Laboratorio de PHP y MySQL

Piero Berni Millet
Dídac Gil de la Iglesia

PID_00152507

Material docente de la UOC

Universitat Oberta de Catalunya
www.uoc.edu

Piero Berni Millet

Piero Berni Millet está ligado al grupo de investigación CEIPAC del Área de Historia Antigua de la Universidad de Barcelona desde 1989, donde lleva a cabo diferentes proyectos informáticos de bases de datos aplicados a las nuevas tecnologías. Desde 1999 colabora con el grupo de investigación Òliba, de los estudios de Humanidades de la UOC, en el diseño y montaje de diferentes exposiciones virtuales, concebidas como un complemento de exposiciones presenciales, que forman parte de un proyecto de investigación sobre patrimonio, museos y recursos digitales.

Dídac Gil de la Iglesia

Ingeniero informático. Trabaja como gestor de sistemas y aplicaciones, colaborando en proyectos de administración de la seguridad y desarrollando aplicaciones web basadas en bases de datos. Ha realizado varios portales web dinámicos para centros de enseñanza y actualmente está trabajando en aplicaciones inteligentes de monitorización y pronto aviso de ataques informáticos.

Primera edición: febrero 2010

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Eureca Media, SL
ISBN: 978-84-692-9427-7
Depósito legal: B-8.145-2010

Contenidos

Puesta en marcha de un SGBD y un servidor web local

Piero Berni Millet
Dídac Gil de la Iglesia

PID_00155713

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En este apartado aprenderéis a instalar un sistema de gestión de bases de datos local para Windows con el que tendréis ocasión de publicar datos en Internet de forma dinámica y en tiempo real.

El sistema estará compuesto, entonces, por los siguientes elementos:

1) Sistema operativo Windows 95/98/Me/NT/2000/XP/Vista.
2) Apache como servidor de web.
3) PHP como módulo de ampliación de Apache para acceder a la base de datos.
4) MySQL como servidor de base de datos.

General public license

A excepción de Windows, podemos obtener todos los elementos del sistema sin ningún tipo de coste al tener licencia GPL (*general public license*). Las bases de esta licencia se encuentran especificadas en la dirección de Internet: http://www.gnu.org/copyleft/gpl.html

En este apartado nos gustaría facilitar al lector la tarea de instalar el sistema completo desde el principio. Para ello, detallaremos los pasos que hay que seguir para la compilación e instalación de los diferentes paquetes, aunque la última palabra siempre la tendrá la documentación de cada paquete en particular. En algún caso, quizá la descripción que hacemos aquí no sea lo suficientemente completa, por lo que el estudiante tendrá que recurrir a las instrucciones de instalación de cada uno de los paquetes.

Una vez completados estos cuatro procesos de instalación, habremos conseguido el objetivo principal de esta guía, objetivo que culminará con la creación de una simple web de ejemplo conectada a una base de datos.

1. Puesta en marcha de un SGBD y un servidor local

1.1. ¿Qué es Apache?

Apache es un servidor de web. Un servidor web es un software que responde a las solicitudes de los navegadores web. En estos momentos, Apache es uno de los servidores web más populares del mundo. Ello se debe, entre otras cosas, a que Apache es un software de alta calidad y de código abierto (*open source*), lo que significa que puede descargarse de forma gratuita desde Internet.

Apache es uno de los mayores éxitos del software libre y su aceptación entre los servidores web es tan grande que ha llegado hasta el punto de llegar a ser un serio competidor del servidor de web de Microsoft (IIS, *Internet information server*). Desde 1996, Apache es el servidor web más popular de Internet, hasta llegar a la actual cota de un 68% de los servidores web frente un 31% sobre IIS (*Fuente: http://news.netcraft.com*). Su desarrollo es continuo y su portabilidad le ha llevado a plataformas como Windows NT/2000/XP y Windows 95/98/Me, a los sistemas Unix y a plataformas como MacOS.

Una de las principales características de Apache es su extensibilidad basada en una gran capacidad de modulación de su código fuente, hecho que ha facilitado la aparición de módulos de extensión como PHP, que evitará el uso de cgi-bin por completo, facilitando así enormemente la programación de aplicaciones en el lado del servidor, en especial en el campo del acceso a bases de datos, así como su agilidad en servir las páginas solicitadas y su seguridad.

1.2. ¿Qué es PHP?

PHP corresponde a las iniciales de *personal home page tools* (herramientas para páginas iniciales personales). Es un lenguaje de programación tipo *script* para entornos web con unas funciones muy semejantes a las de ASP y JSP, utilizado, sobre todo, en servidores Linux para personalizar la información enviada a los usuarios que acceden a un sitio web. Desde un punto de vista técnico, es un lenguaje interpretado de alto nivel, similar en construcciones léxicas y sintácticas a C, C++, Java y Perl, por lo que a quienes ya conozcan estos lenguajes les resultará muy fácil comenzar a escribir código PHP.

El nuevo PHP

El nuevo PHP, cuya versión es la 5, se ha ampliado, respecto la versión 3, en aspectos tan importantes como conceptos de programación orientada a objetos, y su sintaxis es ahora mucho más cercana a la de C, por lo que cualquier programador que haya programado en C no tardará mucho tiempo en aprender a utilizar el lenguaje. Las construcciones sintácticas de PHP son más cercanas a Perl que a C, ya que en su diseño se buscó un lenguaje útil con el que la programación fuese rápida, es decir, que fuese un lenguaje muy productivo al más puro estilo de Perl.

PHP es un lenguaje incrustado (*embedded*) en páginas HTML, es decir, es un lenguaje de programación que se introduce dentro de las páginas HTML. El código PHP se interpreta en el lado del servidor de web, desde donde se genera la página HTML solicitada antes de llevar a cabo su transmisión al navegador. De esta forma, podemos programar aplicaciones asociadas al servidor de web, aumentando, así, la funcionalidad de dicho servidor y convirtiéndolo en un

sistema de desarrollo de aplicaciones cliente/servidor mucho más completo. Su principal objetivo es hacer que desarrolladores de aplicaciones basadas en la web puedan escribir páginas que se generan dinámicamente de una forma sencilla y rápida.

En cuanto a la tecnología del intérprete de PHP, la versión 3 ya era tan rápida como los intérpretes existentes de ASP. Con la versión 4 de PHP, su rendimiento y prestaciones mejoraron todavía más: el intérprete (Zend) era hasta 12 veces más rápido que el de la versión 3; se modularizó todo el diseño interno; se perfeccionó su integración con otros servidores HTTP como el IIS de Microsoft, y se encaró hacia la programación orientada a objetos (Programación OO). Con la versión 5, se ha rediseñado completamente el motor Zend, para crear un lenguaje completamente OO, agilizando más aún su funcionamiento, y extrayendo la compatibilidad con MySQL en un módulo externo (por cuestiones de licencia con MySQL, este SGBD ha dejado de ser "la base de datos" de PHP, para ser una más de las que PHP puede tratar).

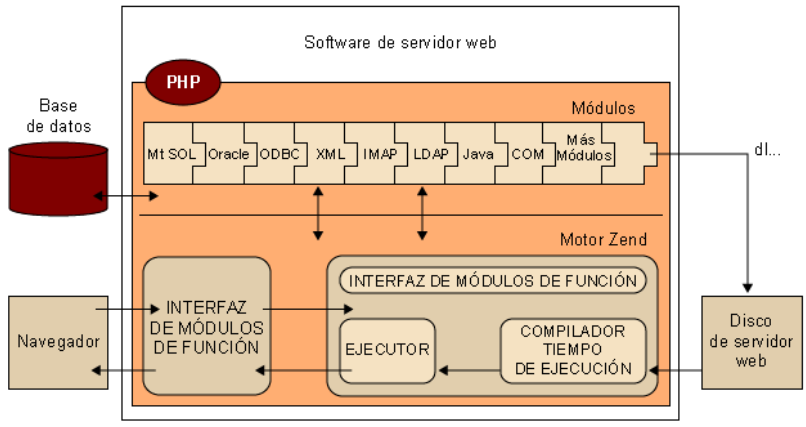
¿Qué es Zend? y ¿qué es PHP?

El nombre Zend se refiere al motor del lenguaje, es decir, el núcleo de PHP.

El término PHP se refiere al sistema completo tal y como aparece desde fuera.

Zend ocupa la parte de **intérprete** (analiza el código de entrada de un *script*, lo traduce y lo ejecuta), y también un poco de la parte de **funcionalidad** (implementa la funcionalidad del sistema). PHP ocupa la parte de funcionalidad y la de **interfaz** (habla con el servidor web, etc.). Juntos forman el paquete completo PHP.

Zend forma realmente el núcleo del lenguaje, mientras que PHP contiene todos los módulos externos (los cuales se pueden cargar en tiempo de ejecución) e incorporados (los que se compilan directamente con PHP) que crean las posibilidades destacadas del lenguaje.



Estructura interna de PHP

PHP proporciona, por tanto, una gran facilidad para acceder a diferentes tipos de bases de datos como Oracle, Sybase, MySQL, PostgreSQL, Adabas, etc. De hecho, es bastante sencillo portar una aplicación escrita con PHP para MySQL a cualquier otro servidor de base de datos, ya que las funciones de acceso que ofrece PHP son, en muchos casos, de sintaxis compartida.

1.3. ¿Qué es MySQL?

MySQL es un sistema de gestión de bases de datos (SGBD) SQL que en algunos aspectos es aproximadamente tan potente como Oracle (http://www.oracle.com/). Cabe mencionar que a mediados del año 2009, Oracle, ha adquirido MySQL.

Sus principales objetivos han sido la velocidad y la robustez. Es un SGBD sencillo y rápido que se adapta perfectamente a entornos en los que el volumen de datos sea del orden de megabytes (en la documentación se habla de su uso con bases de datos de 50 millones de registros). En la versión 5 de MySQL ha incluido el control de transacciones, procedimientos almacenados y *triggers*, por lo que ha rellenado el gran hueco que lo diferenciaba de grandes SGBD como Oracle. Si bien existe la posibilidad de comprar su soporte, Oracle sigue teniendo más aceptación en el mundo empresarial.

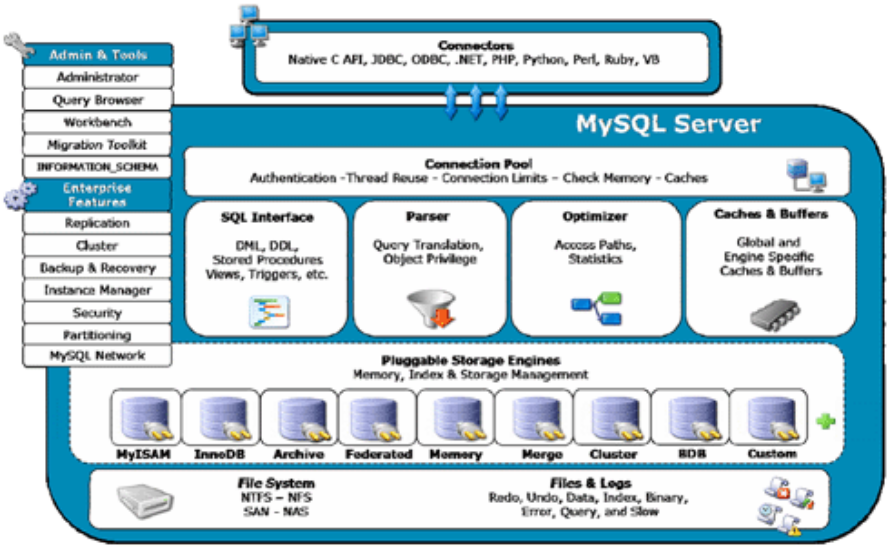
En el último *benchmark* realizado de SGBD, se ha visto un gran crecimiento en la potencia de MySQL, frente a sus competidores.

Sistema Operativo	SGBD	Lenguaje	Operaciones por minuto
Linux	MySQL 5.0	PHP 5	3664
Linux	Pico Lisp 2.2.1	Pico Lisp 2.2.1	2600
Linux	MonetDB 4.9	Java	1833
Windows	MySQL 5.0	Java	1798
Windows	MySQL 4.1	Java	1564
Windows	MySQL 4.1	PHP 5	1542
Linux	DB2 Express 8.2	Java	1537
Linux	Oracle 10g Express	Java	1412
Linux	Sieben Geisslein	Java	600
Linux	MySQL 5.0	Java	587
Windows	MySQL 5.0	PHP 5	444
Windows	MySQL 5.0	Python 2.4	137
Linux	PostgresSQL 8.1	PHP 5	120

En estos últimos años destacadas compañías de software han desarrollado aplicaciones SQL de uso libre y con código fuente (*open source*). En el mundo de GNU/Linux MySQL es, junto a Postgres (http://www.postgresql.org/), uno de los SGBD más populares.

MySQL también puede verse como un conjunto de aplicaciones o *pluggins* funcionando en conjunto.

Existen en su versión actual distintos motores de almacenamiento de datos, entre los que destacan MyISAM (permite índices por cadenas completas) y InnoDB (que permite el uso de transacciones) o la incorporación de *buffers* en memoria que permiten agilizar la respuesta de sus resultados.



MySQL se encuentra, igual que PHP, en fase de pleno desarrollo; se están publicando con regularidad nuevas versiones del sistema, así como herramientas que son básicas en cualquier SGBD actual:

- Dispositivo JDBC para acceder desde Java.
- Dispositivo ODBC para acceder utilizando la API ODBC.
- API de programación para C, Perl, C++, Python y TCL.
- Acceso desde PHP.
- Entornos visuales de gestión de la base de datos.
- Control de acceso basado en una base de datos de administración.

1.4. El funcionamiento del sistema

Como una motivación previa y para que el lector pueda tener una visión global desde el principio, presentamos en este apartado una visión global del sistema. En el siguiente gráfico aparecerán destacadas las partes fundamentales del mismo.

Los pasos que describen la interacción entre el usuario y la base de datos son:

1) El usuario carga una página HTML con un formulario, rellena los datos y se los envía al servidor web.

2) Por medio de la red TCP/IP los datos llegan a Apache.

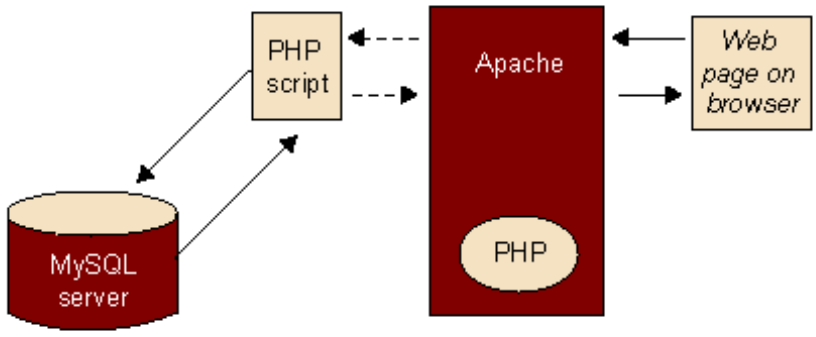
3) El servidor detecta que el usuario solicita una página PHP, por lo que informa al módulo de PHP del programa que hay que ejecutar y le pasa los datos del formulario.

4) El módulo de PHP ejecuta el programa, el cual accede a MySQL utilizando, de nuevo, una comunicación TCP/IP.

5) MySQL procesa la petición del programa PHP y le envía los resultados de vuelta.

6) El módulo PHP recibe los resultados del servidor de base de datos, les da formato en una nueva página HTML y se los devuelve al cliente mediante el servidor Apache.

7) El cliente recibe la página HTML resultado de su petición por medio de la red TCP/IP.



En este primer esquema ya podemos ver que la interacción con la base de datos se hace, en su totalidad, por medio de PHP.

2. Puesta en marcha de un SGBD para Windows (WAMP)

WAMP es el acrónimo para instalaciones Apache, MySQL, PHP/Perl/Python sobre la plataforma Windows que permite la publicación de páginas dinámicas sobre la Web.

WampServer facilita la instalación de los módulos descritos anteriormente, ya que consiste en una sola aplicación que instalar que contiene los servicios para generar un servidor WAMP. Por su simplicidad de instalación, WampServer es una herramienta de gran utilidad a la hora de desarrollar páginas web dinámicas.

Existen en el mercado otras soluciones integradas para la creación de servicios web dinámicos, basados igualmente en MySQL, Apache y PHP/Perl/Python. Una solución bastante distribuida es XAMPP, y existen versiones para Windows, Linux, Mac y Solaris.

A continuación describiremos el proceso de instalación del paquete WampServer en la plataforma Windows XP y su posterior configuración. Seguidamente, presentamos algunos puntos que se deben considerar para instalaciones realizadas sobre Windows 7. Finalmente, daremos una descripción básica sobre los ficheros de configuración de Apache, PHP y MySQL que es conveniente saber para entender cómo solventar algunos problemas o limitaciones que un desarrollador puede encontrarse durante la creación de un portal web dinámico.

2.1. Instalación de WampServer

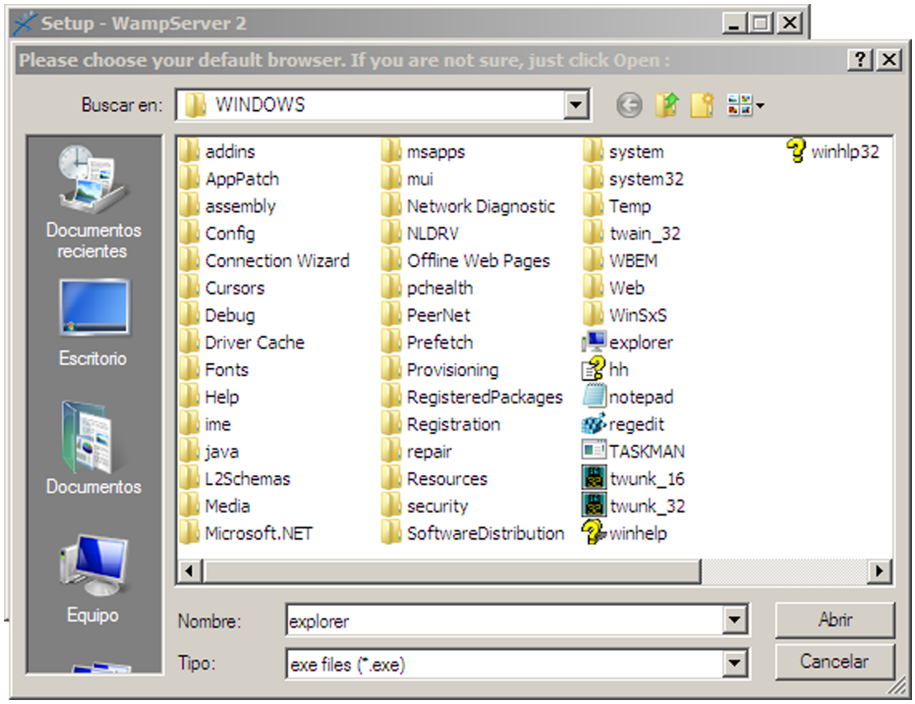
El proceso de instalación del servicio WAMP con WampServer es muy sencillo.

Tras la ejecución del instalador con nombre WampServer2.0i.exe aparece la pantalla de bienvenida al inicio de la instalación. Deberemos simplemente presionar sobre el botón “Siguiente” para empezar con el proceso de instalación.

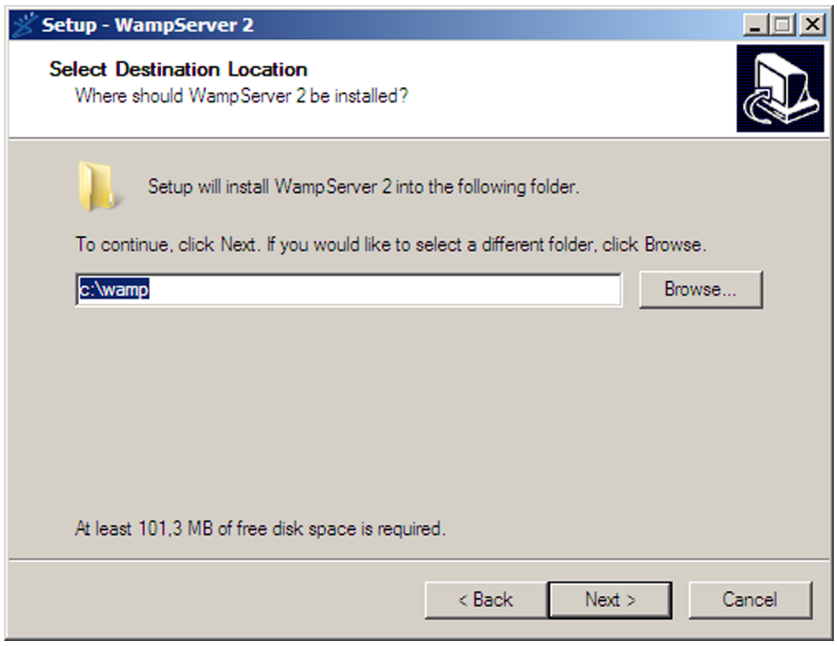
Descargarse la versión más reciente

De la página web oficial del producto WampServer puede descargarse la versión más reciente del servidor. Las capturas de pantalla que se ofrecen en este capítulo se han realizado sobre la instalación del producto WampServer 2.0i con fecha de lanzamiento 11 de julio del 2009.

