear la cuenta de Google con nuestro propio correo, en ese caso tendremos que verificar dicho correo. En todo caso, el email que indiquemos será nuestro nombre de usuario para Google.
- **Contraseña**. Como siempre, lo más compleja posible para evitar ataques de fuerza bruta y hacerse con la autoría de nuestra cuenta.
- **Teléfono móvil**. Es un dato importante y casi obligado hoy en día para hacer lo que se conoce como **verificación de dos pasos**. En esta verificación, cada vez que alguien intente acceder a los servicios de Google con nuestro nombre de usuario, recibirá un código que se envía al teléfono móvil asociado a la cuenta. Solo se podrá a acceder a la cuenta indicando ese código.

  Aunque esto hace el acceso más lento es, sin duda, una forma más segura de acceder a nuestra cuenta.
- **Dirección de correo electrónico actual.** Las razones para indicar este dato son semejantes al anterior. En caso de pérdida de la contraseña, o de que necesitemos verificar la propiedad de la cuenta, se utiliza esta segunda cuenta para probar que somos realmente los propietarios. No es obligatorio.
- **Fecha de nacimiento**. Es obligatorio indicarla.

Tras introducir estos datos, se nos pide aceptar las condiciones de uso de nuestros datos y los datos extra que Google recaba de nosotros, que deberíamos leer atentamente.

[4] Ahora estaremos en la dirección https://myaccount.google.com que hace de portal centralizador de nuestros servicios con Google. Podemos hacer clic en el menú ⋮⋮⋮ para acceder al servicio de Google que deseemos. También podemos directamente ir a la dirección https://drive.google.com para acceder a Google Drive directamente.

La pantalla de Google Drive es nuestro escritorio online desde el que podemos revisar, crear, modificar y, en definitiva, realizar todas las acciones con nuestros documentos y archivos en el espacio que tenemos con Google Drive.

11.4.2 INTERFAZ DE GOOGLE DRIVE

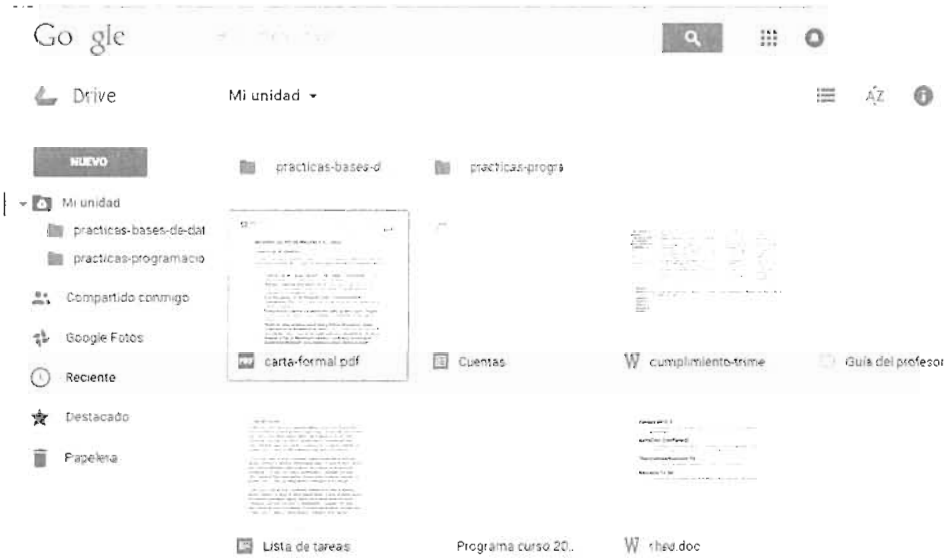


Figura 11.5: Ejemplo de aspecto de Google Drive con algunas carpetas y archivos

Al entrar en Google Drive tenemos un escritorio virtual que controla los archivos y documentos que tenemos en Google Drive. Desde esa interfaz podremos:

- **Navegar por los documentos y carpetas.** Los cuales aparecen en el panel principal de la página. Se navega por ellos igual que en cualquier explorador de archivos de un Sistema Operativo. Para crear documentos en una ubicación específica, debemos llegar hasta la carpeta concreta en la que deseamos examinar o crear nuevos documentos.
- **Visualizar documentos.** Basta simplemente hacer doble clic en ellos. Dependiendo del tipo de archivo se podrá visualizar o no. Normalmente para los tipos de archivo que no son de Google Docs, un menú nos permitirá elegir varios servicios de Internet para abrirlos.
- **Buscar documentos.** Gracias al buscador que Google incorpora en la parte superior; muy útil en el caso de tener muchos documentos.