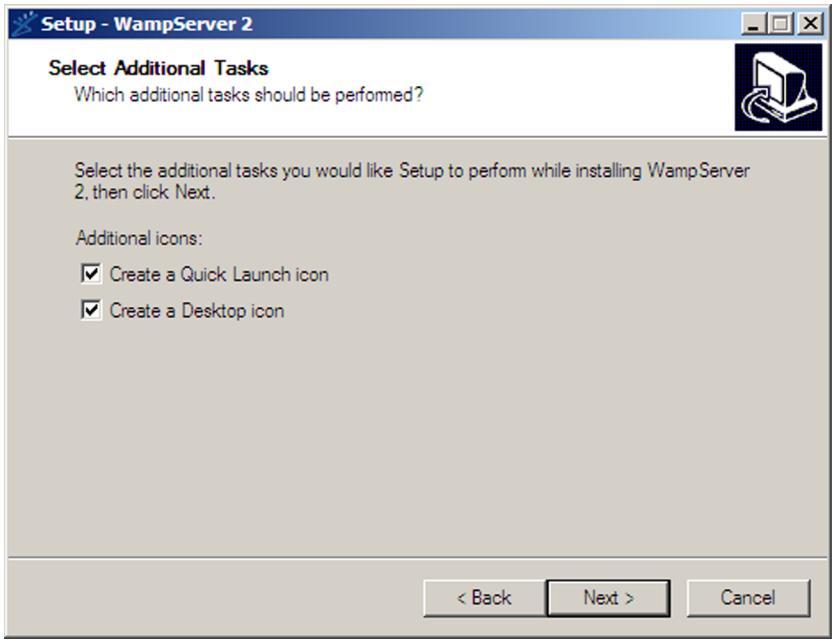
Algunos de los enlaces de WampServer, como por ejemplo para acceder a phpMyAdmin para la gestión de las bases de datos, se realizan a través del navegador web. El instalador nos preguntará en este punto cuál es el navegador web que desearemos usar. En la captura siguiente se muestra cómo se ha elegido el navegador Explorer.exe, es decir, el Internet Explorer existente en la plataforma Windows.

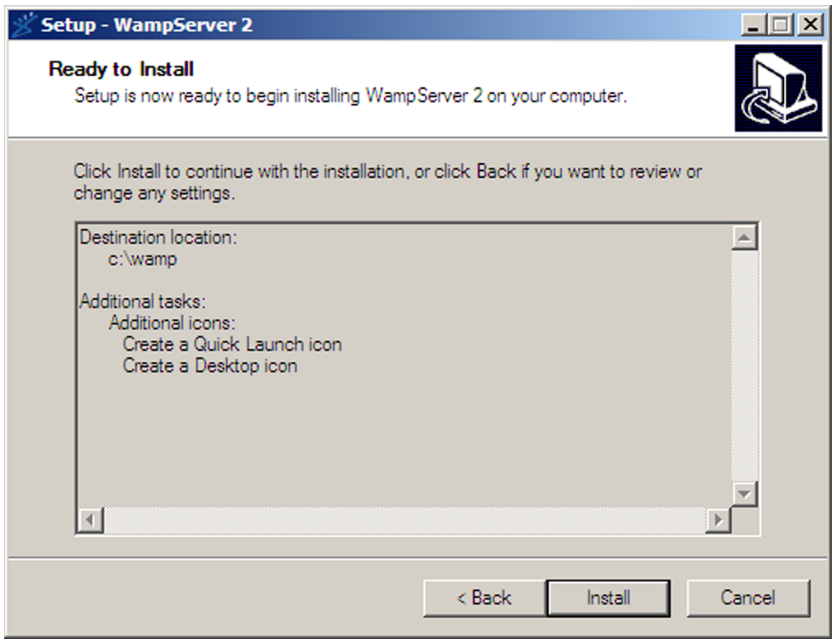


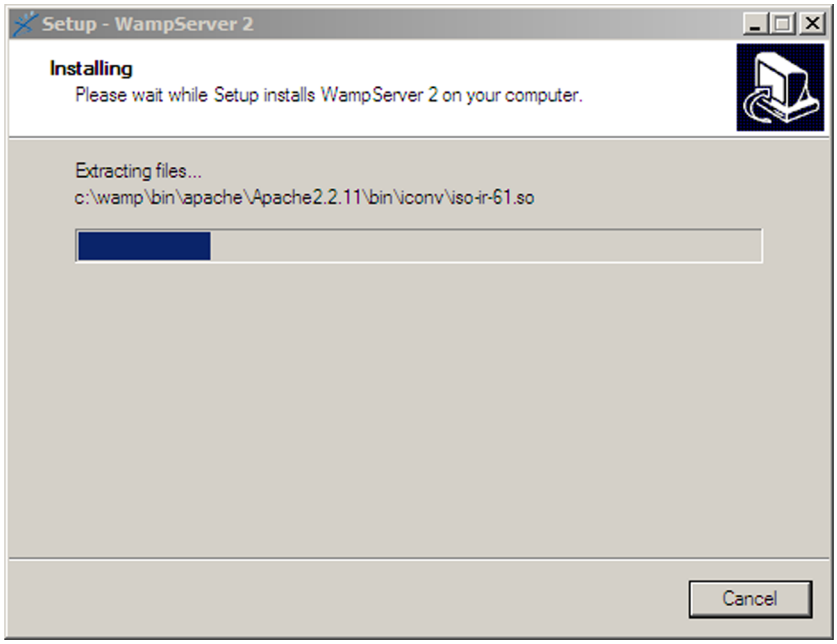
Después de aceptar la licencia GNU para la instalación del software WampServer, se solicita la dirección donde instalar la aplicación. Por defecto, la instalación se realizará en la carpeta C:\wamp.



Una vez entrada la ruta de la instalación, presionaremos en el botón "Siguiente" e "Instalar". Con ello tendremos la aplicación WAMP instalada en nuestro servidor local, pero será necesario un proceso de configuración del servidor web dinámico, muy sencillo también.

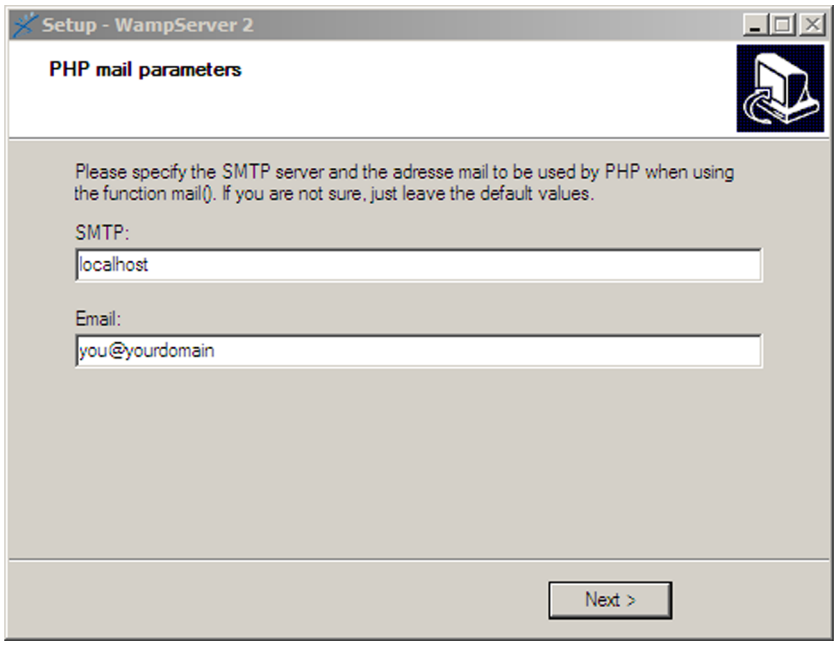






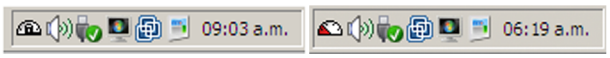
El servidor web WampServer, mediante librerías PHP, permite el envío de correos electrónicos. Un ejemplo cotidiano de envío de correos electrónicos es para el alta a servicios web, tales como registro en foros. El primer paso de configuración del servidor web dinámico es la entrada de los parámetros para usar un servidor de envío de correos electrónicos mediante protocolo SMTP. En caso de disponer de un servidor de correos, entraremos la dirección del servidor de correo electrónico y la dirección de correo electrónico desde donde se mandarán los correos.

En caso de no disponer de servidor de correo electrónico, simplemente presionaremos en el botón “Siguiente”.

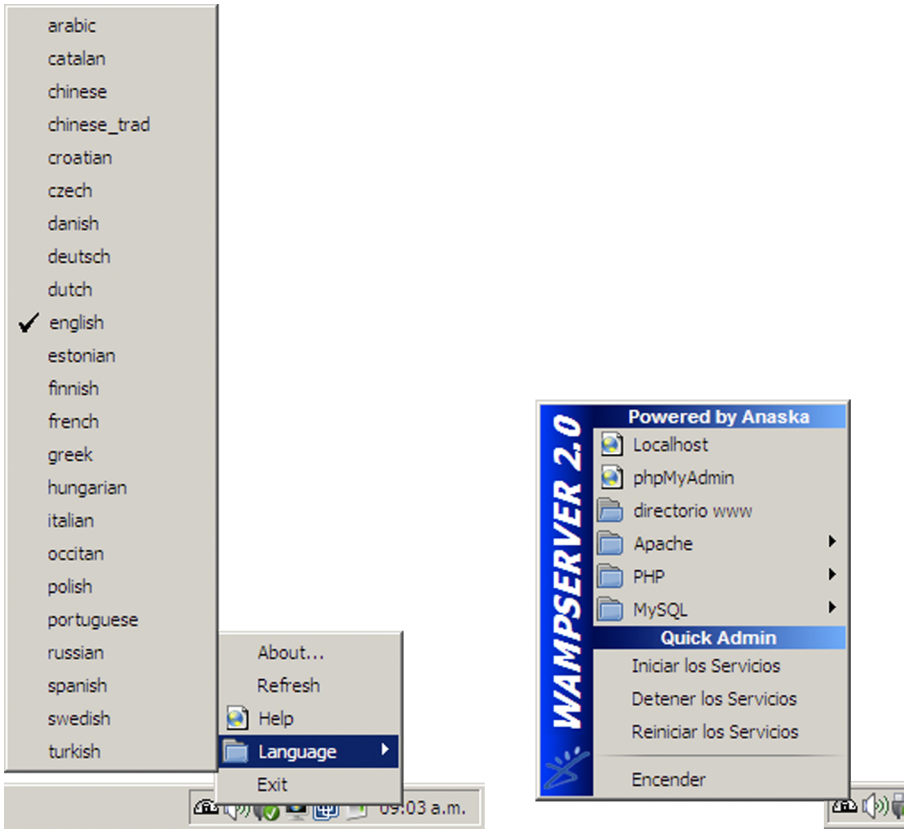


Para finalizar, comprobaremos que la casilla "Iniciar WampServer 2 ahora" está activada y presionaremos en el botón "Finalizar" para finalizar con el proceso de configuración del servidor web dinámico.

Observando la barra de tareas de Windows, podemos detectar que WampServer está en ejecución (el icono en blanco mostrará que el servicio está encendido, mientras que el icono en rojo determina que está apagado). Pinchando sobre el icono de WampServer podemos modificar ciertos parámetros de configuración, como el idioma de los menús de configuración, encender o apagar los servicios web o de bases de datos, activar módulos de Apache, activar extensiones para PHP, etc.

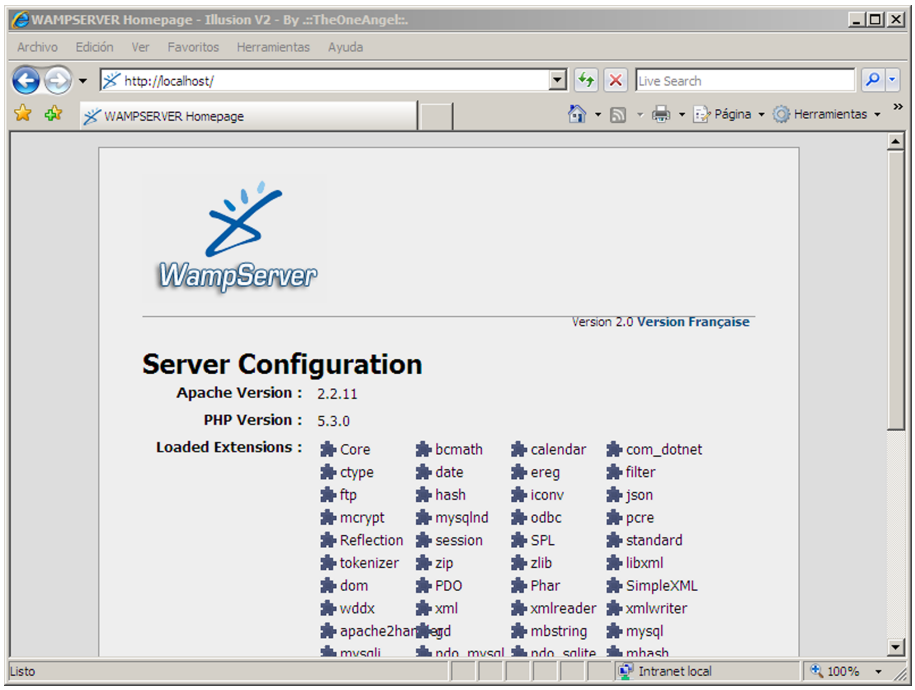


2.2. Primeros pasos con WampServer



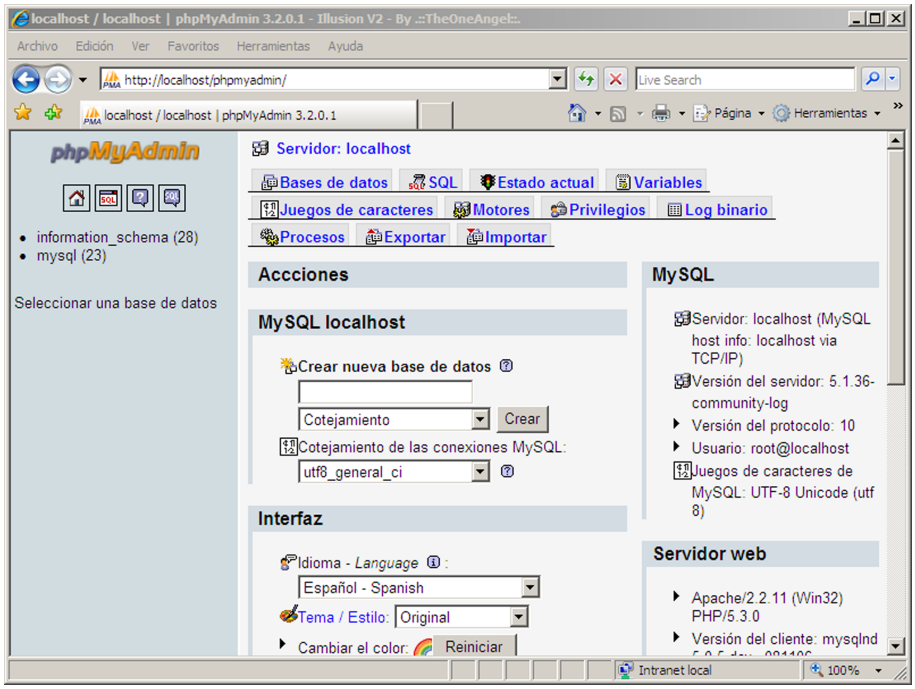
Una vez terminada la instalación y configuración del servidor web dinámico con WampServer, comprobaremos que su ejecución es correcta. Por defecto, cuando llamamos a la raíz del árbol web, se nos mostrará una pantalla de bienvenida con la configuración actual que está usando el servidor web, así como las extensiones de Apache que tiene activadas en dicha ejecución. Encontraremos también un enlace a phpinfo() para ver los módulos activados en el servidor web dinámico a través de PHP.

Para poder ver esta pantalla de bienvenida, abriremos un navegador y escribiremos la dirección “http://localhost”, o pincharemos sobre el enlace “Localhost” que se muestra en el menú de WampServer.

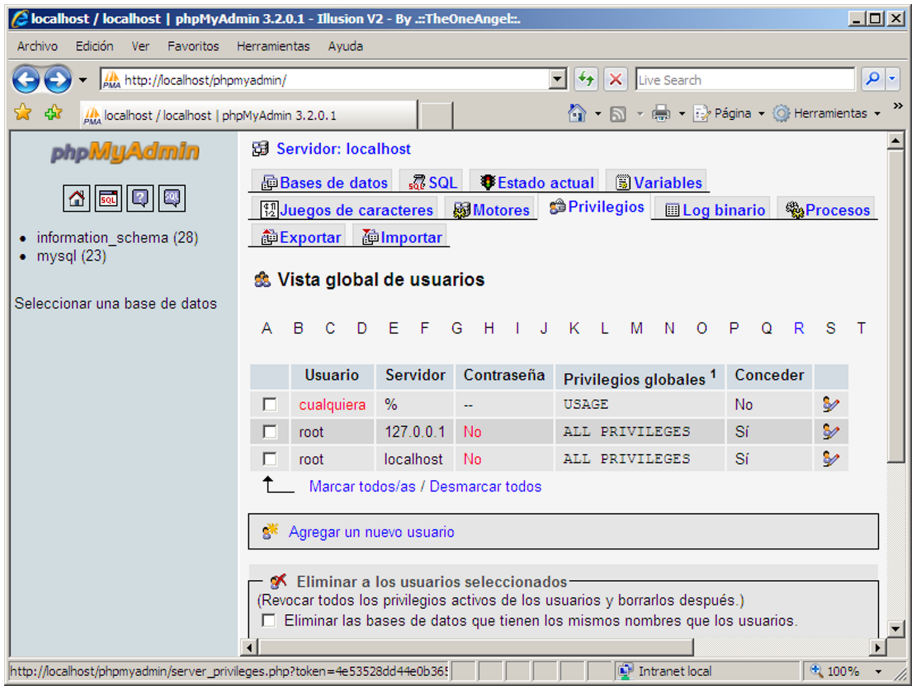


La gestión de las bases de datos podremos realizarla con paquetes gráficos como MySQL GUI Tools, que podemos descargar de MySQL.

Por otro lado, WampServer contiene la herramienta web phpMyAdmin, que está disponible a través de phpMyAdmin, o por medio de uno de los enlaces ofrecidos en la página de bienvenida de WampServer.



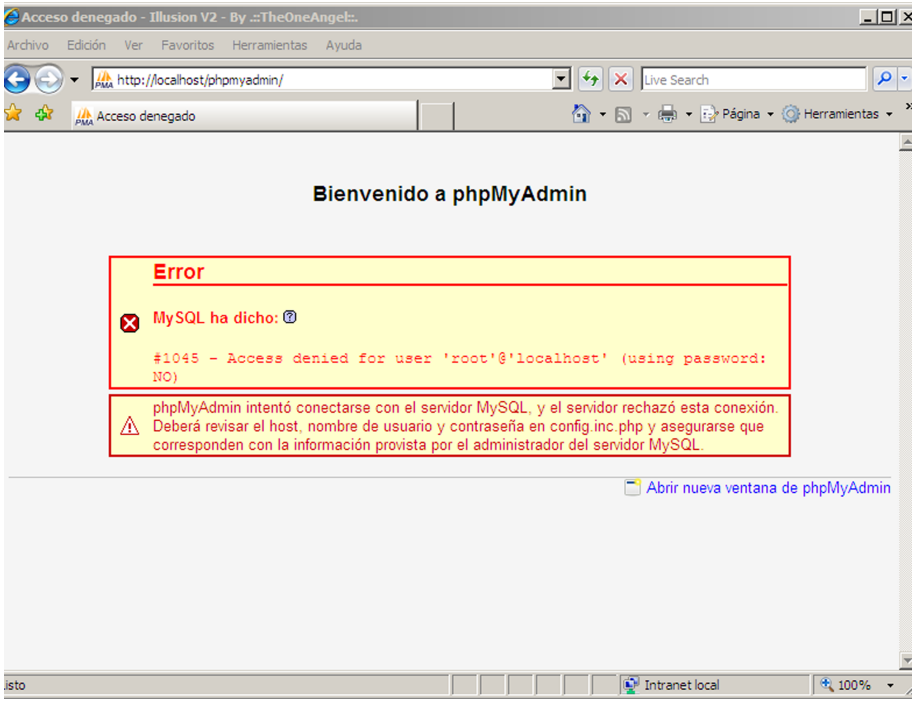
Como podremos comprobar, por defecto el usuario administrador (root) de MySQL está configurado sin contraseña, por lo que la seguridad de acceso a MySQL no es la recomendada. De todos modos, modificar la clave de acceso del administrador a MySQL no es complicado. Para hacerlo, realizaremos los siguientes pasos:



1) Una vez dentro de phpMyAdmin, entraremos en la sección de "Privilegios" e editaremos el usuario root del servidor 127.0.0.1 pinchando sobre el botón de edición .

2) En la sección "Cambio de contraseña", estableceremos la clave de acceso que queremos usar para el usuario *root* para las conexiones a las bases de datos y presionaremos sobre el botón "Continuar" para almacenar los cambios. Desde los Estudios de Graduado Multimedia recomendamos que se establezca como clave "BBDDGM", acrónimo de "Bases de Datos–Graduado Multimedia".

3) Repetiremos el proceso para cambiar la clave del usuario root en localhost, estableciendo la misma contraseña usada en el usuario root en 127.0.0.1.



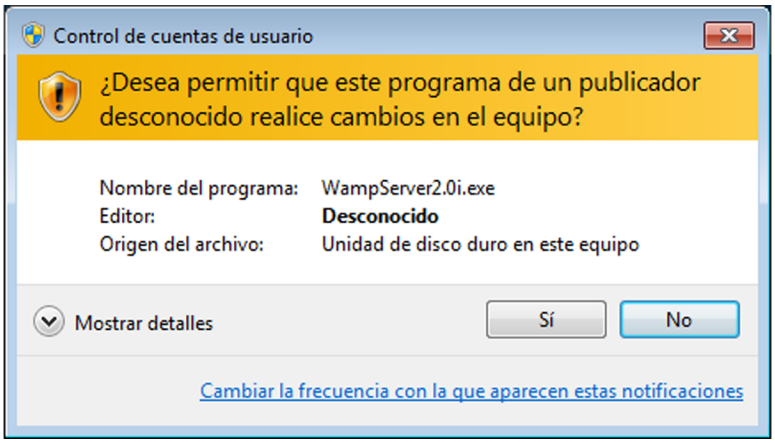
4) Posterior a estos cambios, phpMyAdmin no podrá acceder a las bases de datos para su gestión, dado que está configurado para usar el usuario root sin contraseña. Para permitir a phpMyAdmin que pueda volver a gestionar las bases de datos existentes en MySQL, deberemos modificar el fichero config.inc.php existente dentro de la carpeta C:\wamp\apps\phpmyadmin3.2.0.1 para que la contraseña corresponda con la que hayamos establecido. Hay que tener en cuenta que el directorio "wamp" podrá cambiar en caso de haber realizado la instalación en otro directorio.

```
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = 'BBDDGM';
```

Una vez actualizado el fichero de configuración de phpMyAdmin, el acceso a phpMyAdmin volverá a ser funcional, por lo que sólo será necesario refrescar el navegador web para acceder a http://localhost/phpmyadmin.

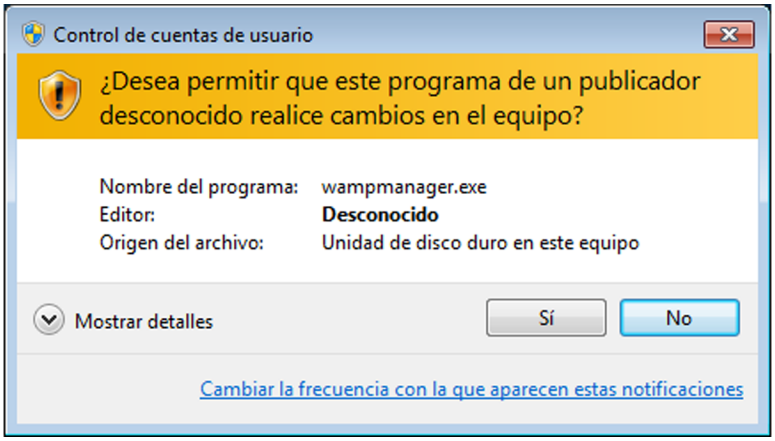
2.3. Consideraciones sobre la instalación sobre Windows 7

El proceso de instalación en Windows 7 sólo se diferencia al de la plataforma XP en lo que a controles de seguridad se refiere. Al ejecutar el instalador, Windows nos prevendrá de que el ejecutable quiere modificar el registro de Windows.

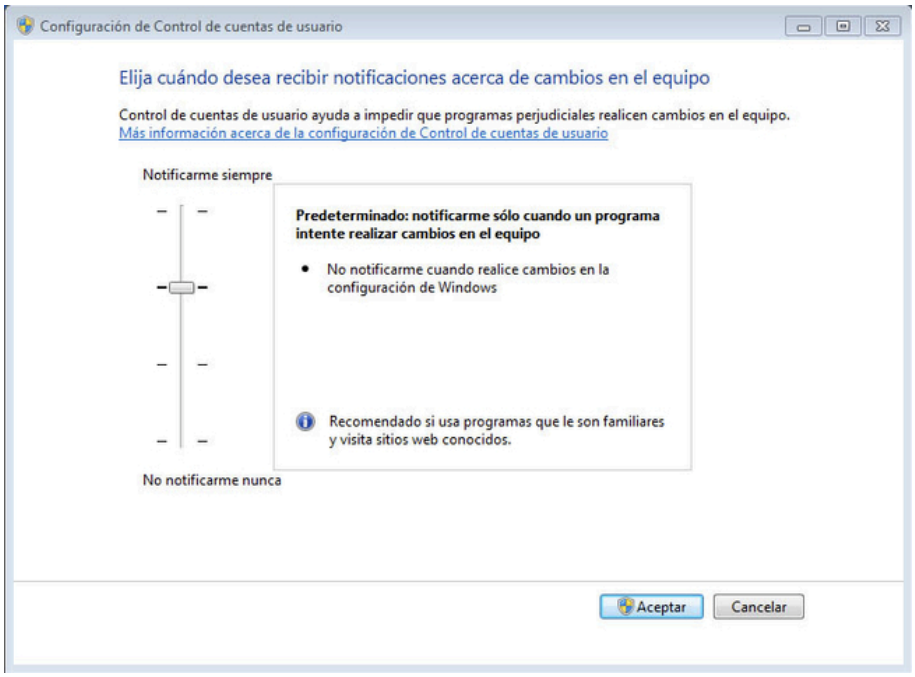


Para poder realizar la instalación, deberemos aceptar que Wamp abra puertos para que nuestros servicios puedan ser accedidos desde otros ordenadores. Por ello, Windows 7 nos notifica sobre el riesgo de que una aplicación esté permitiendo conexiones remotas para acceder a recursos de nuestro ordenador.

Wampmanager.exe es el fichero binario que Windows detecta como posible amenaza de seguridad para nuestro ordenador. Deberemos aceptar la ejecución del servicio presionando el botón "Sí".



Hasta este punto, dada la configuración de seguridad de Windows 7, sería necesario aceptar el mensaje de aviso de amenaza de seguridad cada vez que queramos activar los servicios ofrecidos por WampServer. En Windows 7, es posible regular el nivel de avisos del sistema operativo. Para evitarlo, podemos bajar el nivel de seguridad al mínimo en la Configuración de Control de cuentas de usuario. Dicha modificación puede realizarse desde el Menú de Inicio → Panel de control → Cuentas de usuario → Cambiar configuración de Control de cuentas de usuario.



2.4. Consideraciones sobre los ficheros de configuración

El uso de los servicios Apache, PHP y MySQL mediante paquetes como WampServer provoca que la instalación tenga gran sencillez, pero, por otro lado, pasan a ignorarse muchos de los factores que pueden alterarse para la configuración de los servicios.

A continuación presentamos algunos parámetros de los ficheros de configuración que se deben tener en cuenta durante el desarrollo de portales web dinámicos.

2.4.1. Fichero de configuración de Apache

El fichero de configuración de Apache (httpd.conf) contiene los parámetros para el inicio del servicio Apache.

Como ya hemos mencionado, Apache es el servicio que permite publicar sitios web en nuestro servidor. Para que un sitio web publicado en Apache pueda consultarse desde otros ordenadores, es necesario que el servicio escuche de un puerto para atender a peticiones HTTP que recibe. Las peticiones http, que son las realizadas a servidores web, suelen enlazarse al puerto 80. Por defecto, Apache está configurado para arrancar el servicio http en el puerto 80. Existen otros servicios que se asocian también al puerto 80, tales como el servicio web IIS o Skype. Por ello, en ocasiones es una buena solución enlazar el servicio web Apache a un puerto alternativo, comúnmente el 8080. El parámetro Listen está

Ejemplo

Por ejemplo:
http://localhost:8080/info.php
en caso de que el puerto por defecto se haya cambiado por el puerto 8080.

orientado a este efecto. Deberemos tener en cuenta que en caso de cambiar el puerto de Apache, las peticiones que hagamos en el navegador web deberán indicar el puerto al que deben lanzarse.

```
Listen 127.0.0.0:8080
```

En ocasiones, pueden darse errores en el servidor web o éste puede incluso dejar de funcionar. En estos casos, suele ofrecerse una dirección de correo electrónico para contactar con el administrador del portal web. En el fichero httpd.conf existe un parámetro para definir dicha dirección de correo electrónico:

```
ServerAdmin admin@localhost
```

Los ficheros publicados en un servidor web se encuentran a partir de una carpeta inicial, llamada Raíz del servidor web. La carpeta raíz también es un parámetro configurable dentro de Apache. Por defecto, durante la instalación de Apache, se establece c:/wamp/www como carpeta para la ubicación de los documentos web, pero se puede cambiar con la siguiente directiva:

```
DocumentRoot “c:/wamp/www/”
```

Normalmente, las peticiones web solicitan un documento en concreto, como por ejemplo http://servidor.web.edu/servicio1/documento1.php.

Pero no siempre se solicita específicamente un documento de la web, como por ejemplo en http://servidor.web.edu/servicio1/. En dicho caso, el usuario quiere obtener el documento de índice que se encuentra en el directorio “servicio1”. Para indicar qué nombre y extensión pueden tener los posibles ficheros de índice y su prioridad, se utiliza la directiva DirectoryIndex.

En el siguiente ejemplo, primero se comprobará la existencia del fichero http://servidor.web.edu/servicio1/index.php, posteriormente index.php3, y así sucesivamente.

```
<IfModule dir_module>
       DirectoryIndex index.php index.php3 index.html index.htm
</IfModule>
```

Durante el desarrollo de sitios web, a menudo podemos encontrarnos con que el servidor no responde a las peticiones de la manera esperada. Los registros del servidor pueden servir de mucha utilidad para detectar los motivos por los que un servicio web actúa incorrectamente. La directiva LogLevel nos permite establecer el nivel de *log* que queremos registrar.

```
LogLevel warn
```

Las líneas de configuración que empiezan con el comando AddType detallan a Apache cómo debe interpretar diferentes tipos de ficheros a partir de su extensión. Por ejemplo, ficheros con extensión .tgz deberán ser abiertos por descompresores GZip, o ficheros con extensión .php, deberán ser interpretados por la aplicación httpd-php o, en otras palabras, el motor para el lenguaje PHP.

```
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
AddType application/x-httpd-php .php
```

Dada la cantidad de parámetros y opciones que pueden configurarse en Apache, existe también la posibilidad de modularizar los ficheros de configuración, creando un fichero de configuración principal httpd.conf, y varios ficheros de configuración complementarios. Los ficheros complementarios deben ser declarados en el fichero de configuración principal (httpd.conf) para que, al arrancar el servicio, sus parámetros sean incluidos. Esto se realiza mediante la directiva Include.

```
Include conf/extra/httpd-ssl.conf
```

2.4.2. Fichero de configuración de PHP (PHP.ini)

Dentro del fichero de configuración php.ini se describen, entre otras cosas, las extensiones de PHP que se deben activar, las rutas para la subida de ficheros mediante protocolo POST, los tiempos de ejecución de ficheros PHP y el uso de la memoria RAM.

Los documentos con extensión PHP pueden combinar código HTML con código PHP. El código PHP debe diferenciarse del código HTML, ya que el código PHP debe ser interpretado. Normalmente, el código PHP se delimita mediante las llaves <?php y ?>, pero es posible hacerlo también mediante las llaves <? y ?>. Por defecto, las llaves cortas (short_open_tag) no están habilitadas en WampServer, por lo que deberá modificarse el valor del parámetro short_open_tag (existe la opción de activar las claves cortas mediante el menú de WampServer).

```
short_open_tag = Off
```

Los documentos ASP usan las llaves <% y %> para identificar el código que debe ser interpretado. Si deseamos que nuestras aplicaciones PHP usen las llaves <% y %>, deberemos activar el parámetro asp_tags.

```
asp_tags = On
```

La complejidad de documento PHP puede ser elevada, lo que en algunos casos puede implicar tiempos de ejecución largos y gran cantidad de recursos consumidos. El tiempo de ejecución de un fichero PHP, así como la memoria que su ejecución requiere también son parámetros que pueden definirse dentro del fichero php.ini.

```
max_execution_time = 30
max_input_time = 60
memory_limit = 128M
```

Del mismo modo, puede declararse el tamaño máximo permitido para la subida de ficheros.

```
post_max_size = 8M
```

Así como hemos visto con Apache, PHP también genera trazas de log que son de gran ayuda para la interpretación de errores durante el desarrollo de aplicaciones dinámicas en PHP. En PHP, el nivel de registro de errores puede definirse de la siguiente manera:

```
error_reporting = E_ALL
```

Dentro del propio fichero php.ini se describe información sobre el efecto de cada uno de los parámetros que contiene, por lo que recomendamos realizar al menos una lectura del documento de configuración para familiarizarse con las posibilidades existentes.

2.4.3. Fichero de configuración de MySQL

La configuración de sistema gestor de bases de datos MySQL vía la edición de su fichero de configuración (MY.ini o my.cnf en instalaciones independientes de MySQL) no suele hacerse con tanta frecuencia como en Apache y PHP. No obstante, contiene parámetros de configuración que pueden ser de gran interés.

Normalmente, las comunicaciones hacia MySQL se realizan por protocolo TCP/IP sobre el puerto 3306. Éste es el puerto por defecto en el que MySQL se registra para aceptar peticiones SQL. Este puerto es configurable y puede ser cambiado por un puerto alternativo. El siguiente comando permite cambiar el puerto de MySQL por defecto por el 3333.

```
port = 3333
```

3. Puesta en marcha de un SGBD para Ubuntu GNU/Linux

Linux es un término genérico para referirse a sistemas operativos similares a Unix basados en el núcleo de Linux. Las variantes de estos sistemas se denominan distribuciones de Linux y cada distribución está enfocada para satisfacer las necesidades de un grupo específico de usuarios.

El objetivo de la denominación GNU/Linux (GNU es un acrónimo recursivo que significa '*GNU is Not Unix*') es crear un sistema operativo completamente libre. La base de cada distribución incluye el núcleo Linux, con las bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software.

Ubuntu es una distribución GNU/Linux que ofrece un sistema operativo enfocado a computadores personales con soporte para servidores. Está basado en Debian GNU/Linux (distribución Linux sin ánimo de lucro más popular del mundo) y soporta, principalmente, dos arquitecturas de hardware: Intel x86 y AMD64. Esta distribución ha sido traducida a numerosos idiomas y concentra sus objetivos en la facilidad y libertad de uso. Sus versiones estables se liberan cada 6 meses y se mantienen actualizadas en materia de seguridad hasta 18 meses después de su lanzamiento.

Todos los lanzamientos de Ubuntu se proporcionan sin coste alguno con varias alternativas sobre el tipo de instalación.

1) El **CD de escritorio** (comúnmente llamado live CD), permite probar Ubuntu sin hacer cambios en el equipo y agregar una opción para instalarlo más tarde.

2) El **CD de instalación de servidor** instala Ubuntu permanentemente en una computadora usada como servidor (no se instalará una interfaz grafica de usuario). Se pueden descargar desde los sitios oficiales.

Ubuntu posee una gran colección de aplicaciones sencillas de instalar y configurar desde su entorno de escritorio oficial Gnome. UTF-8 es la codificación de caracteres en forma predeterminada desde la versión 5.04. Ubuntu es la distribución GNU/Linux elegida por la UOC porque es un sistema operativo libre con un escritorio potente y amigable.

3.1. Requisitos del sistema para instalar la versión 8.10 Intrepid Ibex

Se recomienda:

- Procesador: 1 GHz
- Memoria RAM: 512 MB (1 GB mínimo)
- Disco duro: 10 GB mínimo

Instalación a partir de live CD (recomendado) o a partir de Windows con Wubi (para inexpertos):

- Procesador: 600 MHz
- Memoria RAM: 256 MB (1 GB mínimo)
- Disco duro: 10 GB mínimo

Enlaces recomendados

Enlaces externos:

Sitio web oficial de Ubuntu (en inglés)

Comunidad hispana de Ubuntu (en castellano)

Wubi

Wubi es un instalador de Ubuntu para sistemas operativos Windows que permite a usuarios de Windows, no acostumbrados a Linux, poder probar Ubuntu sin el riesgo de perder información durante un formateo o particionado, además de desinstalar Ubuntu desde Windows.

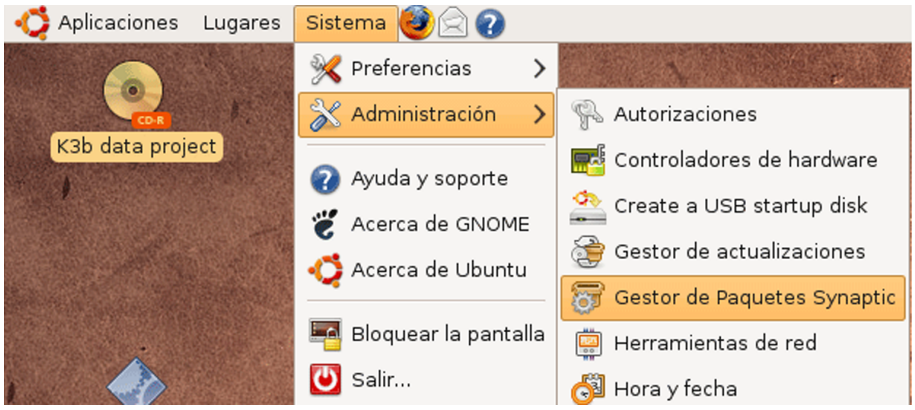
3.2. Instalación del servidor LAMP desde el escritorio de Ubuntu

La arquitectura LAMP se refiere a un conjunto de tecnologías de software necesarias para alcanzar una solución global, en este caso para configurar sitios web o servidores dinámicos. El acrónimo LAMP combina los siguientes programas: **L**inux (sistema operativo), **A**pache (servidor web), **M**ySQL (gestor de bases de datos), **P**erl, **P**HP o **P**ython (lenguajes de programación).

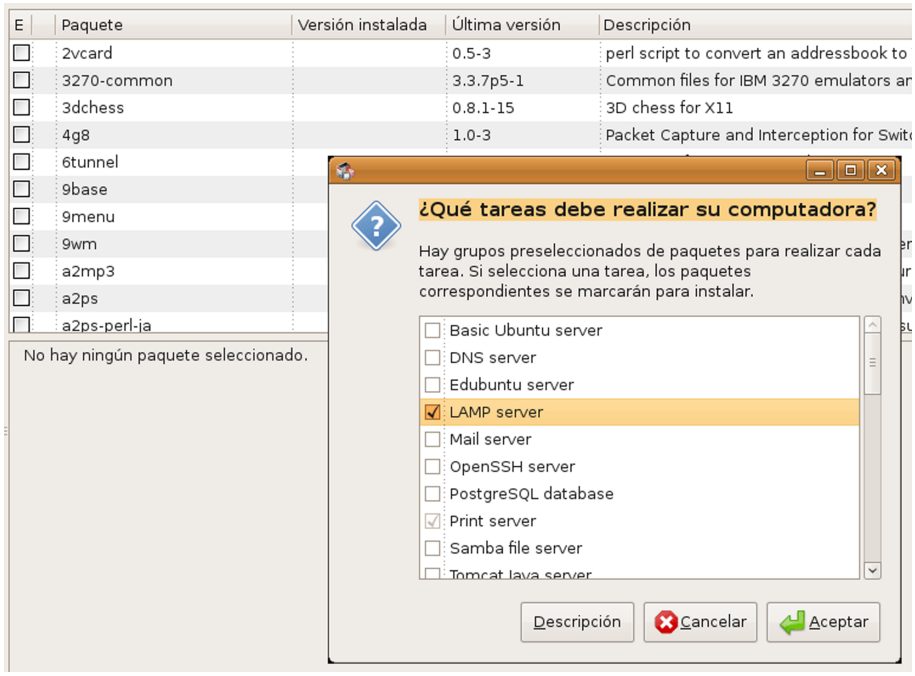
En Ubuntu existen varios modos de añadir, eliminar o actualizar aplicaciones del sistema. Podemos instalar los paquetes desde el CD de instalación o, alternativamente, desde los sitios oficiales de Internet (será necesario tener abierta la conexión a Internet). El gestor de paquetes Synaptic nos permitirá instalar, reinstalar y eliminar paquetes de una manera gráfica muy sencilla.

El proceso de instalación de Apache + Mysql + PHP es rápido y sencillo.

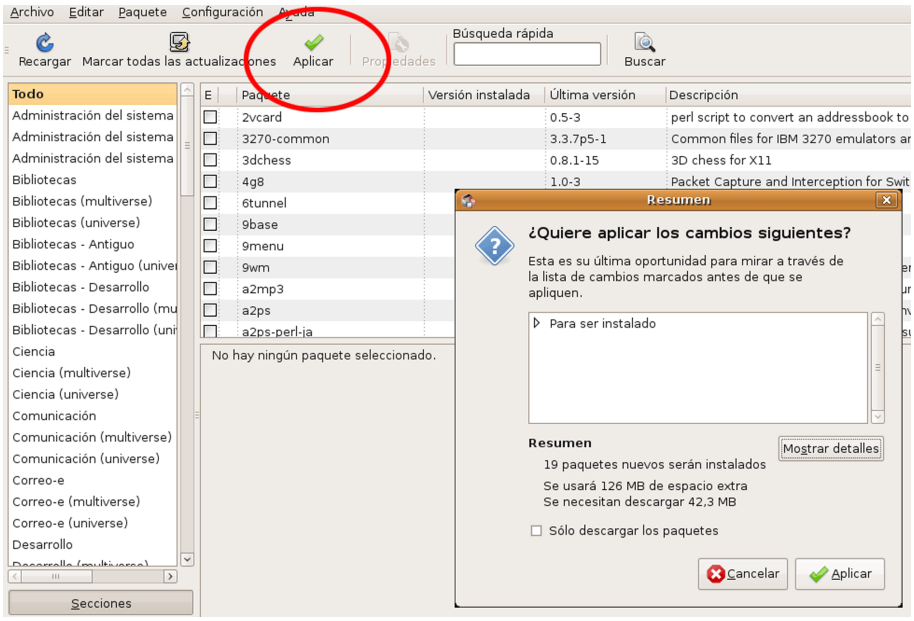
1) Abrimos el Gestor de Paquetes Synaptic desde Menú de Ubuntu: Sistema → Administración → Gestor de Paquetes Synaptic



2) En Gestor de Paquetes Synaptic abrimos el menú "Editar" y ejecutamos la opción "Marcar paquetes por tarea". Se abrirá una ventana que nos pregunta "¿Qué tarea debe realizar su computadora?"; marcamos la casilla "Lamp server".



3) Tras darle al botón "Aceptar", se nos informa de todos los paquetes necesarios para instalar el servidor LAMP. Confirmamos la instalación pulsando el botón "Marcar". Después hacemos clic en el botón "Aplicar" ubicado en el menú para que comience la instalación.