- **Crear nuevas carpetas.** Simplemente haciendo clic en **Mi unidad** (menú superior) y eligiendo **Nueva carpeta**. La carpeta se crea con el nombre que indiquemos y dentro de la carpeta que estuviéramos examinando.
- **Subir archivos.** Google Drive sirve para almacenar todo tipo de archivos, para subirles desde la página web basta con hacer clic en **Mi unidad-Subir archivos** y después elegir el o los archivos que deseemos.
- **Subir carpetas.** Funciona igual que la opción de subir archivos, pero consiste en elegir una carpeta. La carpeta y todo el contenido se subirá a nuestro espacio de Google Drive.
- **Crear nuevos documentos.** Basta hacer clic en el botón **Nuevo** y elegir el tipo de documento a crear.
- **Eliminar documentos y carpetas.** Haciendo clic en el botón derecho tras seleccionar el archivo o carpeta (podemos seleccionar varios usando las mismas técnicas que en los sistemas operativos: tecla Ctrl, Mayúsculas, dibujar un rectángulo, etc) y luego haciendo clic encima de la selección con el botón secundario del ratón y escogiendo la opción Eliminar del menú que aparece.

  Podemos también simplemente arrastrarlos a la papelera.
- **Mover archivos o carpetas.** En el mismo menú secundario comentado, podemos elegir la opción **Mover a**, la cual realizará la acción sobre los archivos y/o carpetas seleccionados. Se nos preguntará a dónde los moveremos.

  Podemos también directamente arrastrarlos con el ratón a la nueva ubicación.
- **Copiar archivos o carpetas.** Se realiza de la misma forma, pero eligiendo **Copiar** en el menú. Se puede hacer también arrastrando si mantenemos pulsada la tecla **Ctrl** del teclado a la vez que hacemos el arrastre con el ratón.
- **Cambiar nombre.** Opción disponible también en el menú secundario.
- **Descargar.** Opción que permite dejar una copia del archivo en nuestro escritorio. Los archivos de Google Docs, al descargar, se convierten automáticamente a formato de Microsoft Office para ser tratados en el ordenador de escritorio, ya que no hay ninguna herramienta de escritorio para manipular documentos de Google Docs.

11.4.3 EDICIÓN DE DOCUMENTOS

Sea porque hemos hecho doble clic en un documento de Google Docs presente en nuestro escritorio o porque lo creamos nuevo, pasaremos a una ventana nueva en la que se nos presenta la interfaz de modificación de documentos que variará en función del tipo de documento: proceso de texto, hoja de cálculo, presentación, etc.

La interfaz nos será familiar puesto que se parece a los programas ofimáticos de escritorio, por lo tanto no es difícil crear y editar documentos.

Google Docs guarda el documento automáticamente y mantiene las versiones previas. Otras opciones que tienen los editores de Google Docs son:

- Cambiar de nombre al archivo (menú **Archivo**).
- Descargar en otro formato (menú Archivo) como Microsoft Office, PDF u Open Office.
- Imprimir (menú **Archivo**)
- Deshacer y rehacer acciones (menú **Edición**)
- Añadir complementos (menú **Complementos**)

11.4.4 CONVERTIR DOCUMENTOS

Si tenemos archivos de Microsoft Office, u otros formatos compatibles, subidos a nuestro espacio de Google Drive. Simplemente haciendo doble clic en ellos podemos convertirles a formato de Google Docs. Suele aparecer un menú con esta opción, o un menú con el texto **Abrir** en la parte superior. También con el botón secundario del ratón encima del documento, elegir **Abrir con ...**, y después seleccionar la aplicación de Google que aparece en el menú.

Hay que tener en cuenta que tendremos dos versiones del archivo al realizar esta acción: el original y el archivo convertido a formato de Google.

11.4.5 TRABAJO SIN CONEXIÓN

Google Drive (y por lo tanto Google Docs) tiene la posibilidad de trabajar sin conexión, accediendo de forma local a los archivos. No es en absoluto la idea principal del servicio, pero sirve para cuando no tenemos conexión por la razón que sea.

La manera de conseguirlo es hacer clic en el botón ⚙ que se encuentra en la zona superior. Después se hace clic en Configuración y, finalmente, en **Trabajar sin conexión**. Está opción solo está disponible en el navegador **Chrome** de Google, ya que es el único que incluye los plugins necesarios para realizar esta acción.