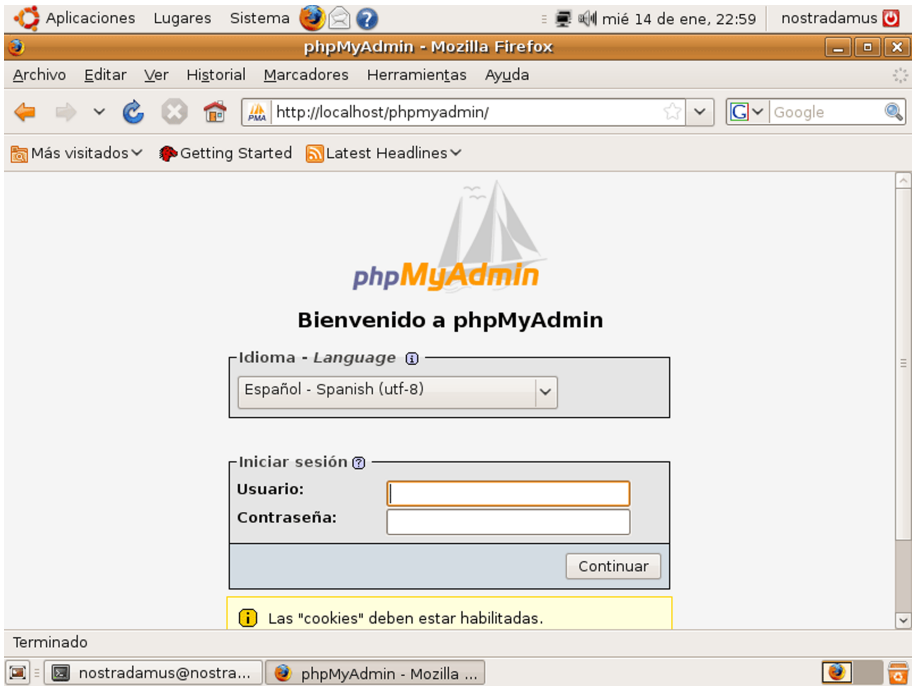
4) Una vez terminado el proceso de descarga de los paquetes comienza la instalación. En la configuración de MySQL 5.0 se nos pregunta qué contraseña queremos asignarle al administrador (root) del servidor de la base de datos. Tras este paso, concluye la instalación del Servidor Lamp con Synaptic.

5) Abrimos Firefox y escribimos http://localhost (alternativamente, http://127.0.0.1) y nos deberá aparecer la página de bienvenida de Apache. Recordad que el directorio raíz del servidor HTTP se encuentra en /var/www, y allí es donde se deberán colocar todos los archivos del sitio web.

También es recomendable instalar con Synaptic el paquete phpmyadmin para administrar MySQL y sus bases de datos por medio de la interfaz web de esta aplicación. Tras marcar el paquete e instalarlo, bastará con escribir en el navegador:

```
http://localhost/phpmyadmin/
```

La página de inicio de phpMyAdmin nos pedirá un nombre de usuario (root) y la contraseña que definimos en su momento al instalar MySQL (paso 4).



3.3. Instalación del servidor LAMP desde la consola de Ubuntu

La consola de Ubuntu (también conocida como intérprete de comandos, terminal o consola) es un programa informático que recibe y procesa órdenes de texto y tiene como función actuar como interfaz entre el usuario y el sistema operativo.

En este apartado vamos a ver cómo instalar el servidor LAMP desde la terminal de Ubuntu. Para labores de administración en terminal, Ubuntu incluye un comando llamado **sudo** con el que se evita el uso del usuario root (administrador de Linux). Esta instalación sirve tanto para Ubuntu como para Debian (los usuarios de Debian deberán identificarse como root y no utilizar el sudo en los comandos).

Los paquetes de software son accesibles a partir de la lista de repositorios ubicada normalmente en /etc/apt/source.list. Si quisiéramos actualizar el índice de paquetes sincronizándolo con las últimas versiones disponibles de cada uno, escribiríamos:

```
> sudo apt-get update
```

A continuación vamos a instalar los paquetes necesarios para montar un servidor LAMP desde la terminal de Ubuntu. Abrimos la terminal (Aplicaciones -> Accesorios -> Terminal).

1) Para instalar Apache 2.0 escribimos el siguiente comando y pulsamos "Enter":

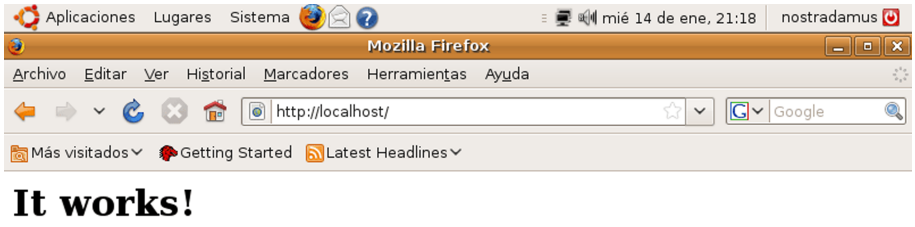
```
> sudo apt-get install apache2
```

La terminal nos pedirá la contraseña de root (del sistema operativo); la introducimos y pulsamos de nuevo "Enter".

Los archivos de configuración de apache están ubicados en /etc/apache2/apache2.conf.

La carpeta raíz de la web se encuentra en /var/www/.

Para comprobar que el servidor web está funcionando, abrimos nuestro navegador web (Firefox en Ubuntu) y escribimos http://localhost/; aparecerá la página de bienvenida de Apache.



2) A continuación instalaremos los paquetes PHP5 y el módulo php5 de Apache 2.0. Abrimos de nuevo la terminal de Ubuntu y escribimos:

```
> sudo apt-get install php5 libapache2-mod-php5
```

Tras la instalación de PHP, reiniciamos Apache con el comando:

```
> sudo /etc/init.d/apache2 restart
```

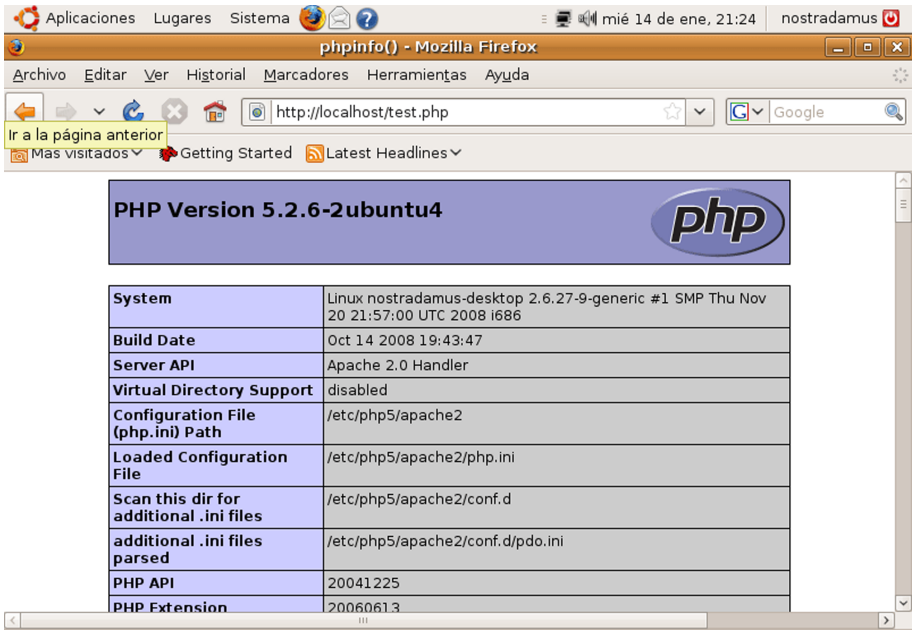
Para comprobar que PHP funciona correctamente como módulo de Apache, creamos un archivo de texto y dentro de ese archivo escribimos <?php phpinfo(); ?>

```
> sudo nano /var/www/test.php
```

o alternativamente

```
> sudo gedit /var/www/test.php
```

Luego abrimos el navegador web y escribimos http://localhost/test.php; deberá aparecer la pantalla con toda la configuración de PHP.

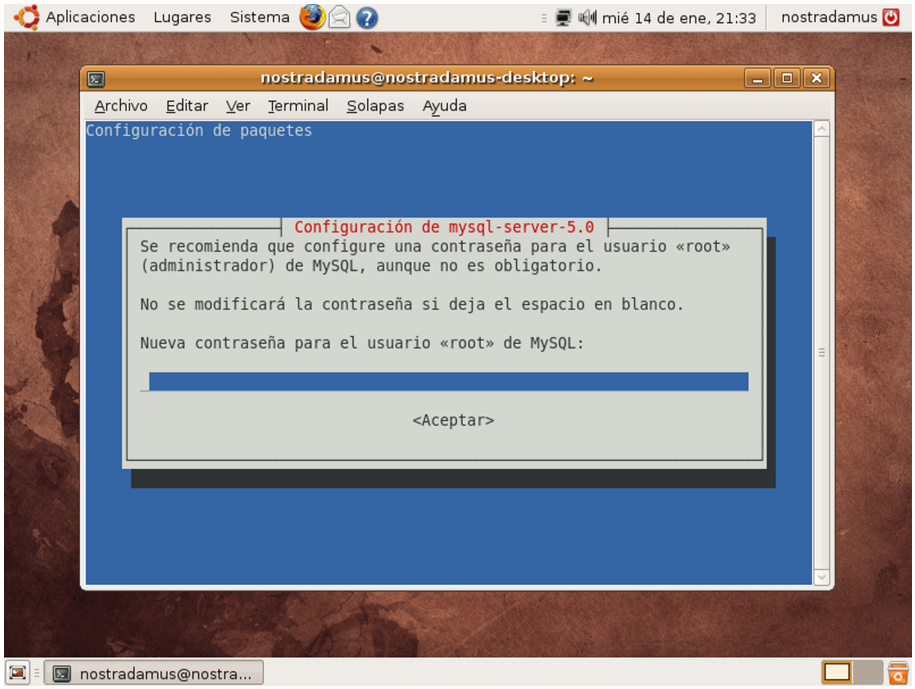


El archivo de configuración de PHP se encuentra en /etc/php5/apache2/php.ini (cada vez que lo modifiquemos habrá que reiniciar el servidor http para que los cambios tengan efecto).

3) Instalaremos ahora el servidor MySQL 5, el programa cliente mysql5, y el módulo mysql para php5. Los tres paquetes que se deben instalar se indican con el siguiente orden:

```
> apt-get install mysql-server mysql-client php5-mysql
```

4) Al finalizar el proceso de instalación se nos pedirá introducir la contraseña de root del administrador del servidor MySQL (introducimos la nueva contraseña y la confirmamos).



Si no hemos introducido la contraseña de root, lo podremos hacer más adelante, cuando lo creamos necesario. Por defecto, MySQL crea un usuario root sin ninguna contraseña. Podemos añadir la nueva contraseña de administrador conectándonos al servidor de base de datos con el programa cliente mysql, y, una vez dentro de la terminal de mysql, la cambiamos.

```
> mysql –u root
```

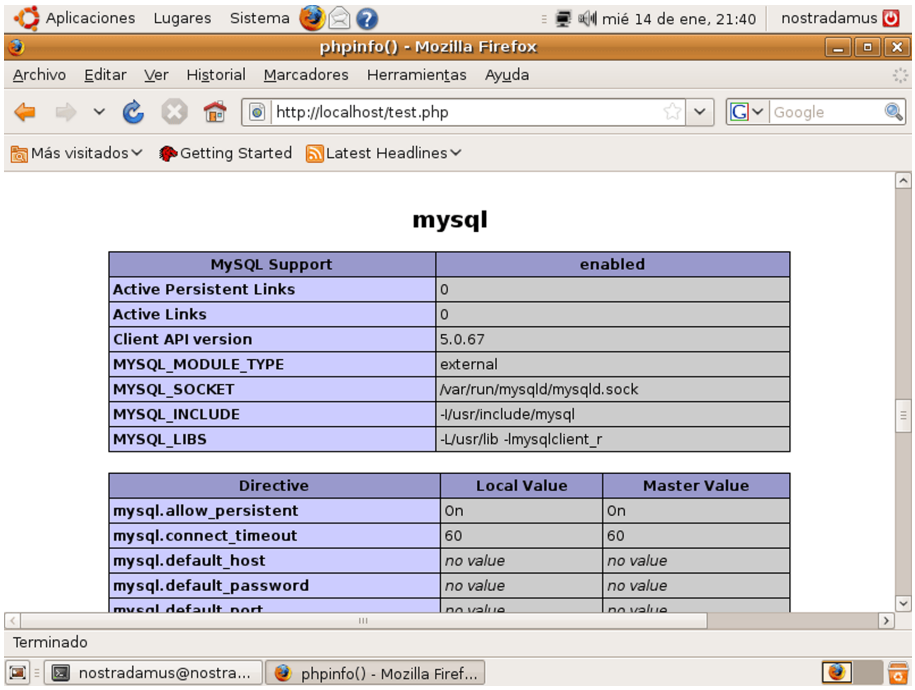
En la terminal de mysql escribimos:

```
mysql> USE mysql;
mysql> UPDATE user SET Password=PASSWORD('nuevo-password')
WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

5) La instalación del paquete php5-mysql (paso 3) hace accesible MySQL desde Apache con la API de PHP, pero será necesario reiniciar Apache para que este cambio tenga efecto.

```
> sudo /etc/init.d/apache2 restart
```

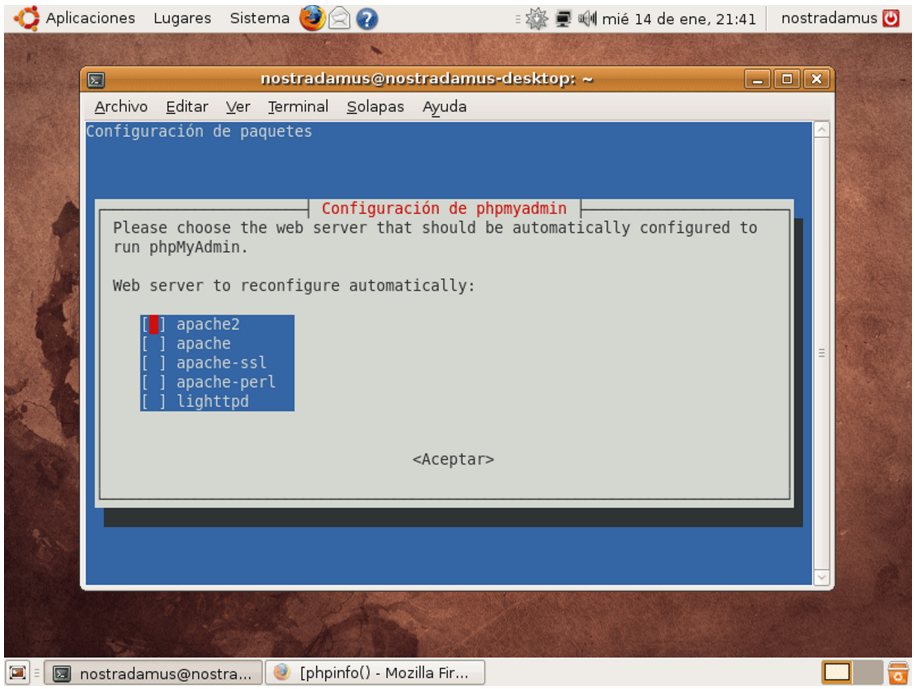
Volvemos a abrir la página de configuración de PHP con el navegador y comprobamos que esté disponible el módulo Mysql de PHP en Apache.



6) Opcionalmente, vamos a instalar el paquete de PhpMyAdmin desde la consola. Para ello, escribimos:

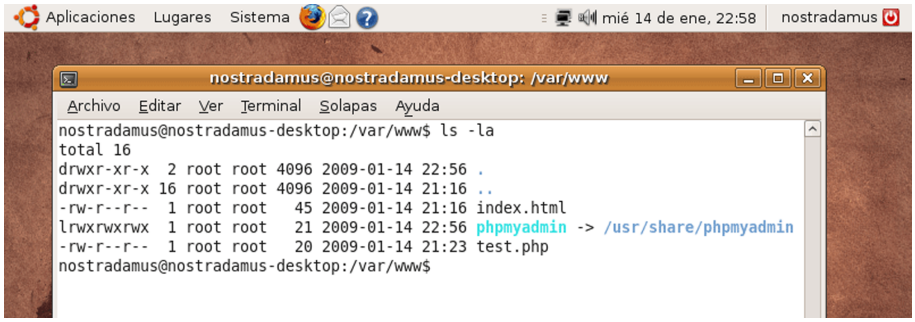
```
> sudo apt-get install phpmyadmin
```

Al final de la instalación marcamos la casilla apache2 (Apache 2.0).



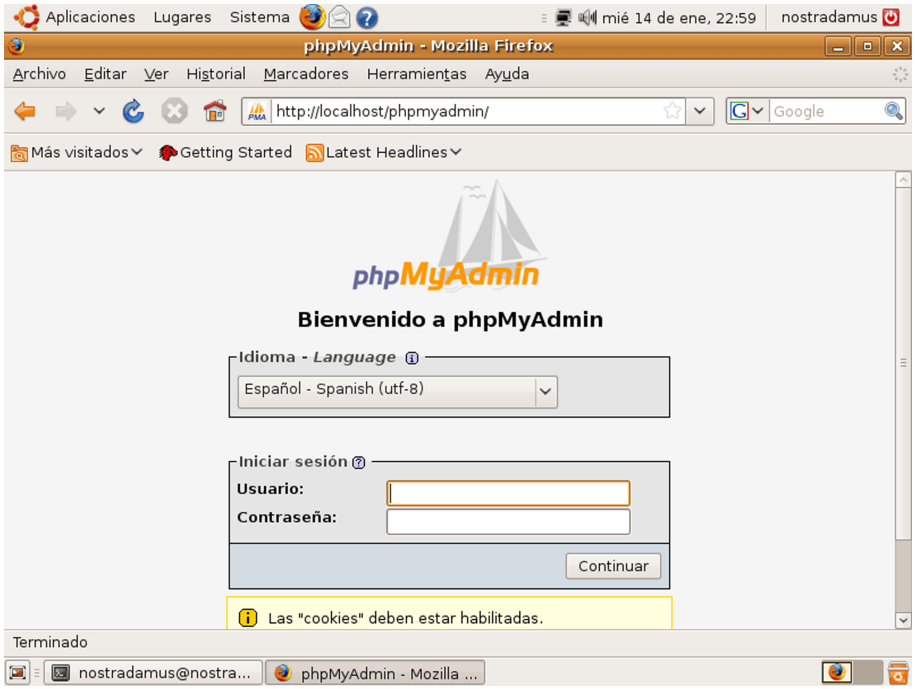
Después de instalar PhpMyAdmin desde los repositorios, hay que hacer un enlace simbólico desde /usr/share/phpmyadmin hacia /var/www/phpmyadmin para tener acceso a la aplicación web.

```
> sudo ln –s /usr/share/phpmyadmin /var/www
```



El archivo de configuración de PhpMyAdmin se encuentra en el directorio: /etc/phpmyadmin

Abrimos PhpMyAdmin con Firefox e introducimos las claves del administrador de MySQL para acceder a la aplicación.



Orientación a objetos en PHP

Dídac Gil de la Iglesia

PID_00155710

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

PHP es un lenguaje de programación estructurado con extensiones de orientación a objetos (OO). A continuación se explica cómo programar siguiendo una metodología orientada a objetos bajo PHP.

1. ¿Porqué usar PHP OO?

En PHP, una función declarada en una página PHP podrá ser usada inmediatamente después a su declaración.

Como primera aproximación, uno podría plantearse programar en cada fichero PHP todas las funciones necesarias que vayan a usarse en el fichero. En proyectos grandes, esto supone un alto coste de mantenimiento debido a la complejidad de seguimiento del código y a la duplicación del mismo código en distintas páginas. En caso de detectarse un error y el código que lo corrige, o en caso de querer aplicar una mejora en una función, deberán revisarse todos los *scripts* PHP para comprobar si contienen la función que hay que corregir.

Ved también

Dentro de la documentación en línea disponible en la web de PHP.net, existe una sección dedicada a la programación orientada a objetos. La programación OO se introdujo en PHP versión 4 y ha sido ampliada dentro de la versión 5. En caso de querer exprimir la OO al máximo en PHP, es recomendable realizar una lectura de las funcionalidades disponibles en OO y de las que puedan aparecer en futuras versiones.

Sin mucha dificultad puede observarse que la práctica de repetir códigos en *scripts* PHP queda lejos de ser una vía óptima para la programación en proyectos.

PHP permite definir clases, que poseen variables (también llamadas propiedades) y métodos (funciones). A continuación vemos un ejemplo donde se crea la clase Matematicas con 2 variables y dos métodos.

```php
<?php
class Matematicas{
    //Declaracion de las variables internas
    $error="";

    //Funciones

    //Sumar los valores
    function suma($valor1, $valor2)
    {
       if(is_numeric($valor1) && is_numeric($valor2))
       {
           $this->error="";
           return $valor1 + $valor2;
       }
       else
       {
           $this->error="Un valor no es numerico";
           return "NaN"
       }
    }

    //Restar los valores
    function resta($valor1, $valor2)
    {
       if(is_numeric($valor1) && is_numeric($valor2))
       {
           $this->error="";
           return $valor1 - $valor2;
       }
       else
       {
           $this->error="Un valor no es numerico";
           return "NaN"
       }
    }

    function obtener_error(){
       return $error;
    }

}
?>
```

Fichero matematicas_clase.php

Las clases permiten organizar mejor el código, ampliar nuestra productividad y eficiencia mediante la reutilización de código, la multiplicidad y la versatilidad, conceptos que trataremos en los siguientes puntos.

2. Organización del código

En PHP orientado a objetos, cada clase se programa dentro de un único fichero.

Posteriormente a la creación del fichero con la definición de la clase, pueden crearse instancias de la clase, llamadas objetos. Si hacemos un símil con el diseño de una silla, el prototipo de la silla es la "clase". Se definen las medidas, los materiales (variables) y los usos (funciones) que la silla puede tener. Cada silla física generada a partir de ese diseño será un "objeto" y podrá usarse dentro de diferentes ubicaciones, entornos y para diferentes propósitos.

En el siguiente ejemplo, se ha programado una clase llamada Texto que permite: formatear cadenas de texto para imprimirlas correctamente en HTML, formatear una cadena de caracteres que contenga una consulta SQL para minimizar el peligro de un ataque de seguridad, concatenar cadenas y eliminar espacios innecesarios.

```php
<?php
class Texto{
    //Declaracion de las variables internas
    $cadena="";

    //Funciones
    //Obtener el valor inicial de la cadena
    function entrar_cadena($variable){
        $this->cadena = $variable;
    }

    //Elimina los espacios existentes al inicio y fin de la cadena
    function limpiar_espacios(){
        return trim($this->cadena);
    }

    //Se inserta una cadena de caracteres delante o detrás de la cadena a trabajar
    function insertar_delante ($textoDelante){
       this->$cadena = $textoDelante . $this->$cadena;
    }

    function insertar_detras ($textoDetras){
       this->$cadena = $this->$cadena . $textoDetras;
    }


    //Escapa los caracteres especiales para evitar errores en SQL
    function preparar_SQL(){
        return addslashes($this->$cadena);
    }

    //Escribe en formato HTML los caracteres especiales, tales como los acentos
    function preparar_HTML(){
        return htmlspecialchars($this->cadena);
    }

    function mostrar_cadena(){
        return $this->cadena;
    }

}
?>
```

Fichero texto_clase.php

En el fichero aplicacion.php se crea un objeto de la clase texto con el nombre de *$cadena*. Sólo mediante la instancia *$cadena*, podemos acceder a las funciones existentes dentro de la clase.

```
<?php
    require_once("texto_clase.php"); //incorporación del fichero texto_clase.php para
    poder hacer uso de las clases declaradas en él

    $cadena = new Texto();  //creación de un objeto ($cadena) de la clase Texto
    $cadena->entrar_cadena($_POST['usuario']); //uso de una función de la clase Texto
    $cadena->limpiar_espacios();

    echo "Se ha seleccionado el usuario $cadena->preparar_HTML()";

    //Preparamos la consulta SQL para  validar el usuario
    $cadena->insertar_delante("SELECT nombre_completo FROM usuarios WHERE login='");
    $cadena->insertar_detras("'");

    $sql = $cadena->preparar_SQL();
    ...
?>
```

Fichero aplicacion.php

Con la programación orientada a objetos, las clases quedan localizadas dentro de un único fichero PHP. Durante el proceso de programación, suelen darse errores dentro del comportamiento de las aplicaciones, por lo que es necesario efectuar un análisis para detectar la función que está incorrectamente implementada. En la programación orientada a objetos, dada su organización, solamente será necesario corregir el error dentro de la clase correspondiente, y todos los objetos de la clase creados se beneficiarán de la corrección.

Por ejemplo, el programador podría detectar que ciertas cadenas al usarse con *htmlspecialchars* no son convertidas a lenguaje HTML, por lo que podría preferir usar la función *htmlentities*. Si el programador ha creado la clase Texto, simplemente deberá modificar la función *preparar_HTML*.

```
function preparar_HTML(){
    return htmlentities($this->cadena);
}
```

Si, por el contrario, no ha usado la clase, deberá buscar todas las referencias a *htmlspecialchars* dentro de su proyecto y sustituirlas por la nueva función, lo que supondrá una sobrecarga de tiempo, y un alto potencial a olvidarse alguna referencia sin corregir.

3. Reutilización de código

La creación de un portal dinámico en PHP suele componerse de páginas de presentación, formularios, obtención de información existente dentro de bases de datos, tratamiento de datos obtenidos desde formularios, bases de datos, portales externos, páginas XML, RSS, etc. Hay funcionalidades comunes a la mayoría de los proyectos, como la validación de variables o el acceso a bases de datos, por lo que el programador tiende a reutilizar código implementado en proyectos anteriores, con lo que incrementa así su eficiencia y evita posibles errores de programación.

Estamos acostumbrados a reutilizar código de terceras personas continuamente, como por ejemplo la API de PHP, con funciones tales como *mysql_query()*, *require()* o hasta un simple *echo()*.

A mayor escala, un programador podrá aprovechar un subconjunto de las funciones creadas en proyectos anteriores en sus nuevos proyectos, como por ejemplo funciones para establecer y gestionar sesiones, o validar los usuarios en la base de datos.

Sin hacer uso de la programación OO tendríamos nuestro código repartido en varios ficheros PHP. Además, el código PHP que se podría reutilizar estaría entremezclado con código HTML del portal.

Usando programación OO, todas las funciones quedarán dentro de una o varias clases y, por lo tanto, el correspondiente código se encontrará en un solo fichero PHP, lo que permitirá separar el código PHP de las líneas HTML. Para usarlo en una página PHP, el programador simplemente tendrá que cargar el fichero correspondiente a la clase que quiere reutilizar en la página PHP. La carga del fichero con la clase se realizará con la llamada *require_once()*.

4. Multiplicidad

Como ya hemos detallado, una clase es un "prototipo" que define variables y funciones que pueden realizarse sobre estas variables.

> Una vez definida una clase, podemos instanciar tantos objetos de la clase como sean necesarios, teniendo todos ellos las características de la clase, es decir, sus variables y funciones.

Para hacer más simple el entendimiento del concepto multiplicidad, expondremos el siguiente ejemplo.

Dentro de una aplicación web en la que se debe hacer un traspaso de dinero entre dos cuentas bancarias, existe una cuenta a la que se hará un ingreso de dinero (cuenta destino, B) que proviene de una cuenta a la que se le deberá sustraer dicho saldo (cuenta origen, A).

Un traspaso de un saldo supone una consulta de saldo en la cuenta origen y una actualización en la misma cuenta para restar el saldo que se va a traspasar. Por la parte del beneficiario, implica una consulta del saldo actual para poder realizar una actualización que lo incremente con el traspaso.

Para ello, se ha creado una clase Cuenta que permite obtener el saldo de una cuenta corriente, así como actualizar su saldo.

```php
<?php
class Cuenta{
    //Declaracion de las variables internas
    $num_cuenta;
    $saldo;

    //Declaración del número de cuenta bancaria
    function definir_cuenta($numero_cuenta)
    {
        $this->num_cuenta = $numero_cuenta;
    }

    //Consulta el saldo de la cuenta asociada
    function obtener_saldo()
    {
        $this->saldo = sql_obtener("SELECT saldo FROM cuentas WHERE numero_cuenta = $this->num_cuenta");
        return $this->saldo;
    }

    //Funciones
    //Dar valores a las variables
    function entrar_saldo($valor)
    {
        $this->saldo= $valor;
        sql_actualizar("UPDATE saldo FROM cuentas WHERE numero_cuenta = $this->num_cuenta");
    }

}

?>
```

Fichero cuenta_clase.php

En la aplicación PHP, se hace uso de la clase creada Cuenta y la clase Matematicas.

```
<?php
    require_once("matematicas_clase.php");
    require_once("cuenta_clase.php");
    …

    //Definimos una función que permite el traspaso entre cuentas. Usa dos
    objetos de la clase Cuenta y un valor numérico que representa el importe
    a traspasar entre las cuentas. La función hace uso de la clase Matematicas
    para comprobar que existe saldo suficiente en la cuenta origen para el
    traspaso.

    function traspaso_saldo($origen, $destino, $importe)
    {
       $mensaje="";
       $calculadora=new Matematicas();

       //comprobar saldo origen
       if($calculadora->resta($origen->obtener_saldo(), $importe) <0) {
           $error=$calculadora->obtener_error();
           if($error <> "") return "El campo importe no es numérico";
           else             return "No hay saldo suficiente en la cuenta origen";
       }
       else{
           //hay saldo suficiente. Se realiza el traspaso
           $importe_origen=$calculadora->resta($origen->obtener_saldo(), $importe);
           $origen->entrar_saldo($importe_origen);

           $importe_destino=$calculadora->suma($importe, $destino-> obtener_saldo());
           $destino->entrar_saldo($importe_destino);
           return "Importe traspasado";
       }
    }
```

```
    //Se consulta el saldo actual de cada una de las cuentas,
    probablemente desde una base de datos, y se muestran estos
    saldos por pantalla
    $cuenta_A = new Cuenta();
    $cuenta_A->definir_cuenta("0123 4567 8901 2345");
    $saldo_origen=$cuenta_A->obtener_saldo();
    echo "El saldo de la cuenta A antes del traspaso es ".$saldo_origen."€";

    $cuenta_B = new Cuenta();
    $cuenta_B->definir_cuenta("9876 5432 1098 7654");
    $saldo_destino=$cuenta_B->obtener_saldo();
    echo "El saldo de la cuenta B antes del traspaso es ".$saldo_destino."€";

    //Se lee, a partir de variables de un formulario Web enviado,
    la cantidad de dinero a transferir desde la cuenta A
    a la cuenta B
    $importe_traspaso=$_POST['cantidad_dinero'];

    //Se usa la función traspaso_saldo para realizar la
    transferencia bancaria
    traspaso_saldo($cuenta_A, $cuenta_B, $importe_traspaso)

    //Para comprobar la correcta transferencia bancaria, se vuelven
    a mostrar los
    saldos actualizados.
    $saldo_origen=$cuenta_A->obtener_saldo();
    echo "El saldo de la cuenta A despues del traspaso es ".$saldo_origen."€";

    $saldo_destino=$cuenta_B->obtener_saldo();
    echo "El saldo de la cuenta B despues del traspaso es ".$saldo_destino."€";

}
?>
```

Dado que se va a trabajar con dos cuentas bancarias, se han instanciado dos objetos de la clase Cuenta. La multiplicidad es la capacidad de usar múltiples instancias de una misma clase, con lo que se puede simplificar el código de una aplicación PHP y mantener el estado de varios objetos al mismo tiempo.

5. Herencia

En ocasiones, el código no puede reutilizarse tal cual, sino que debe especializarse o refinarse para tener en cuenta nuevas necesidades. Uno podría crear una nueva clase, reescribiendo todo el código reaprovechable en ella e incluir también las nuevas utilidades implementando las funciones necesarias. No obstante, eso limitaría la posible reutilización de código.

Una de las ventajas de la programación orientada a objetos es el uso de la herencia, que soluciona el problema anterior. La herencia permite crear una nueva clase que "herede" las características (variables y métodos) de otra clase. En PHP la herencia se indica mediante el código EXTENDS.

La clase Cuenta vista en el subapartado anterior tenía la limitación de ser monodivisa, por lo que las cuentas origen y destino deben compartir la misma moneda para poder hacer las transferencias bancarias.

```
<?php
require_once("cuenta_clase.php");

class Cuenta_Multidivisa extends Cuenta{
    //Declaracion de las variables internas
    $divisa;

    //Establece el numero de cuenta bancaria y la divisa por defecto con la que se
    quieren mostrar los valores de la cuenta.
    function definir_cuenta($valor, $divisa)
    {
        $this->num_cuenta = $valor;
        $this->divisa=$divisa;
    }

    //Permite obtener la divisa por defecto usada para la cuenta
    function consultar_divisa()
    {
        return $this->divisa;
    }

    //Consulta el valor de la cuenta en una divisa
    function aplicar_cambio($divisa_destino)
    {
        $coeficiente=obtener_cambio_BancoEuropeo($this->divisa, $divisa_destino);
        return $this->saldo * $coeficiente;
    }

    //Cambia la divisa por defecto con la que se quieren mostrar valores de la
    cuenta.
    Actualiza la cantidad de saldo para mostrar su correcto valor en la divisa
    correspondiente.
    function cambiar_divisa ($nueva_divisa)
    {
        if ($this->saldo = 0 || $this->divisa = "")
              $this->divisa= $nueva_divisa;
        else{
              $this->saldo = $this->aplicar_cambio($nueva_divisa);
              $this->divisa= $nueva_divisa;
    }

}
?>
```

Fichero cuenta_multidivisa_clase.php

Haciendo una clase Cuenta_Multidivisa que extienda la clase Cuenta, podrá añadirse una nueva funcionalidad que haga el cambio entre divisas consultando el estado actual del cambio en el Banco Europeo.

6. Visibilidad

En la programación orientada a objetos sobre PHP, las funciones y las variables existentes en los objetos tienen tres niveles de visibilidad, clasificados en Public, Protected y Private, de más a menos permisivo respectivamente.

Imaginemos la siguiente aplicación PHP que hace uso de la clase Cuenta_Multidivisa.

```
<?php
    require_once("cuenta_multidivisa_clase.php");

    //Se consulta el saldo actual de cada una de las cuentas, probablemente desde
    una base de datos
    $cuenta_A = new Cuenta_Multidivisa();
    $cuenta_A->definir_cuenta("0123 4567 8901 2345", "Peseta");

    $cuenta_A->entrar_valores(5000);

    $cuenta_A->divisa ="Euro";
}
?>
```

Hasta el momento sólo habíamos accedido a las funciones de la clase Cuenta_Multidivisa pero, así como podemos acceder a las funciones de la clase, también es posible acceder a sus variables.

Por defecto, la visibilidad de las funciones y las variables es Publica, lo que quiere decir que, una vez instanciado un objeto de la clase, puede accederse directamente a sus variables internas y a sus funciones.

En ciertos casos, puede ser beneficioso acceder a las variables internas de un objeto. En otros casos, como el expuesto en el código anterior, puede ser una fuente de errores. Podemos ver que el objeto cuenta_A, gracias a que ha podido accederse a la variable divisa, ha pasado de tener 5.000 pesetas a tener 5.000 euros.

Veamos ahora el mismo código mejorado para evitar tales errores.

```
<?php
require_once("cuenta_clase.php");

class Cuentas_Multidivisa extends Cuenta{
    //Declaracion de las variables internas
    private $divisa;

    //Establece el numero de cuenta bancaria y la divisa por defecto con la que
    se quieren mostrar los valores de la cuenta.
    public function definir_cuenta($valor, $divisa)
    {
        $this->num_cuenta = $valor;
        $this->divisa=$divisa;
    }

    //Permite obtener la divisa por defecto usada para la cuenta
    public function consultar_divisa()
    {
        return $this->divisa;
    }

    //Consulta el valor en una divisa destino
    public function aplicar_cambio($divisa_destino)
    {
        $coeficiente=obtener_cambio_BancoEuropeo($this->divisa, $divisa_destino);
        return $this->saldo * $coeficiente;
    }

    //Cambia la divisa por defecto con la que se quieren mostrar valores de
    la cuenta.
    Actualiza la cantidad de saldo para mostrar su correcto valor en la divisa
    correspondiente.
    public function cambiar_divisa ($nueva_divisa)
    {
        if ($this->saldo = 0 || $this->divisa = "")
              $this->divisa= $nueva_divisa;
        else{
              $this->saldo = $this->aplicar_cambio($nueva_divisa);
              $this->divisa= $nueva_divisa;
    }

}
?>
```

Fichero cuentas_multidivisa_clase.php

Siendo Private, la variable divisa sólo podrá ser accesible dentro de las funciones de la clase. Las variables y funciones Private y Protected no son directamente visibles desde los objetos, por lo que la llamada siguiente dará un error de acceso:

```
$saldo_A->divisa =”Euro”;
```

Como consecuencia, se obligará al uso de la función pública establecer_divisa, que tiene implementados los cambios de divisa pertinentes para mantener la consistencia de los datos.

```
<?php
    require_once("cuentas_multidivisa_clase.php");

    //Se consulta el saldo actual de cada una de las cuentas, probablemente desde una
    base de datos
    $cuenta_A = new Cuenta_Multidivisa();
    $cuenta_A->definir_cuenta("0123 4567 8901 2345");

    $cuenta_A->establecer_divisa("Peseta");
    $cuenta_A->entrar_valores(5000);

    $cuenta_A->divisa ="Euro";   //Error. No es posible acceder
                                  //a $saldo_A->divisa_origen

    $cuenta_A->establecer_divisa("Euro");  //El saldo ha pasado a ser 30.05
}
?>
```

Así como en el ejemplo del cambio de divisa, existen diversidad de casos donde variables, e incluso funciones internas, deberán ser privadas para obligar a la lectura o modificación de estas variables a través de funciones donde se hayan implementado los controles deseados.

Como ejemplos, podemos pensar en las variables de usuario en la autenticación de una aplicación. El cambio de contraseña deberá ser mediante la llamada a una función, asegurando así que la nueva contraseña tiene la complejidad adecuada, se actualiza el valor tanto en la sesión como en la base de datos, etc.

```
<?php
class sesiones{
    //Declaracion de las variables internas
    private $login;
    private $password;

    public cambio_password($old, $new_password) {
       if(compleja($new_password && $this->password == $old) ) {
              actualizar_password($this->login, $new_password);
              $this->password = $new_password;
              return 0;  //Sin errores
       else
              return -1; //Con errores
    }
}
?>
```

7. Sobrecarga de clases

Como hemos visto hasta ahora, para poder instanciar objetos pertenecientes a las clases implementadas, debemos usar alguno de los comandos “require”, “include”, “require_once” o “include_once”.

En ocasiones, la lista de clases que vamos a requerir para una aplicación PHP puede ser larga, por lo que deberemos hacer una gran cantidad de sentencias “require” para poder hacer uso de todas ellas.

En PHP versión 5, se ha creado la función __autoload() que, en caso de no disponerse de una clase en la creación de un objeto, es lanzada para tratar de evitar errores.

Un programador puede definir el contenido de la función __autoload() para tratar dichos casos.

Como habréis observado, las clases generadas: Texto, Matematicas, Cuenta, y Cuenta_Multidivisa están declaradas dentro de ficheros .php que tienen por nombre texto_clase.php, matematicas_clase.php. cuenta_clase.php y cuenta_multidivisa_clase.php. Generalizando, podemos deducir que cada clase se encuentra en un fichero .php que tiene como nombre <nombre_clase>_clase.php

Aprovechando la funcionalidad __autoload(), y una estructura de ficheros bien definida, es posible solicitar la carga de las clases bajo demanda.

Veamos el siguiente código.

```
<?php
   function __autoload($nombre_clase) {
          require_once $nombre_clase . '_clase.php';
   }

   $cuenta_A = new Cuenta();

?>
```

La creación del objeto cuenta_A por si solo daría un error de ejecución al no conocerse la existencia de la clase Cuenta, ya que no ha sido cargada. En la versión 5 de PHP, antes de lanzar dicho mensaje de error se ejecutara la función “__autoload()” pasando como parámetro el nombre de la clase faltante. En nuestro caso, se lanzará “__autoload(Cuenta)”, haciendo que se ejecute la siguiente sentencia:

```
require_once Cuenta_clase.php;
```

Dicha sentencia corregirá el problema cargando la clase Cuenta. Cabe notar también que la clase Cuenta se añadirá solamente cuando sea requerida para la instanciación de un objeto. Su uso nos puede ahorrar la molestia de tener que escribir el correspondiente “require_once” para cada una de las clases que vayamos a usar.

8. Constructores y destructores

Otro de los conceptos existentes en la programación orientada a objetos son los constructores y los destructores.

Se trata de la implementación de funciones que se lanzan automáticamente en el proceso de creación y de destrucción de los objetos usados en una aplicación PHP.

Los **constructores** suelen usarse para la inicialización de variables durante el proceso de instanciación de objetos, y tienen como nombre de función __construct().

Los **destructores**, por otro lado, suelen usarse para hacer salidas controladas de aplicaciones, tales como el cierre de ficheros, cierre de sesiones con bases de datos o similares. La función destructor tiene el nombre __destruct().

La función constructora __construct permite la entrada de variables para la instanciación.

```
<?php
   class Cuenta_2{
          private $divisa_origen;
          private $saldo;
          private $numero_cuenta;

          public function __construct($cuenta, $divisa) {
                $this->numero_cuenta=$cuenta;
                $this->divisa_origen=$divisa;
          }
   }

   ...
?>
```

Como ejemplo práctico de constructores y destructores presentamos el acceso a bases de datos. Permitirá establecer conexiones con una base de datos definida por defecto de manera automática con sólo instanciar un objeto, y asegurar su desconexión al salir del fichero PHP.