Una posibilidad más interesante es descargar Google Drive como aplicación para el ordenador de escritorio que tengamos. Eso permite, tras configurar el programa, elegir un directorio en nuestro ordenador (que tiene que ser Windows o Mac) para que se sincronice con nuestro espacio de Google Drive en la nube. De este modo si hacemos un cambio a un documento en Internet, el programa lo detecta y hace que en nuestro escritorio también aparezca.

Al instalar el programa simplemente se nos preguntará por nuestra cuenta en Google. Entonces se creará un directorio dentro del directorio de nuestro perfil de usuario.

Tras la instalación de la aplicación, ésta se arranca cada vez que se inicia sesión en el sistema. Entonces comparará el estado de los archivos en el equipo con los archivos en la nube y sincronizará ambos dejando la última versión (en caso de conflicto dejará dos copias del mismo archivo). Podemos incluso instalar la aplicación en varios ordenadores, de modo que en todos ellos dispongamos de los archivos en un directorio del sistema.

Cuando queramos modificar un archivo de Google Docs en el directorio sincronizado, abrirá nuestro navegador predeterminado y desde él se realizará la edición, ya que Google Docs no dispone de aplicación en el escritorio.

Figura 11.6: Ejemplo de aspecto de la aplicación Drive mostrándose desde el explorador de archivos en un equipo Windows

11.4.6 TRABAJO COLABORATIVO CON GOOGLE DOCS

Indudablemente es una de las grandes ventajas de trabajar con servicios de ofimática online. Como el documento está en la nube cualquier persona (a la que permitamos el acceso) puede editar o ver el documento simplemente con disponer de conexión a Internet y un navegador.

Esta forma de trabajo da una mayor agilidad cuando varias personas deben de realizar un documento conjuntamente.

11.4.6.1 COMPARTIR DOCUMENTOS

Figura 11.7: Panel de selección de usuarios para compartir un documento

El trabajo colaborativo lo inicia un usuario de Google Drive que crea un documento de tipo Google Docs (sea del tipo que sea). A partir de ahí desde el escritorio se puede hacer clic en el botón de compartir ( ) o, si se está editando el documento, hacer clic en el botón **Compartir**.

Después un panel nos permite indicar los usuarios con los que compartimos nuestro documento y de qué manera lo hacemos. Podemos compartir de estas tres formas:

- **Para editar**. Pueden cambiar, añadir o borrar el contenido del documento.
- **Para comentar**. Pueden ver el documento y añadir comentarios. Pero no modificar el mismo.
- **Para ver**. Solo podrán ver el contenido del documento.

En ese mismo panel, podremos indicar un mensaje. Ese mensaje aparece con el email que les llegará a su buzón de correo (cuya dirección hemos indicado en el panel).

Tras pulsar enviar, llegará al buzón de las personas indicadas un enlace hacia el documento (además del mensaje que hemos escrito).

11.4.6.2 COMPARTIR ENLACE

Cuando se hace clic en el botón de compartir, disponemos de la opción **Obtener enlace para compartir**, la cual crea un enlace. Ese enlace permite ver (no editar) el documento. Aquellas personas a las que enviemos este enlace tendrán a su disposición el documento.

También podemos indicar personas concretas y Google se encarga de enviarlas la invitación con el enlace.

Finalmente, el enlace **Avanzadas** nos permite especificar aún mejor qué hacer con el enlace. Podremos:

- Compartir el enlace por **Google+**, **Twitter** y **Facebook**.
- Podemos hacer que se pueda acceder sin siquiera tener el enlace al documento. Se podrá buscar el documento a través de los buscadores de Internet (como el propio Google). Podemos también especificar que sean los usuarios que conozcan el enlace (opción por defecto). También podemos especificar qué usuarios concretos pueden utilizar el enlace.
- El acceso al documento se puede explicitar también en el cuadro (si dejamos solo leer o dejamos escribir).

11.4.6.3 ESCRITURA COLABORATIVA

Google Docs admite que el mismo documento sea modificado por más de una persona. De hecho, incluso admite que se haga esa modificación al mismo tiempo. Es decir, si hay tres personas con permisos de escritura en el documento, y todos les están editando, cada usuario ve marcado en su documento los cursores de los otros, viendo además como van cambiando el documento.