```
<?php
require("parametros_conexion.php"); // fichero que contiene user, password,
                                    // server, DBname
class ConexionBBDD{
   private conexion;

   public function __construct() {
          $this->conexion = mysql_connect($server, $user, $password);
          mysql_select_db( $DBname , $this->conexion );
   }

   public function __destruct () {
          mysql_close( $this->conexion );
   }

}

…

?>
```

Uso de formularios en HTML para enviar y recopilar datos

Piero Berni Millet

PID_00155708

Universitat Oberta de Catalunya
www.uoc.edu

Índice

1. Introducción a CGI y su entorno

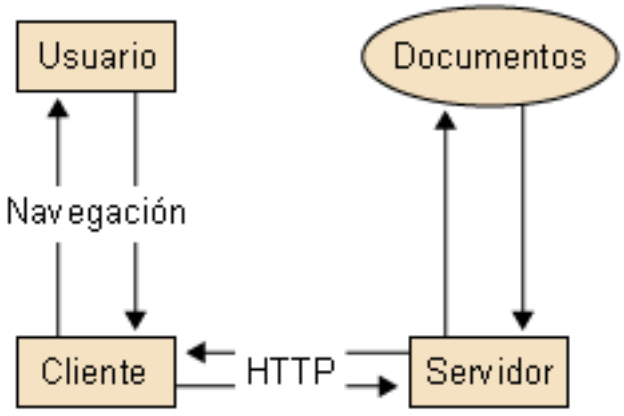
La interfaz de pasarela común (CGI[1]) es la puerta de acceso que hay entre una página web y el servidor de Internet donde reside la página.

(1)CGI es la sigla inglesa de *common gateway interface*.

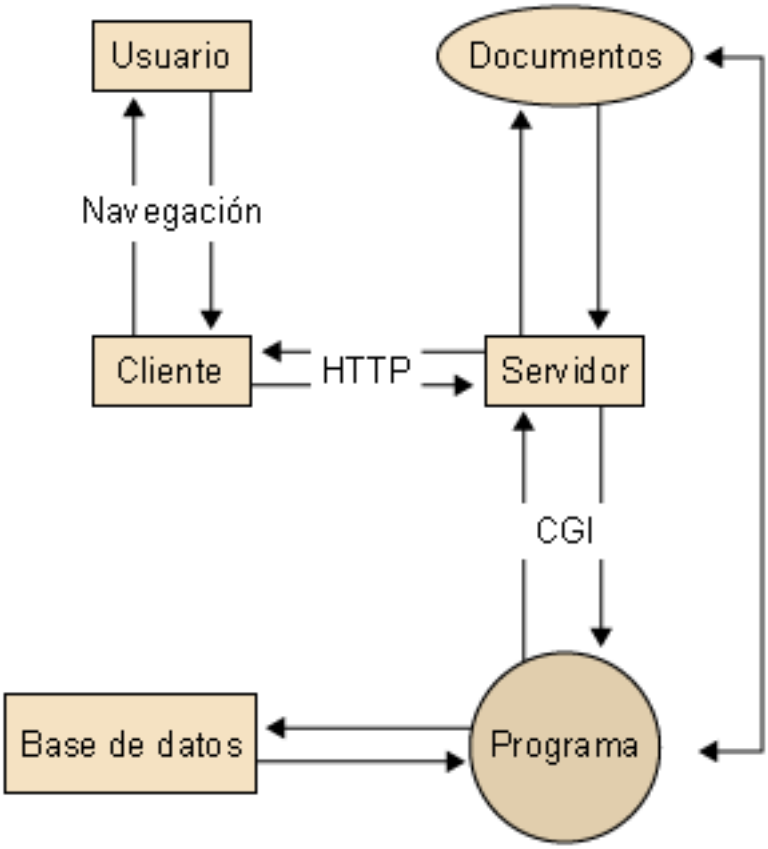
Si habéis utilizado un navegador web, os habréis encontrado con páginas web que permiten interrogar bases de datos para obtener información. ¿Qué está sucediendo detrás de la página? En resumen, el navegador envía solicitudes al servidor y el servidor envía respuestas al navegador. Este intercambio es una cuestión sencilla cuando la solicitud es mostrar otra página web. Pero cuando el navegador desea algo más complejo, como el último parte meteorológico en España, hay grandes posibilidades de que el gestor de la información detrás de la página web sea un programa escrito con un *script* CGI.

Lenguaje de un *script* CGI

Un *script* CGI puede ser escrito en cualquier lenguaje que pueda leer de STDIN (entrada estándar), escribir en STDOUT (la salida estándar) y leer variables de entorno como virtualmente cualquier lenguaje de programación, incluyendo C, Perl, PHP, o incluso *scripts* de Shell de UNIX y LINUX.



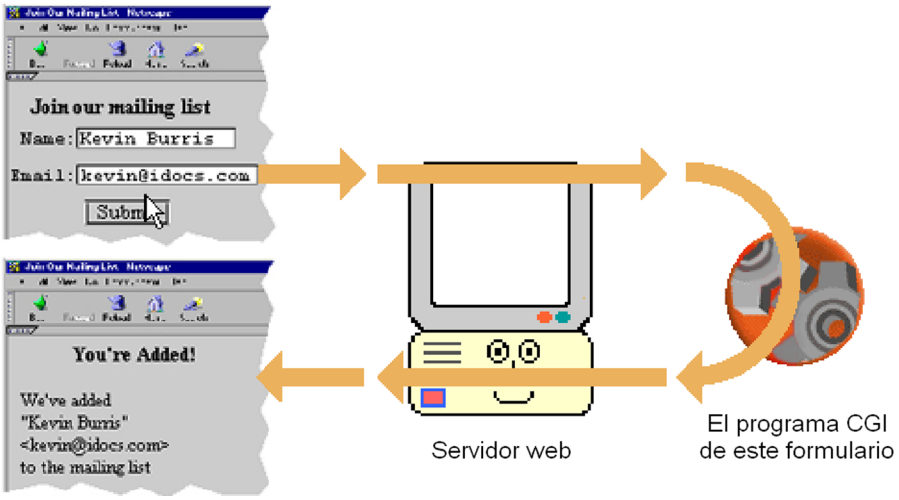
Solicitud http sencilla para mostrar otra página web



Solicitud http combinada con CGI

La programación CGI consiste en el diseño y la escritura de programas que reciben sus órdenes iniciales desde una página web. Los formularios HTML se han convertido en el método preferido para enviar datos a través de Internet debido a su facilidad de uso y de configuración.

La idea básica detrás de un formulario es sencilla, tal y como se ilustra en la siguiente figura. El visitante abre un formulario y lo rellena; el navegador envía los datos al servidor; el programa CGI captura la corriente de datos del formulario y la procesa; el programa CGI devuelve algún tipo de respuesta al visitante a través del servidor.



La idea básica detrás de un formulario

2. Uso de formularios HTML/XHTML

Las etiquetas <form> son la base para el traspaso de datos a los programas CGI en el servidor.

> Un formulario de una página web es una colección de campos de entrada que comunican su contenido al servidor.

En el mundo de Internet y de las páginas web se suelen utilizar para múltiples propósitos: formularios de pedidos de una tienda electrónica, cuestionarios, encuestas, libros de invitados, consultas a bases de datos, etc. Todos ellos devuelven datos al propietario de la página web para que sean procesados posteriormente.

En este apartado vamos a ver sólo el modo de situar un formulario en una página web. Más adelante aprenderemos a procesar los datos que se reciban desde un formulario.

2.1. Tipos de entradas de formularios

Los elementos de los formularios HTML/XHTML ofrecen una variedad de formas de aceptar las entradas de los datos de los visitantes: entradas de líneas de texto, campos de texto formados por múltiples líneas, casillas de verificación, botones de opción, listas de opciones de varios tipos, campos ocultos para administrar la información, etc.

Podéis colocar cualquiera de estos elementos en el cuerpo de un documento HTML/XHTML si se delimitan con la etiqueta <form> y su respectiva etiqueta de fin, </form>. Todos los elementos dentro de estas dos etiquetas componen un único formulario. Los elementos de entrada de datos más comunes son: <input>, <select>, y <textarea>.

2.2. La etiqueta <input>

Los elementos de entrada de un formulario se crean principalmente con la etiqueta <input>. Ésta define el comportamiento y la posición del campo para que acepte datos del visitante en un formulario. Esta etiqueta acepta una serie de atributos como *type*, *name*, *value*, *size*, *maxlength*, etc.

En el siguiente ejemplo se muestra un formulario con tres elementos para introducir texto, “nombre”, “apellidos” y “direccion”:

- Los dos primeros elementos <input> son cuadros de texto de una única línea (type="text") de 30 y 50 caracteres de ancho (*size*); permite al usuario teclear sólo 30 y 50 caracteres (maxlength), respectivamente; el contenido del campo se muestra, por defecto, en blanco al no existir el atributo value.

- El tercer campo es otro cuadro de texto con un tamaño de visualización de 75 caracteres y con capacidad máxima para 100 caracteres; establece un valor inicial (value=" Calle ") como encabezamiento del resto de la información que va a rellenar el usuario.

El nombre de la variable que identifica cada campo se indica en el atributo *name*. Este nombre es imprescindible para que el servidor pueda procesar el formulario.

```
<html>
<head><title>Ejemplo de formulario sencillo</title></head>
<body>

<h3>Formulario muy sencillo</h3>

<form action=" formulario.php" method="post">
  Escribe tu nombre:
  <input name="nombre" type="text" value="" size="30" maxlength="30" />
  <br/>
Escribe tus apellidos:
<input name="apellidos" type="text" value="" size="50" maxlength="50" />
  <br/>
Escribe tu dirección particular:
<input name="direccion" type="text" value="Calle " size="75" maxlength="100" />
  <br/>
  <input type="submit" value="Enviar" />
</form>

</body>
</html>
```

El atributo *type* indica el tipo de control que se incluye en el formulario y que puede tener los siguientes valores:

```
"text | password | checkbox | radio | submit | reset | file
| hidden | image | button"
```

A continuación presentamos el significado de los controles en función de su tipo:

1) type="password". Oculta la entrada del usuario reemplazándola por asteriscos en la pantalla, por lo que es ideal para escribir contraseñas y otros datos privados. El navegador los transmite sin encriptar cuando se envían al servidor.

```
Contraseña <br/>
```

<input type=”password” name=”contrasena” value=”” />

2) type=”checkbox”. Las casillas de verificación son controles de formulario que permiten al usuario seleccionar y deseleccionar opciones individualmente. Si se quiere mostrar un checkbox seleccionado por defecto, se utiliza el atributo *checked*. Las casillas de verificación no seleccionadas no aparecen en los datos enviados.

```
Puestos de trabajo buscados <br/>
<input name="puesto_directivo" type="checkbox" value="directivo"/> Directivo
<input name="puesto_tecnico" type="checkbox" value="tecnico"/> Técnico
<input name="puesto_empleado" type="checkbox" value="empleado"/> Empleado
```

3) type=”radio”. Botones radiales. Los controles de tipo radiobutton son similares a los controles de tipo checkbox, pero presentan una diferencia muy importante: son mutuamente excluyentes. Son adecuados para aceptar un valor individual de un conjunto de alternativas. Todos los botones radiales de un mismo grupo reciben el mismo nombre. Solamente el botón radial seleccionado del grupo aparece en los datos enviados. Uno de ellos puede estar seleccionado desde el inicio si se le incluye el atributo *checked*.

```
Tipo de inscripción <br/>
<input type=”radio” name=”inscripcion” value=”completa”
checked=”checked” /> Cuota completa: 180 €
<input type=”radio” name=”inscripcion” value=”general” />
Cuota general: 80 €
<input type=”radio” name=”inscripcion” value=”estudiante” />
Cuota estudiante: 50 €
```

4) type=”hidden”. Los campos ocultos se emplean para añadir información oculta en el formulario, por lo que no aparece en la pantalla de la página web, de modo que el usuario desconoce que el formulario los incluye. Se utiliza para comunicar información sobre el estado de la interacción entre el cliente y el servidor, que necesita el servidor pero que no es necesario o no es posible que la establezca el usuario; por ejemplo, un identificador de transacción.

```
<input type="hidden" name="url_previa" value="/articulo/primero.html" />
```

5) type=”submit”. Botón de envío de formulario. La mayoría de los formularios disponen de un botón para enviar al servidor los datos introducidos por el usuario. El valor del atributo *value* es el texto que muestra el botón. Si no se establece el atributo *value*, el navegador muestra el texto predefinido Enviar consulta. Pero si se proporciona un atributo *name*, podremos incluir los atributos *name* y *value* entre los datos enviados. Esta opción es útil cuando se coloca en el formulario más de un botón de envío, ya que brinda un modo de identificar en qué botón del formulario se ejecutó la acción.

```
<input type="submit" name="buscar" value="Buscar" />
```

6) type="reset". Botón de reseteo o restauración del formulario. Se trata de un botón especial que borra todos los datos introducidos por el usuario y devuelve el formulario a su estado original.

```
<input type="reset" name="limpiar" value="Borrar datos del
formulario" />
```

2.3. La etiqueta <select>...</select>

Las casillas de verificación y los botones de selección son una buena solución para crear preguntas y respuestas de selección múltiple en un formulario, pero pueden llegar a convertirse en tediosas y confusas cuando aparecen en pantalla largas listas de elementos. Una manera alternativa y más compacta de representar listas de elementos es la etiqueta select que se representa como un menú desplegable o emergente, y que consta de una o de múltiples opciones. Sus atributos son los siguientes:

- name="un_nombre". El nombre de la variable que identifica este campo.
- size="valor". Número de filas que se muestran de la lista (por defecto sólo se muestra una).
- multiple. La presencia de este atributo indica que se puede seleccionar más de una opción al mismo tiempo.

Las listas desplegables se definen con la etiqueta <select> y cada elemento de la lista se define mediante la etiqueta <option>. El elemento de una lista desplegable tiene dos atributos comunes: *selected* = "selected", indica si el elemento aparece seleccionado por defecto al cargarse la página; *value* = "texto", el valor que se envía al servidor cuando el usuario elige esa opción. A continuación podemos ver un ejemplo donde se crea una lista desplegable para elegir un sistema operativo:

```
<label for="so">Sistema operativo</label> <br/>
<select id="so" name="so">
  <option value="" selected="selected">- selecciona -</option>
  <option value="windows">Windows</option>
  <option value="mac">Mac</option>
  <option value="linux">Linux</option>
  <option value="otro">Otro</option>
</select>
```

2.4. La etiqueta <textarea>...</textarea>

Como hemos visto, el tipo de entrada <input type="text"> limita la entrada de datos a una sola línea de caracteres. La etiqueta <textarea> da más libertad al usuario al permitir introducir una gran cantidad de texto repartido en una línea de texto. El texto que se extiende hasta la etiqueta de cierre se emplea para inicializar el valor del campo:

```
<form action="proyecto.php" method="post">
   <label for="titulo">Título del proyecto</label> <br/>
   <input type="text" name="titulo" value="" />
  <label for="descripcion">Descripción del proyecto</label>
      <br/>
   <textarea name="descripcion" cols="40" rows="5">
      </textarea>
</form>
```

Los atributos específicos son los siguientes:

- name="un_nombre". El nombre de la variable que identifica el campo.
- rows="numero". Número de líneas que mostrarán en el cuadro de texto.
- cols="numero". Número de caracteres que se muestran en cada fila.

2.5. Distintas maneras de enviar los datos de un formulario (atributo *method*)

La etiqueta <form> de HTML se vale de los atributos especiales *action* y *method* para informar al navegador de cómo codificar los datos y dónde enviarlos:

```
<form action="http://comoras.uoc.edu/aplicacion.php" method="post">
...
</form>
```

El atributo *action* = "url" proporciona la URL absoluta o relativa de la aplicación que va a recibir y procesar los datos del formulario. La aplicación es típicamente un *script* de un programa (escrito usualmente en Perl o PHP), que puede separar los datos y darles formato.

El atributo *method* indica cómo deben enviarse los datos desde el formulario al servidor. La diferencia entre estos dos métodos es la siguiente:

- Con el método GET los parámetros se pasan como parte de la URL que llama a la aplicación para procesar el formulario del lado del servidor. El navegador agrega los datos a la URL asignada en action, separados por el signo de interrogación "?". El método GET tiene varias limitaciones importantes: no puede manejar muchos datos debido a que algunos sistemas operativos limitan el número y la longitud de argumentos de línea de co-

mandos (en general, admite como máximo el envío de unos 500 bytes de información), que pueden pasarse a una aplicación a la vez. Además, GET coloca los parámetros del formulario directamente en la URL, donde pueden ser capturados con facilidad por los usuarios de Internet, poniendo en peligro la seguridad del sistema de envío. La otra gran limitación del método GET es que no permite el envío de archivos adjuntos con el formulario.

- El método POST es el preferido y el más recomendado. Este método también envía los datos como un añadido a una URL, pero los envía después de que se hayan enviado todas las cabeceras de solicitud del servidor, y por lo tanto invisible al usuario. La principal ventaja de POST es la de permitir enviar formularios con muchos campos o con campos de texto extensos.

Si no sabéis qué método elegir para un formulario, existe una regla general que dice que el método GET se debe utilizar en los formularios que no modifican la información (por ejemplo, en un formulario de búsqueda). Por su parte, el método POST se debería utilizar cuando el formulario modifica la información original (insertar, modificar o borrar alguna información).

El atributo *enctype* especifica el tipo de codificación empleada al enviar el formulario al servidor (sólo se indica de manera explícita en los formularios que permiten adjuntar archivos). El valor predeterminado es: application/x-www-form-urlencoded. Aparte de esta codificación, el tipo de codificación para adjuntar un archivo se indica con multipart/form-data.

El formato multipart/form-data se utiliza para los formularios que contienen campos para seleccionar archivos que el usuario puede obtener.

El formato text/plain se puede utilizar junto con la URL de tipo *mailto* en el atributo *action* para enviar el formulario a una dirección de correo electrónico. Por ejemplo, para enviar los datos de un formulario por correo electrónico, sin ser procesados por un servidor, escribiremos:

```
<form method="post" action="mailto:user@server.com" enctype="text/ plain">
```

2.6. Codificación estándar *application/x-www-form-unlencoded*

Los formularios lanzan sus datos con la codificación estándar de la URL. Las reglas principales de este formato de datos se explican a continuación.

Todos los datos que se han introducido desde un formulario se envían al servidor o al programa CGI como pares de nombre/valor. Los pares de nombre/valor se pasan siempre al servidor como name=value y cada nuevo par se separa por medio de un signo &, name1=value1&name2=value2.

Por ejemplo, observad la siguiente entrada:

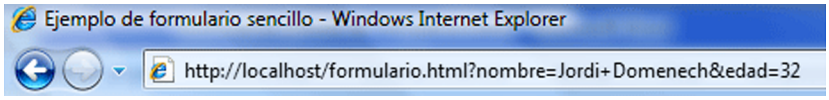
```
<h3>Formulario muy sencillo</h3>
<form method="get">
  Escribe tu nombre:
  <input type="text" name="nombre" value="" />
  <br/>
  Escribe tu edad:
  <input type="text" name="edad" value="" />
  <br/>
  <input type="submit" value="Enviar" />
</form>
```

Formulario muy sencillo

Escribe tu nombre: Jordi Domenech
Escribe tu edad: 32
Enviar

Al enviar los datos del ejemplo ilustrado con el método GET, los datos aparecen adjuntos a la URL y codificados con el siguiente formato:

```
nombre=Jordi+Domenech&edad=32
```



Los datos de los dos campos del ejemplo aparecen detrás del signo de interrogación como pares de nombre=valor. El signo & separa los pares entre sí. Dentro de cada par, el valor de la variable se indica mediante un signo de igualdad (=). Los espacios en blanco se sustituyen por un signo más (+).

Los caracteres especiales de los datos que se introducen en los campos de un formulario se codifican al ser incluidos en la cadena de la URL. Por ejemplo, los caracteres reservados a la URL (?, &, =, #, %, /) se deben codificar si no se quiere que el servidor los interprete según sus significados especiales. Los caracteres no alfanuméricos se codifican convirtiéndolos a sus valores hexadecimales, con el signo de porcentaje (%) delante del número HEX. Por ejemplo, el signo interrogación (?) se codifica como %3F.

Por ejemplo, la siguiente entrada:

Formulario muy sencillo

Escribe tu nombre: María Fernández
Escribe tu edad: 33
Enviar

producirá esta codificación de los datos:

```
nombre=Mar%C3%ADa+Fern%C3%A1ndez&edad=33
```

El lenguaje PHP proporciona una manera sencilla de manejar formularios y permite crear programas interactivos que sepan responder según los datos que el usuario suministre. Al diseñar un formulario, deberemos indicar el fichero PHP que lo procesará, así como el método por el que se le pasará la información del formulario.

3. Leer datos de un formulario con PHP

Antes de ver cómo se manejan los datos desde el lado del servidor con PHP, vamos a construir un sencillo formulario de ejemplo sumando algunos de los conceptos y controles de formularios explicados hasta ahora.

Antes, copiaremos el siguiente bloque de código PHP en un archivo de texto y le pondremos el nombre formulario.php.

```
<?php
/* Muestra los datos enviados a través de un formulario HTML
a través del estado actual de PHP */
phpinfo( );
?>
```

Copiamos el siguiente código HTML/XHTML en un archivo de texto y le ponemos el nombre formulario.html.

```
<html>
<head><title>Ejemplo de formulario sencillo</title></head>
   <body>
   <h3>Formulario muy sencillo</h3>
     <form action="formulario.php" method="post">
     <input type="hidden" name="codigo_solicitud" value="X76345P09" />
     Escribe tu nombre:
     <input type="text" name="nombre" value="" />
     <br/>
     Escribe tu edad:
     <input type="text" name="edad" value="" />
     <br/>


     Escribe una contraseña:
     <input type="password" name="contrasena" value="" />
     <br/>
     Sexo:
     <input type="radio" name="sexo" value="hombre" checked="checked" />
       Hombre
     <input type="radio" name="sexo" value="mujer" /> Mujer
     <br/>
     Puestos de trabajo buscados:
     <input name="puesto_directivo" type="checkbox" value="directivo"/>
       Directivo
     <input name="puesto_tecnico" type="checkbox" value="tecnico"/>
       Técnico
     <input name="puesto_empleado" type="checkbox" value="empleado"/>
       Empleado
     <br/>
```

```
    <label for="so">Sistema operativo preferido</label>
    <select id="so" name="so">
      <option value="" selected="selected">- selecciona -</option>
      <option value="windows">Windows</option>
      <option value="mac">Mac</option>
      <option value="linux">Linux</option>
      <option value="otro">Otro</option>
    </select>
    <br/>
    <label for="observaciones">Observaciones</label> <br/>
    <textarea id="observaciones" name="observaciones" cols="40"
      rows="5"></textarea>
    <br/>
    <input type="submit" value="Enviar" />
    </form>
  </body>
</html>
```

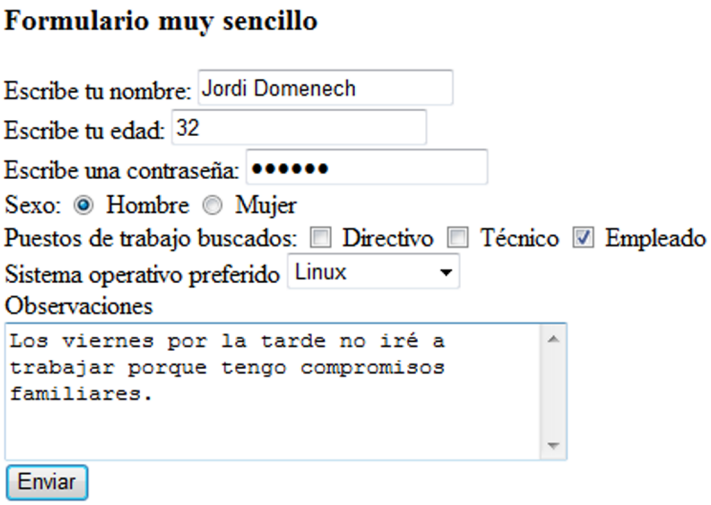
Guardamos los dos ficheros en el directorio raíz del servidor web de Apache:

```
C:\GMMD\apache\htdocs\
```

El método de envío es POST y action proporciona la URL de la aplicación que va a recibir y a procesar los datos, "formulario.php"; el atributo *enctype* no ha sido especificado y, por lo tanto, el formulario lanzará sus datos con la codificación estándar de la URL.

```
<form action="formulario.php" method="post">
```

Podemos visualizar el formulario en la pantalla del navegador introduciendo http://localhost/form.html. A continuación, rellenamos los datos del formulario tal y como aparecen colocados en la siguiente ilustración:



Al pulsar el botón “Enviar”, el contenido del formulario es enviado a la página que indicamos en el atributo *action* de la etiqueta <form>. Las variables de dicho formulario pasan a estar automáticamente disponibles en el *script* gracias al servidor Apache y a PHP.

PHP Variables

Variable	Value
_REQUEST["codigo_solicitud"]	X76345P09
_REQUEST["nombre"]	Jordi Domenech
_REQUEST["edad"]	32
_REQUEST["contrasena"]	secret
_REQUEST["sexo"]	hombre
_REQUEST ["puesto_empleado"]	empleado
_REQUEST["so"]	linux
_REQUEST["observaciones"]	Los viernes por la tarde no iré a trabajar porque tengo compromisos familiares.
_REQUEST["__utmc"]	1
_REQUEST["__utma"]	1.1215844192.1231071363.1231404711.1231409323.20
_REQUEST["__utmz"]	1.1231071363.1.1.utmccn=(direct)|utmcsr=(direct)|utmcmd=(none)
_POST["codigo_solicitud"]	X76345P09
_POST["nombre"]	Jordi Domenech
_POST["edad"]	32
_POST["contrasena"]	secret
_POST["sexo"]	hombre
_POST["puesto_empleado"]	empleado
_POST["so"]	linux
_POST["observaciones"]	Los viernes por la tarde no iré a trabajar porque tengo compromisos familiares.

3.1. Variables predefinidas de PHP y su relación con los formularios

Los datos del formulario enviados con los métodos POST o GET se encuentran disponibles en las variables predefinidas de PHP. Estas variables se conocen también como superglobales porque siempre están disponibles en todos los ámbitos a lo largo de un *script*. PHP ofrece un conjunto de variables o matrices predefinidas que contienen variables del servidor web, el entorno y entradas del usuario. Las variables superglobales de PHP son: $GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST, $_ENV.

Las variables superglobales que nos interesan analizar al procesar un formulario son: $_SERVER, $_REQUEST, $_GET, $_POST. Todas ellas son del tipo *array* asociativo:

Array asociativo

Un *array* asociativo es un tipo abstracto de dato formado por una colección de claves únicas y una colección de valores, con una asociación uno a uno. En vez de estar organizados con índices numéricos en función de su posición dentro del *array* (*arrays* escalares), lo están por claves (*key*) que pueden ser cadenas de texto. Con ello, tenemos la posibilidad de poner cualquier tipo de dato para especificar el índice.

- $_SERVER: Información del servidor y el entorno de ejecución.
- $_REQUEST: Variables de petición HTTP. Un valor tipo *array* asociativo que contiene de manera predeterminada los datos de $_GET, $_POST, y $_COOKIE.
- $_GET: Variables HTTP GET.
- $_POST: Variables HTTP POST.

A continuación vamos a modificar el *script* formulario.php para recoger y mostrar en la pantalla del navegador los datos enviados desde el formulario HTML:

```
<?php
   echo "Método HTTP empleado al enviar el formulario fue:
   ",$HTTP_SERVER_VARS['REQUEST_METHOD'],"<br>";
   echo "Código de solicitud: ".
   $HTTP_POST_VARS['codigo_solicitud']."<br/>";
   echo "Nombre: ". $HTTP_POST_VARS['nombre'],"<br/>";
   echo "Edad: ". $HTTP_POST_VARS['edad'],"<br/>";
   echo "Contraseña: ". $HTTP_POST_VARS['contrasena'],"<br/>";
   echo "Sexo: ". $HTTP_POST_VARS['sexo'],"<br/>";
   echo "Puesto de trabajo buscado: ".
   $HTTP_POST_VARS['puesto_empleado'],"<br/>";
   echo "Sistema operativo preferido: ". $HTTP_POST_VARS['so'],"<br/>";
   echo "Observaciones: ". $HTTP_POST_VARS['observaciones'],"<br/>";
?>
```

El resultado en pantalla será:

```
Método HTTP empleado al enviar el formulario fue: POST
Código de solicitud: X76345P09
Nombre: Jordi Domenech
Edad: 32
Contraseña: secret
Sexo: hombre
Puesto de trabajo buscado: empleado
Sistema operativo preferido: Linux
Observaciones: Los viernes por la tarde no iré a trabajar
porque tengo compromisos familiares.
```

Los *arrays* asociativos globales $_GET, $_POST y $_REQUEST se pueden utilizar conjuntamente con las funciones de PHP, each() y list(), para recorrer la matriz, y obtener así los nombres de cada variable del formulario con su valor

asignado. Estas variables se comportan en realidad como un *array* asociativo. Para cada elemento del formulario se crea un vínculo entre la clave y el valor asociado.

Se pueden utilizar conjuntamente las funciones de PHP, each() y list(), para recorrer la matriz $ _POST, y obtener así los nombres de cada variable del formulario con su valor asignado.

A continuación vamos a reescribir el *script* formulario.php para recorrer la matriz $_POST y visualizar las variables pasadas con el método POST. Haremos el recorrido mediante la función each():

```
<?php
  echo "<p>Los valores enviados con el método POST:<p/>";
    while(list($nombre,$valor) = each($_POST)){
  echo $nombre." = ".$valor."<br/>"; }
?>
```

La función each() va recuperando valores del array $_POST y avanza el puntero. La función list() asigna en cada operación los valores del *array* a la lista de variables ($nombre, $valor) pasadas como argumento.

El resultado en pantalla será:

```
Los valores enviados con el método POST:
codigo_solicitud = X76345P09
nombre = Jordi Domenech
edad = 32
contrasena = secret
sexo = hombre
puesto_empleado = empleado
so = Linux
observaciones = Los viernes por la tarde no iré a trabajar
porque tengo compromisos familiares.
```

Desarrollo web con PHP y MySQL

Ejemplos prácticos

Piero Berni Millet

PID_00155711

Universitat Oberta de Catalunya
www.uoc.edu

Índice

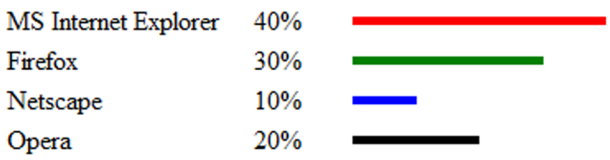
1. Gráfico de barras con PHP y HTML

Combinando PHP con HTML podemos generar fácilmente gráficas estadísticas de barras para datos porcentuales sin necesidad de recurrir a un *plug-in* extra de Flash o a una imagen GIF pregenerada dinámicamente con una librería específica de PHP.

Vamos a ver cómo se construye un gráfico de barras para una encuesta sobre navegadores de Internet, valiéndonos simplemente de las propiedades de las tablas HTML para emular dicho gráfico.

Navegadores	% de votos	Color barra gráfica
MS Internet Explorer	40	*red*
Firefox	30	*green*
Netscape	10	*blue*
Opera	20	*black*

¿Cual es tu navegador de Internet favorito?



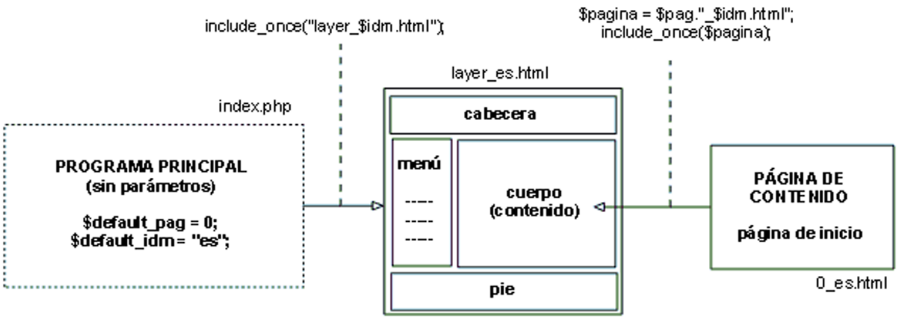
Programa: encuesta.php
Llamada: http://localhost/encuesta.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
       <title>Gráfico de barras dinámico con PHP y MYSQL</title>
</head>
<body>
<?php
// Array multidimensional con los datos de la encuesta y el color de la barra:
// Navegador, % votos, color barra gráfica
 $datos = array(
    array( "MS Internet Explorer", 40, "red" ),
    array( "Firefox", 30, "green" ),
    array( "Netscape", 10, "blue" ),
    array( "Opera", 20, "black" )
 );
?>
<h3>¿Cuál es tu navegador de Internet favorito?</h3>
<table width="600" cellspacing="0" cellpadding="2">
       <?php
        # Leo datos del array
        foreach( $datos as $d ) {
            $navegador = $d[0];
            $porcentual = $d[1];
            $color = $d[2];
       ?>
       <tr>
             <td width="25%"><?=$navegador?></td>
             <td width="10%"><?=$porcentual?>%</td>
             <td>
                   <!--Barras gráficas -->
                   <table width="<?=$porcentual?>%" bgcolor="<?=$color?>">
                         <tr><td> </td></tr>
                   </table>
             </td>
       <tr>

       <?php } ?>
</table>
</body>
</html>
```

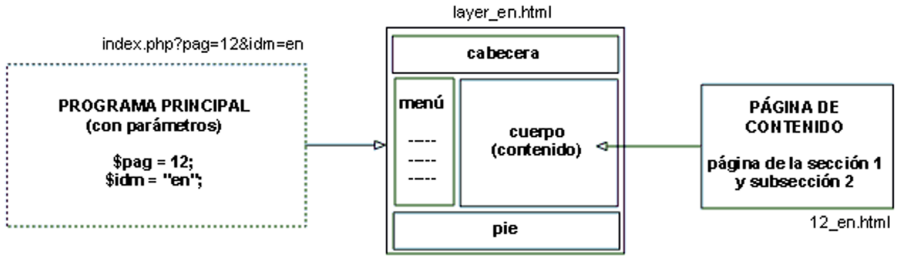
2. Web dinámica multilenguaje

Combinando PHP con HTML podemos crear sitios web dinámicos multilenguaje con un alto grado de productividad. El sitio web de este ejemplo está pensado para soportar tres o más idiomas: español (es), inglés (en) y catalán (cat). Los contenidos se encuentran desvinculados de la plantilla web que controla el diseño, la presentación y la navegación, lo que simplifica el mantenimiento del sitio cuando se modifican o se añaden nuevos temas.

Estructura de la web multilenguaje



Abro la página de inicio del sitio (home page)



Abro desde el menú una página de contenido para el idioma seleccionado

El programa principal (index.php)

Se accede al sitio mediante el *script* principal **index.php**. El recorrido por las diferentes páginas de un idioma determinado se controla con los parámetros 'pag' e 'idm' que se pasan en la URL.

Por ejemplo:

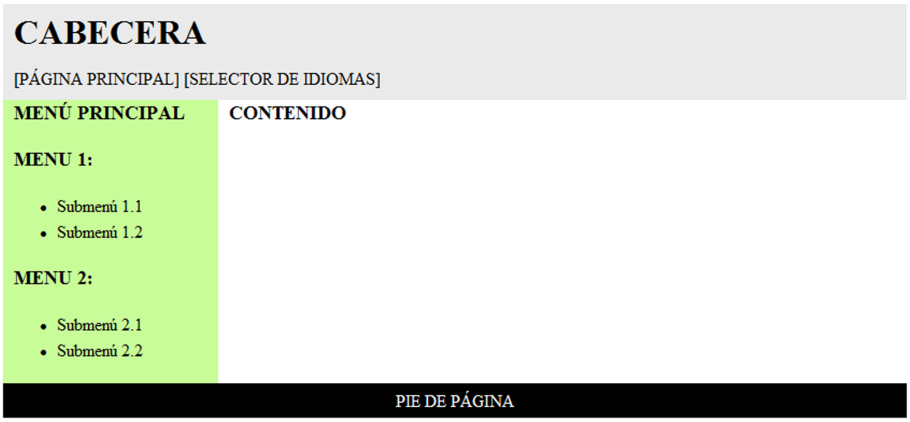
- http://host/index.php Si no se incluyen los parámetros en la URL se carga la página principal del sitio en el idioma predeterminado (ver index.php).

- http://host/index.php?pag=01&idm=es En este caso, los parámetros indicados cargarán la página de contenido 01 (01_es.html) para el idioma es (español).

El código del programa principal es el siguiente:

```
<?php
        // CARGO PARÀMETROS PASADO POR EL URL
        # Por ejemplo: http://host/index.php?pag=01&idm=es

        # Página del sitio web: pag=[número de la página a abrir]
        # Si no se ha especificado una página de contenido
        # cargo el contenido de la página inicial (0_$idm.html)
        $default_pag = 0;
        $pag = isset($_GET['pag']) && !empty($_GET['pag']) ? $_GET['pag'] : $default_pag;

        # Idioma del sitio web: idm=[es,en,cat]
        $idm = isset($_GET['idm']) && !empty($_GET['idm']) ? $_GET['idm'] : "";

        # Si no se ha pasado el idioma, o el indicado no está entre
        # los predeterminados, selecciono el idioma por defecto (es)
        $default_idm = "es";
        $idm = ($idm == "es" || $idm == "en" || $idm == "cat") ? $idm : $default_idm ;

        # Abro la plantilla para el idioma seleccionado
        include_once("layer_$idm.html");
?>
```

La plantilla (layer_$idm.html)



La plantilla del sitio se estructura en cuatro secciones:

- **Cabecera**. Espacio reservado para el título del sitio con su *banner*. En la parte inferior izquierda se colocan el enlace de "ir a la página de inicio", más los tres enlaces que permiten seleccionar un idioma.

```
<a href="index.php?pag=0&idm=es">Página de inicio</a>   
<a href="index.php?pag=0&idm=es">Español</a> |
<a href="index.php?pag=0&idm=en">Inglés</a> |
<a href="index.php?pag=0&idm=cat">Catalán</a> |
```

- **Menú principal**. La columna de la izquierda está reservada para el menú de navegación, que se estructura en dos niveles (sección y subsecciones). Haciendo clic en los enlaces del menú se abren las páginas de contenido en el espacio central de la plantilla.

```
<h3>MENU 1:</h3>
<ul>
        <li><a href="index.php?pag=11&idm=es">Submenú 1.1</a></li>
        <li><a href="index.php?pag=12&idm=es">Submenú 1.2</a></li>
</ul>
<h3>MENU 2:</h3>
<ul>
        <li><a href="index.php?pag=21&idm=es">Submenú 2.1</a></li>
        <li><a href="index.php?pag=21&idm=es">Submenú 2.2</a></li>
</ul>
```

- **Contenido**. Aquí se incrustan dinámicamente los contenidos del sito. Por ejemplo, los ficheros del sitio en español tienen la siguiente nomenclatura: 0_es.html (página de inicio o de bienvenida), 21_es.html (la página de contenido del la sección 2.1 del menú 2.

- **Pie de página**. Espacio reservado para los datos de contacto del sitio (dirección, correo electrónico).

Puesto que el sitio web soporta tres idiomas, la plantilla deberá triplicarse para cada uno de ellos con las nomenclaturas: layer_es.html, layer_en.html, y layer_cat.html

Incrustar los contenidos en el cuerpo de la plantilla

El mecanismo que controla las páginas de contenido y las incrusta en la sección Contenido se basa un pequeño *script* de PHP que debe residir en esa posición de la plantilla del sito. Este programa comprueba si existe la página de contenido en el sistema de archivos del servidor web. En caso de no existir, por ejemplo, el contenido 2.1_es.html, se muestra un mensaje de error.

```
<p><b>CONTENIDO</b></p>
<?php

# Ruta absoluta al directorio del sitio en el sistema de ficheros
$path = "/var/www/web/";
# Nombre del fichero de contenido
$pagina = $pag."_$idm.html";

# TEST: ¿existe fichero de contenido?
$file = $path.$pagina;
if(file_exists($file))
     include_once($pagina);
    else
     echo "El archivo de contenido $pagina no existe";
?>
</div>
```

El código completo de la plantilla HTML es el siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Web dinàmica multillenguatge</title>
   <style type="text/css">


      body{
         margin:0;
         padding:0;
         line-height: 1.5em;
      }


      b{font-size: 110%;}
      em{color: red;}


      #maincontainer{
         width: 840px; /*Width of main container*/
         margin: 0 auto; /*Center container on page*/
      }


      #topsection{
         background: #EAEAEA;
         height: 90px; /*Height of top section*/
      }


      #topsection h1{
         margin: 0;
         padding-top: 15px;
      }


      #contentwrapper{
         float: left;
         width: 100%;
      }
```

```
      #contentcolumn{
         margin-left: 200px; /*Set left margin to LeftColumnWidth*/
      }

      #leftcolumn{
         float: left;
         width: 200px; /*Width of left column*/
         margin-left: -840px;
         /*Set left margin to -(MainContainerWidth)*/
         background: #C8FC98;
      }

      #footer{
         clear: left;
         width: 100%;
         background: black;
         color: #FFF;
         text-align: center;
         padding: 4px 0;
      }

      #footer a{
         color: #FFFF80;
      }

      .innertube{
         margin: 10px; /*Margins for inner DIV inside each column
         (to provide padding)*/
         margin-top: 0;
      }

   </style>

</head>

<body>
   <div id="maincontainer">

      <div id="topsection">
         <div class="innertube"><h1>CABECERA</h1>
            <p>[PÁGINA PRINCIPAL] [SELECTOR DE IDIOMAS]</p>
         </div>
      </div>

      <div id="contentwrapper">
         <div id="contentcolumn">
            <div class="innertube">
               <p><b>CONTENIDO</b></p>
            </div>
         </div>
      </div>
```

```
      <div id="leftcolumn">
         <div class="innertube">
            <p><b>MENÚ PRINCIPAL</b></p>
            <h3>MENU 1:</h3>
            <ul>
               <li>Submenú 1.1</li>
               <li>Submenú 1.2</li>
            </ul>
            <h3>MENU 2:</h3>
            <ul>
               <li>Submenú 2.1</li>
               <li>Submenú 2.2</li>
            </ul>

         </div>
      </div>


      <div id="footer">
         PIE DE PÀGINA

      </div>
   </div>
</body>
</html>
```

3. Formulario para enviar los datos a una cuenta Gmail

Generalmente, cuando se rellenan los campos de un formulario HTML y se envían al servidor mediante un *script* de PHP, los datos se registran en una base de datos. En otros casos, lo que se suele hacer es, tras procesarlos desde el lado del servidor, enviarlos a una cuenta de correo electrónico. Típicamente para el envío de correo con PHP se utiliza la función mail(), pero esta función tiene varias limitaciones, por ejemplo, hay que tener una cuenta de usuario registrada en el servidor que aloja la web y la función mail() no soporta el envío de adjuntos.

Afortunadamente, los datos recogidos mediante un formulario HTML pueden ser enviados a su destinatario mediante un servicio de correo externo tipo Gmail. Para ello, es necesario utilizar una clase de PHP llamada PHPMailer que facilita el envío de correo, añadiendo funcionalidad en el envío de correos con adjuntos, en formato HTML y con diferentes codificaciones, soporte para imágenes embebidas, *headers* personalizados y además funciona con múltiples servidores de correo, entre los que se encuentra Gmail.

Para llevar a cabo este ejemplo práctico se necesita tener una cuenta de correo Gmail y bajarse la clase phpmailer para php5/6 de la página de sus desarrolladores.

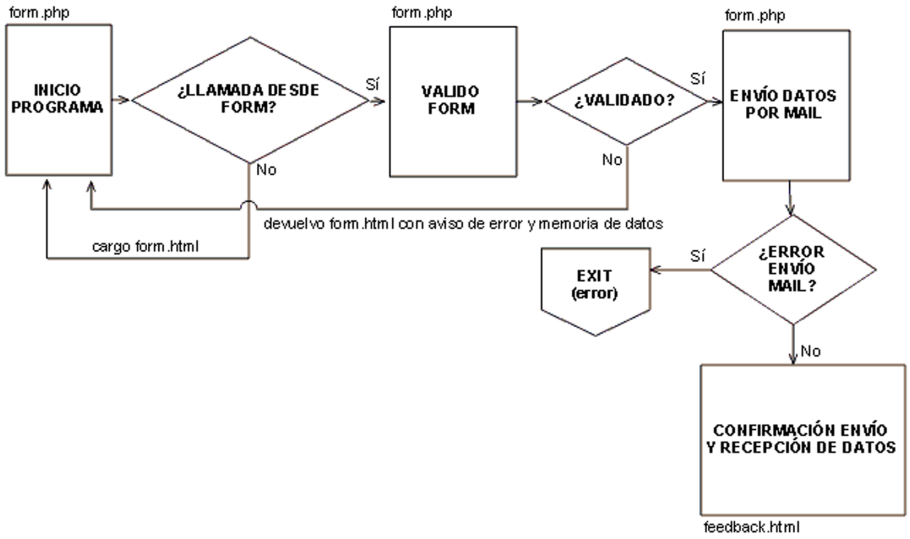
El código de la aplicación se distribuye en tres ficheros:

- **form.php**. El *script* de PHP principal que carga el formulario HTML, valida sus datos y los envía por correo a su destinatario mediante una cuenta Gmail.

- **form.html**. El archivo con el formulario HTML.

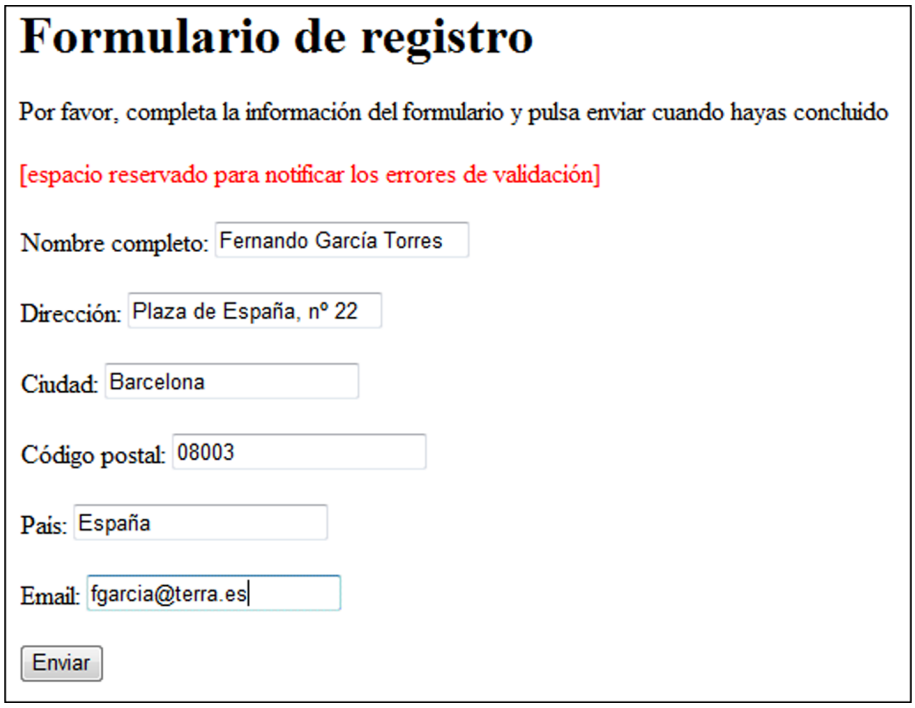
- **feedback.php**. Una sencilla página web con la confirmación de envío y recepción de los datos.

Lógica de aplicación:



Inicio del programa

El programa se arranca desde el *script* **form.php**. Si no existe una llamada desde el formulario (si no se ha pulsado el botón "Enviar"), se carga el formulario vacío (**form.htm**) por primera vez.



En el ejemplo particular, la validación del formulario tendrá en cuenta las siguientes reglas:

- Todos los campos son de dato obligatorio (no pueden estar vacíos).
- El valor de código postal debe ser numérico (desde 00001 a 52999).
- El valor del correo electrónico debe ser sintácticamente correcto.

Si no se cumple la validación, se devuelve el formulario con la notificación del error y la memoria de datos.

Si la validación de los datos del formulario es correcta, éstos se envían por correo mediante la clase phpmailer.

Finalmente, el usuario recibe la confirmación del envío.

Fichero principal (form.php)