Figura 11.8: Documento que tres usuarios escriben a la vez. Aparecen marcados los cursores de los otros dos usuarios

11.4.6.4 ESCRITURA DE COMENTARIOS

Los comentarios permiten escribir avisos como texto aparte, es decir el texto de los comentarios no forma parte del documento.

Cuando un usuario escribe un comentario, todos los demás le ven (el texto con comentarios aparece resaltado de color amarillo) y, si tienen permiso, pueden incluso responder. Así los comentarios de Google Docs pasan a establecer una forma de chat muy interesante que ayuda al debate a la hora de trabajar un documento.

Figura 11.9: Documento con un comentario. El comentario se ha respondido dos veces.

Cuando un comentario se da por solucionado, se puede hacer clic en el botón **Resolver** que aparece a la derecha del comentario. En cuanto resolvamos un comentario, deja de aparecer en el documento, pero seguirá estando disponible en el botón de Comentarios.

11.4.6.5 ESCRITURA DE SUGERENCIAS

Normalmente el modo de trabajo al escribir documentos en Google Docs es el de edición, en el cual el funcionamiento es el comentado: cada usuario con permiso de escritura en el documento puede escribir y sus cambios se graban al instante (o casi al instante) en el documento.

Los demás usuarios con permisos de escritura (o visualización) ven el texto escrito por el otro usuario, tal cual.

Si embargo si hacemos clic en el botón Edición podemos elegir el modo **Sugerencias**. En este modo los cambios que hagamos al documento no son definitivos. Tienen que ser aprobados por nosotros u otro usuario que tenga permiso. Al escribir en este modo, lo que escribamos aparece resaltado con un subrayado especial de colores, el resto de usuarios también distingue este tipo de escritura por el color

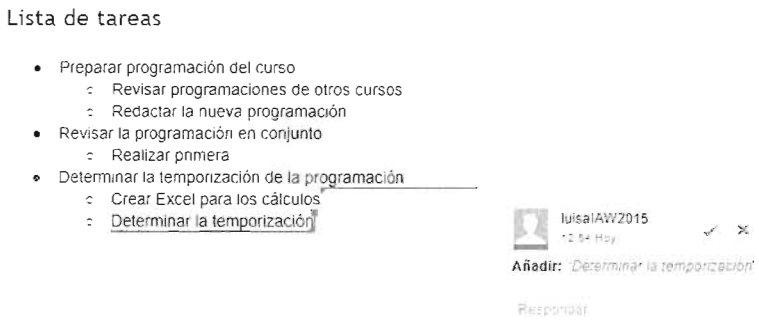


Figura 11.10: Documento con texto en modo de sugerencia.

A partir de ahí, dos botones permiten aceptar o cancelar la sugerencia. Si aceptamos, lo que se ha escrito pasa a ser definitivo. Si cancelamos, se anula el texto. Cualquier usuario con permisos de escritura puede aceptar o cancelar sugerencias.

11.4.6.6 REVISIÓN DE LOS CAMBIOS

Podemos observar el historial de los cambios realizados en un documento a través del enlace que está a la derecha de la ayuda, que también se usa para saber si el documento está guardado o no.

Cada revisión nos muestra los cambios realizados y quién y cuándo lo hizo. Podemos, incluso, restaurar una revisión, dejando el documento en el estado exacto que estuviera en ese momento.

Figura 11.11: Documento mostrando el panel de revisiones.

11.5 PRÁCTICAS PROPUESTAS

Práctica 11.1: Documento colaborativo en Google Docs

- Si no tienes una cuenta en Google Drive, créate una (valdría cualquiera creada en otro servicio de Google).
- A continuación ponte de acuerdo con un par de compañeros que también tengan cuenta.
- Escribid un documento conjunto sobre cualquier tema que se os ocurra
- Revisad lo que cada uno ha escrito
- Ponte en modo sugerencia y espera que otro compañero te apruebe la sugerencia
- Pon a otro compañero en modo lectura con comentarios y pide que escriba un comentario
- Responde a ese comentario y escribe un par más
- Anula el primer comentario
- Escribe un trozo de texto largo y después de unas horas, elimínalo volviendo a la versión previa del documento.

Práctica 11.2: Documento colaborativo en Office Online

- Intenta realizar las mismas acciones con una cuenta de Office Online
- Compara en ambos casos las posibilidades

Práctica 11.3: Presentación online con Google Docs

- Realiza una presentación en Google Docs
- Haz que la presentación pueda ser vista por cualquier persona
- Haz otra presentación y consigue otro enlace para verla, pero ahora consigue que solo dos personas la puedan ver

Práctica 11.4: Formulario online

- Crea un formulario que haga un pequeño test sobre películas favoritas. Una pregunta pedirá elegir el género favorito (Acción, Comedia, Suspense,...). Otra pregunta pedirá de qué país son las películas que más le gustan (EEUU, España, México, Argentina, Francia, Italia...)
- Envía el enlace al formulario a varias personas y consigue que cuando una persona responda al formulario, automáticamente vea los resultados.
- Además haz que los resultados se almacenen en una hoja de cálculo para que puedas analizarlos más fácilmente

11.6 RESUMEN DE LA UNIDAD

- Las aplicaciones de ofimática web posibilitan llevar a la práctica el paradigma BYOD, por el que los trabajadores utilizan sus propios dispositivos para el trabajo. Ya que se accede a un servicio centralizado en Internet y accesible desde cualquier dispositivo con conexión.
- Las ventajas de este software son: mínimos requisitos de instalación y almacenamiento, centralización del documento, acceso multiplataforma, seguridad y facilidad para colaborar.
- Tienen también desventajas, todas ellas relacionadas con el hecho de que los documentos no están bajo nuestro control, dependemos del servicio que estamos utilizando.
- Pueden ser tan importante las desventajas, que podemos optar por servicios de ofimática para el acceso dentro de la Intranet empresarial gracias a software como por ejemplo, SharePoint de Microsoft.
- Los más importantes servicios de ofimática web son: Google Docs, Microsoft Office Online, Zoho Docs y Thinkfree Online. Todos ellos son semejantes, aunque dominan el mercado los dos primeros.
- Google Docs solo requiere para su uso una cuenta con Google. Accediendo al servicio Google Drive podremos crear y editar documentos.
- Sobre cada documento podremos:
  - Trabajar sin conexión.
  - Compartir el documento, sea para editar o solo para verle, a otros usuarios.
  - Escribir de forma colaborativa los contenidos, revisando las diferentes versiones y escribiendo y respondiendo a comentarios.
  - Trabajar en modo sugerencia hasta que otros usuarios (o nosotros mismos) aprueben dicha sugerencia.
  - Organizar los documentos en carpetas y subcarpetas. Estableciendo una organización a medida en nuestra cuenta.
  - Utilizar una carpeta sincronizada en nuestro ordenador de escritorio para tener, directamente en ella, los documentos que vamos editando en la nube. Y viceversa, que en la nube aparezcan los documentos que editamos en nuestro ordenador.

11.7 TEST DE REPASO

A qué hacen referencia las siglas BYOD?

A la prohibición que muchas empresas imponen a sus trabajadores de no usar dispositivos electrónicos personales.

Al hecho de que los propios trabajadores lleven sus propios dispositivos electrónicos.

A que cada vez menos ordenadores y otros dispositivos requieran cable.

d) A que los propios trabajadores tengan datos personales en los dispositivos de la empresa.

¿Cuál de las siguientes no es una ventaja de la ofimática web?

a) Mínimos requisitos de instalación

b) Delegación del control de la información

Facilidad para la colaboración

d) Mayor seguridad

¿Cuál de las siguientes no es una desventaja de la ofimática web?

a) Copias de seguridad

b) Compatibilidad

c) Menor tradición

d) Centralización de los documentos

¿Cuál de las siguientes soluciones para empresas online son gratuitas?

a) Office 365

b) Office Online

c) Google Docs

d) Google Apps

5.- **Los archivos creados con servicios de ofimática web se guardan normalmente...**

a) En servidores en Internet

b) En el ordenador del usuario

c) En la cuenta de correo del usuario

d) No se guardan, solo se pueden utilizar mientras no se cierren

¿En qué formato se descargan los archivos de Google Docs?

Se convierten a formato compatible de Microsoft Office.

En el formato nativo de Google Docs.

En XML, concretamente Open Document

Se guarda solo un enlace, realmente solo se pueden abrir en la nube.

¿Qué tipo de producto es Microsoft Sharepoint?

Un servicio de compartición y creación de documentos, inicialmente orientado a Intranets.

Un servidor web con capacidad de abrir documentos de Word y Excel.

Una producto ofimático de escritorio al estilo de Microsoft Office.

d) Un Sistema Operativo.

8.- **¿En Google Docs si dos usuarios intentan escribir al mismo tiempo?**

a) No se permite escribir al mismo tiempo, el documento se bloquea hasta que el primero que lo abrió, lo cierre.

b) No se permite escribir al mismo tiempo si ocurre, el documento se bloquea hasta que ambos usuarios cierren sesión.

Se permite escribir al mismo tiempo, pero no sabremos que varios usuarios han estado escribiendo a la vez.

Se permite y seremos conscientes de qué usuarios están conectados y lo que están haciendo.

¿Cuál de estas herramientas permiten trabajar offline?

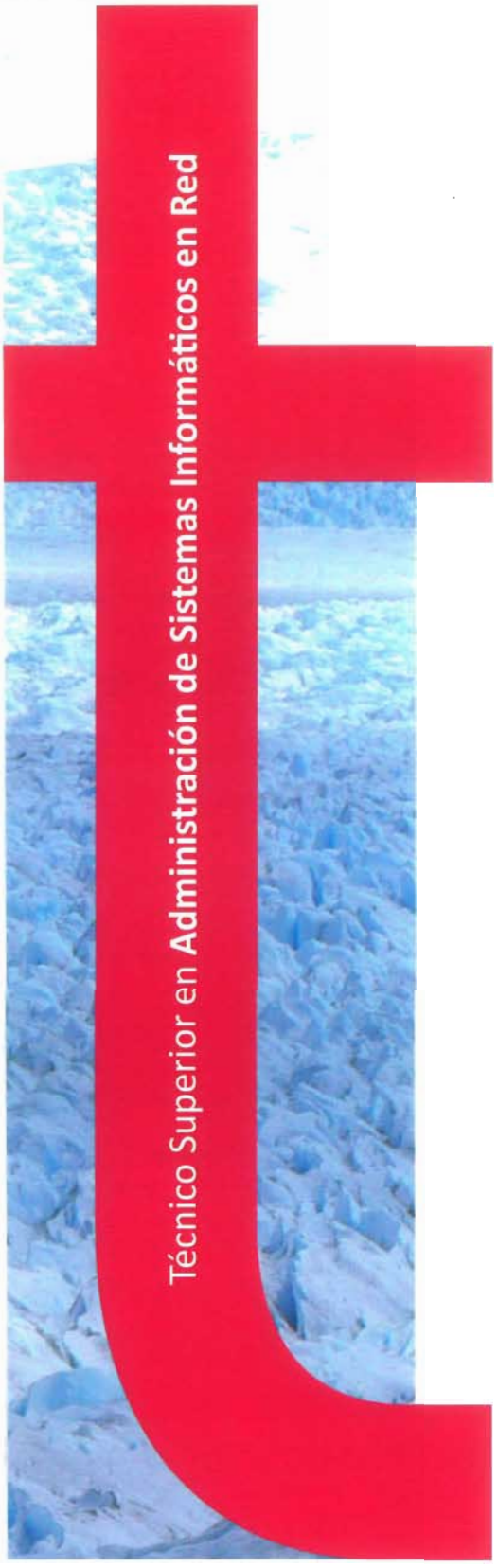
Google Docs

Microsoft Office Online

Zoho Docs

Thinkfree Online

Todas las anteriores



Implantación de Aplicaciones Web

Este libro puede ser de gran utilidad para aquellos estudiantes, docentes y profesionales que quieran empezar a formarse o profundizar en la Implantación de Aplicaciones Web.

Cada unidad está escrita la siguiente forma:

- Se exponen los contenidos teóricos de la misma tratando de fomentar la aplicación inmediata de los mismos, para lo cual se proponen algunas actividades.
- Tras los contenidos se plantean prácticas que se explican y resuelven en el propio texto y que se proponen para realizar a la vez que se estudia la parte teórica.
- Hay una página final de resumen de la unidad con las ideas más interesantes vistas en la misma.
- Al final de cada unidad se han elaborado preguntas de tipo test para evaluar nuestra comprensión del texto.

Apache, PHP, MySQL, WordPress y Drupal, bajo los sistemas Windows y Linux, son las tecnologías sobre las que camina este texto. La idea no es tanto dominar estas tecnologías concretas, como explicarlas con una profundidad suficiente como para trabajar profesionalmente con ellas o migrar a otras tecnologías sin grandes dificultades.

Formación profesional

Garceta grupo editorial

www.garceta.es

978-84-1622-830-0

9 788416 228300