```
<?php

// LLAMADA DESDE FORMULARIO
if(isset($_POST['submit']) ) {

# Preparo datos pasados desde el form
$FormData = array();

# Limpio espacios en blanco y caracteres de escape
foreach($_POST as $key=>$value) {
   $FormData[$key] = stripslashes(trim($value));
}
```

```
// VALIDO FORM
# Códigos de error
# 0 - sin error
# 1 - datos requeridos
# 2 - código postal incorrecto
# 3 - mail sintaxis incorrecta

$error_frm = 0;

# TEST 1: datos requeridos
if(empty($FormData['nombre']) || empty($FormData['direccion']) ||
empty($FormData['ciudad']) || empty($FormData['cod_postal'])
   || empty($FormData['pais']) ||  empty($FormData['email']) ) {
   $error_frm = 1;
}


# TEST 2: valido código postal con una expresión regular
# Los códigos postales en España van desde 00001 al 52999
if(!$error_frm && !preg_match("/^([1-9]{2}|[0-9][1-9]|[1-9][0-9])
[0-9]{3}$/", $FormData['cod_postal']) ) {
   $error_frm = 2;
}

# TEST3: valido email con una expresión regular
if(!$error_frm && !preg_match("/^[A-z0-9\._-]+@[A-z0-9][A-z0-9-]*
(\.[A-z0-9_-]+)*\.([A-z]{2,6})$/", $FormData['email']) ) {
   $error_frm = 3;
}
```

```
// PASADOS TODOS LOS TESTS ---
if(!$error_frm){

   // ENVÍO DATOS MEDIANTE UNA CUENTA GMAIL

   #  Cargo clase phpmailer
   include_once('phpMailer_v2.3/class.phpmailer.php');
   # Creo una nueva instancia
   $mail = new PHPMailer();

   # Servicio Gmail
   # Le digo que voy a usar SMTP (el modo de envío)
   $mail->IsSMTP();
   # GMail usa SSL/TLS como protocolo de comunicación/autenticación
   $mail->Host = 'ssl://smtp.gmail.com';
   # El Puerto de comunicación del servidor SMTP de Gmail
   $mail->Port = 465;
   # Activo la autentificación SMTP
   $mail->SMTPAuth = true;
   # Mi nombre de usuario en Gmail.
   $mail->Username = 'usuario@gmail.com';
   # Mi contraseña en Gmail
   $mail->Password = '*******';

   # Propiedades del Mensaje
   # Codificación de caracteres
   $mail->CharSet = "UTF-8";
   # mail y nombre del remitente
   $mail->From = $FormData['email'];
   $mail->FromName = $FormData['nombre'];
   # Mail del destinatario
   $mail->AddAddress('usuario@gmail.com');
   # Puedo añadir otro destinatario más
   $mail->AddAddress('usuario@uoc.edu');

   # El tema del mensaje
   $mail->Subject = "Formulario de registro";
   # Mail en formato HTML
   $mail->IsHTML(true);
   # Caracteres por línea
   $mail->WordWrap = 50;
   # Cuerpo del mensaje

$cuerpo = "
FORMULARIO DE REGISTRO \n
Fecha de registro:".date(d."-".m."-".Y)."\n
-----------------------------------------------\n
Nombre: ".$FormData['nombre']."\n
Dirección: ".$FormData['direccion']."\n
Ciudad: ".$FormData['ciudad']."\n
Código Postal: ".$FormData['cod_postal']."\n
País: ".$FormData['pais']."\n
Email: ".$FormData['email']."\n
-----------------------------------------------\n
";
   # Cargo el contenido del mensaje con los saltos de linea convertidos en <br />
   $mail->Body=nl2br(htmlentities($cuerpo, ENT_COMPAT));
   # También cargo el contenido del mensaje con formato de texto plano
   $mail->AltBody = $cuerpo;
```

```
   // ENVÍO MENSAJE
   # Send Mail
   if(!$mail->Send()) {
   echo "El mensaje no puede ser enviado. <p>";
      echo "phpMailer Error: " . $mail->ErrorInfo;
         exit;
   }


   // CONFIRMACIÓN DE ENVÍO Y RECEPCIÓN DE DATOS ---
   include_once('feedback.html');
   exit;

   }
   else {
      # Devuelvo form con memoria de datos y el aviso de error de validación
      include_once('form.html');
   }

} else
   include_once('form.html');

?>
```

Formulario HTML (form.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Formulario para enviar los datos a una cuenta Gmail</title>
</head>


<body>
  <Hl>Formulario de registro</Hl>
      <p>Por favor, completa la información del formulario y pulsa enviar
      cuando hayas concluido</p>
  <!-- espacio reservado para notificar los errores de validación -->
      <?php if($error_frm == 1) { ?>
            <p style="color: red;">
              ERROR: todos los campos son de dato requerido
            </p>
      <?php } elseif($error_frm == 2) { ?>
       <p style="color: red;">ERROR: el código postal no es correcto</p>
      <?php } elseif($error_frm == 3) { ?>
       <p style="color: red;">ERROR: el email no es correcto</p>
      <?php } ?>


<form action="form.php" method="post">
   <p>Nombre completo:
      <label>
<input type="text" name="nombre" id="nombre" value="<?=$FormData['nombre']?>" />
      </label>
   </p>
```

```
   <p>Dirección:
      <label>
         <input type="text" name="direccion" id="direccion"
         value="<?=$FormData['direccion']?>" />
      </label>
   </p>

   <p>Ciudad:
      <label>
         <input type="text" name="ciudad" id="ciudad" value="<?=$FormData['ciudad']?>" />
      </label>
   </p>

   <p>Código postal:
      <label>
         <input type="text" name="cod_postal" id="cod_postal"
         value="<?=$FormData['cod_postal']?>" />
      </label>
   </p>

   <p>País:
      <label>
         <input type="text" name="pais" id="pais" value="<?=$FormData['pais']?>" />
      </label>
   </p>

   <p>Email:
   <label>
         <input type="text" name="email" id="email" value="<?=$FormData['email']?>" />
      </label>
   </p>
   <p>
      <input type="submit" name="submit" id="submit" value="Enviar" />
   </p>
</form>

</body>
</html>
```

Confirmación de envío (feedback.html)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
       <title>Documento sin t&iacute;tulo</title>
</head>
```

```
<body>
	<h1>Formulario de registro</h1>
	<hr />
	<p>Apreciado <?=$FormData['nombre']?></p>
	<p>Los datos de su solicitud se han recibido correctamente.</p>
	<p>
		La matrícula se formalizará una vez que se compruebe el pago de su
		inscripción.
	</p>
	<p>
		Le tendremos informado mediante su cuenta de correo <a
		href="mailto:<?=$FormData['email']?>"><?=$FormData['email']?></a>
	</p>
</body>
</html>
```

4. El formulario anterior con código de seguridad anti *spambots* (captcha)

Captcha es el acrónimo de *Completely Automated Public Turing test to tell Computers and Humans Apart* (prueba de Turing pública y automática para diferenciar máquinas y humanos).

Los captchas son utilizados para evitar que robots, también llamados *spambots*, puedan utilizar ciertos servicios. Por ejemplo, para que no puedan participar en encuestas, registrarse para usar cuentas de correo electrónico (o su uso para envío de correo basura) o, más recientemente, para evitar que correo basura pueda ser enviado por un robot (el remitente debe pasar el test antes de que se entregue al destinatario).

La idea básica para crear un captcha es generar primero un texto aleatorio; luego, basado en este texto, generamos una imagen que se muestra al usuario, y, finalmente, se compara el texto ingresado por el usuario con la palabra aleatoria generada.

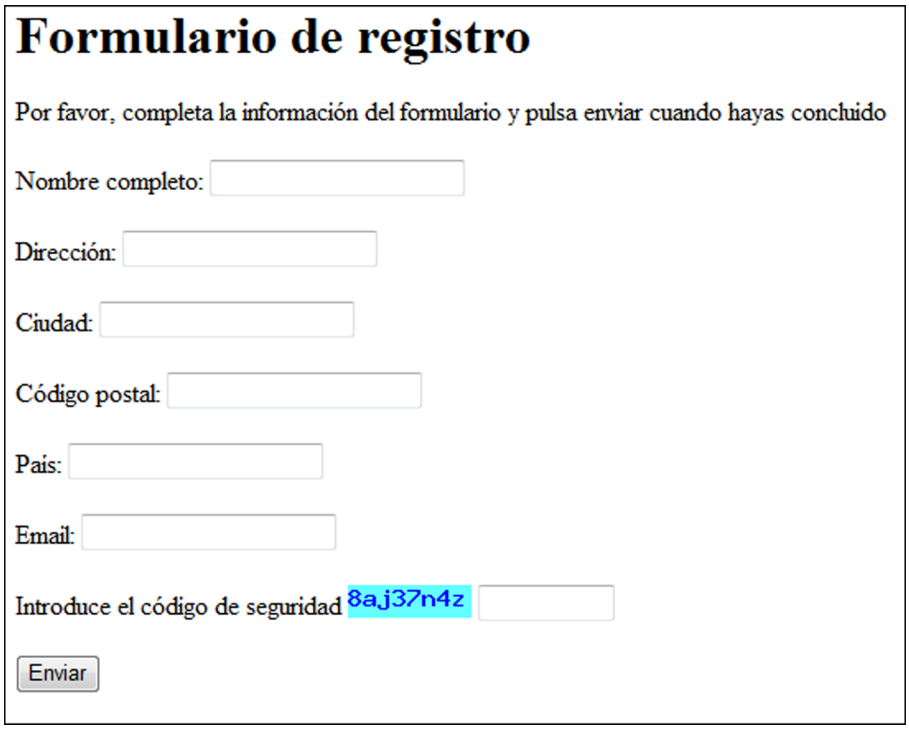
Generar el captcha (captcha.php)

El primer paso consiste en crear una imagen con un texto aleatorio mediante el *script* **captcha.php**, el cual creará automáticamente el texto y salvará su valor en la variable de sesión $_SESSION['captcha'].

Lo siguiente es mostrar la imagen generada en el formulario HTML para que el usuario pueda leerlo e ingresar el texto para ser verificado. Para ello, editamos el archivo **form.html** y añadimos al final las siguientes líneas de código.

```
<p>
      Introduzco el código de seguridad
      <img src="captcha.php" alt="" width="75" height="20" vspace="1" align="top" />
      <input name="captcha" type="text" id="captcha" size="8" maxlength="10" />
</p>
```

El formulario HTML con el captcha tendrá el siguiente aspecto:



El paso siguiente consiste en verificar el texto ingresado en el formulario y compararlo con la variable de sesión que contiene el texto generado aleatoriamente. Para ello, será necesario hacer algunas modificaciones en el programa principal **form.php**.

Al comienzo del programa inicializamos la sesión para poder leer el dato:

```
<?php
        session_start();
        // LLAMADA DESDE FORMULARIO
        if(isset($_POST['submit']) ) {
        ...
        ...
?>
```

En la sección de validación del formulario añadimos un nuevo test con el que comparamos el texto ingresado con el texto que tenemos en la variable de sesión.

```
...
...
// VALIDO FORM
# Códigos de error
# 0 - sin error
# 1 - datos requeridos
# 2 - código postal incorrecto
# 3 - mail sintaxis incorrecta
# 4 - código de seguridad incorrecto
...
...
# TEST4: valido captcha
 if(!$error_frm && $_SESSION['captcha'] != $FormData['captcha']) {
    $error_frm = 4;
 }
...
...
```

Finalmente, en la parte inicial de **form.html** añadimos el mensaje de error de validación del captcha.

```
<?php } elseif($error_frm == 4) { ?>
       <p style="color: red;">ERROR: código de seguridad incorrecto</p>
<?php } ?>
```

Fichero generador del captcha (captcha.php)

```
<?php
// Crear CAPTCHA con PHP

# Creo una sesión con PHP
session_start();
# Salvo en una variable de sesión el valor del captcha
$_SESSION['captcha'] = randomText(8);


# Genero la imagen con las funciones de la librería GD de PHP
$img = imagecreate(75, 20);
$fondo = imagecolorallocate($img, 100, 255, 255);
$color_texto = imagecolorallocate($img, 0, 0, 255);
imagestring($img, 5, 0, 0, $_SESSION['captcha'], $color_texto);


# Envío la imagen tras las cabeceras HTTP
header("Content-type: image/png");
imagepng($img);
exit;
```

```
# Genero el valor del captcha
function randomText($length) {
    $pattern = "1234567890abcdefghijklmnopqrstuvwxyz";
    for($i=0;$i<$length;$i++) {
      $key .= $pattern{rand(0,35)};
    }
    return $key;
}
?>
```

5. Geolocalización con GeoIp y Google Maps

El geoposicionamiento por IP tiene interesantes beneficios en aplicaciones de blog, Chat, foros, seguridad, *urban mapping*, etc. Por ejemplo, para combatir prácticas indeseables como el *phishing* en el desarrollo de aplicaciones distribuidas en entornos abiertos y heterogéneos.

¿Cómo se realiza el geoposicionamiento de visitantes?

1) Se obtiene la IP del visitante.

2) Se averigua la *IP Address Look Up* de esa IP para saber su situación. Para esto utilizamos la API de un servicio de *Geolocation by IP Address* para acceder a la información geográfica de sus bases de datos.

3) Se genera el mapa con la API de Google Maps, geoposicionando dentro del mapa la información obtenida con el paso anterior.

Existen en Internet multitud de servicios de *Geolocation by IP Address*, comerciales y gratuitos, con los que es posible averiguar la *IP Address Look Up* de los usuarios con información geográfica del tipo: país, región, ciudad, latitud, longitud, ZIP *code*, *time zone*, velocidad de conexión, ISP, etc.

Productos comerciales realizados con el respaldo de bases de datos de pago son, por ejemplo:

- IP2Location™
- Geobytes
- MaxMind, GeoIP

Productos hechos con el respaldo de bases de datos de acceso gratuito son:

- HostIP.info
- NetGeo - The Internet Geographic Database

Tanto los productos comerciales como gratuitos ponen a disposición de los usuarios una API para localizar direcciones IP con información geográfica compatible para diferentes lenguajes. Por ejemplo, MaxMind ofrece su *Open Source binary API* para los lenguajes C, Perl, Apache, Java, Python, Ruby, etc. En el caso de hostip.info hay una API para localizar direcciones IP con información geográfica que nos permite hacer, entre otras cosas, saludar a los visitantes con la bandera del país desde donde se conectan.

Casi todos estos productos tienen en su página web un localizador *on-line*. Los hay que incluso posicionan el resultado de nuestra búsqueda en Google Maps; por ejemplo:

- Geo IP Tool
- Ip-adress
- Seomoz

Por norma general, la información de los productos con respaldo de bases de datos de acceso gratuito no es excesivamente completa. Muchas IP no quedan localizadas y de algunas ciudades no disponen de la latitud y la longitud (con lo que hay que excluirlas y no se pueden situar en el mapa).

Los productos con respaldo de bases de datos de pago requieren una licencia de uso que debe renovarse cada cierto tiempo. Las bases de datos licenciadas se actualizan en línea en nuestro servidor tras su instalación y una vez registrados. Por ejemplo, MaxMind ofrece para su producto GeoIP, ocho bases de datos de diferentes propósitos. Afortunadamente para los desarrolladores web como nosotros, MaxMind tiene dos versiones gratuitas de sus bases de datos GeoIP Country y GeoIP City, aunque, como es lógico, hay limitaciones en el volumen de información y sobre el grado de exactitud de los datos de país y ciudad.

Estas dos bases de datos se pueden descargar de Maxmind, junto con el código fuente o binario de GeoIP, y la API de PHP, para así instalar la aplicación en un servidor de Internet.

5.1. Acceso a la base de datos GeoIP para resolver la *IP Address Look Up* de una IP

Una vez instalada la base de datos GeoIP y la extensión de GeoIP de PHP, ya es posible obtener la *IP Address Look Up* con la ayuda del siguiente *script* (**geoip.php**), que recibirá la IP que debe resolver como parámetro de la URL: http://host/geoip.php?ip=213.73.38.93.

Este *script* devolverá un texto simple con la información de la consulta encadenada con un formato especial, tal y como se ve en el siguiente ejemplo:

```
VERSION:GEO-106FREE 20080101 Build 1 Copyright (c) 2007 Max-
Mind LLC
&COUNTRY:Spain&CITY:Barcelona&LAT:41.3833007812&LONG:2.18330001831
```

Programa: geoip.php

```
<?php

// Compruebo que se ha pasado el parámetro por la URL
if(!isset($_GET['ip']) || empty($_GET['ip'])) {
    echo "ERROR: no se pasó la IP en el URL";
    exit;
}

// Compruebo con una expresión regular que la IP es correcta
if(!preg_match( "/^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$/", $_GET['ip']) ) {
    echo "ERROR: la dirección IP no es correcta";

}

// GeoIP IP Address Look Up
$GeoIpData = array();

# Información sobre la versión instalada de la base de datos GeoIP
$GeoIpData['VERSION'] = geoip_database_info(GEOIP_COUNTRY_EDITION)."\n";
# País
$GeoIpData['COUNTRY'] = geoip_country_name_by_name($_GET['ip']);
# Ciudad, latitud, longitud
$datos = geoip_record_by_name($_GET['ip']);
$GeoIpData['CITY'] = $datos['city'];
$GeoIpData['LAT'] = $datos['latitude'];
$GeoIpData['LONG'] = $datos['longitude'];

# Junto los elementos del array en una cadena de texto con formato (nombre:valor)
$elementos = array();
while (list($key, $value) = each($GeoIpData))  {
    $elementos[] = $key.":".$value;
}

# Encadeno los elementos con el signo &
$resultado = implode("&", $elementos);

# Devuelvo el resultado como flujo de datos tras una cabecera HTTP
header('Content-Type: text/plain');
echo $resultado;
exit;

?>
```

5.2. Geoposicionar en un mapa de Google la procedencia geográfica del visitante

La aplicación **geolocacion.php** muestra en un mapa de Google Maps la ubicación geográfica de la persona que visita la página web. Este programa le pasa a **geoip.php** una IP de la que queremos obtener información geográfica, y este último le retorna los datos de la consulta de la base de datos GeoIP. La IP que se debe geolocalizar se puede obtener automáticamente por medio de la variable $_SERVER['REMOTE_ADDR'], o introducir manualmente como parámetro de la URL: http://host/geolocacion.php?ip=213.73.38.93

Para poder utilizar el servicio de Google Maps será necesario darse de alta:

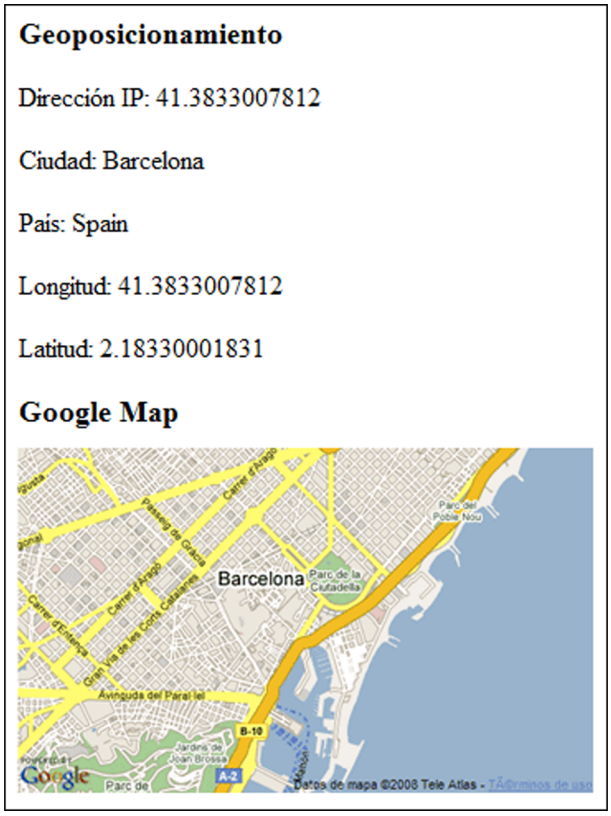
1) El registro se efectúa desde la página Google Maps y es gratuito.

2) Habrá que especificar la URL y el directorio de la página que usará el servicio de mapas de Google.

3) El proceso termina con la obtención de una clave de API de Google Maps que deberemos añadir, más tarde, al objeto <script> que nos mostrará el mapa.

4) Las coordenadas geográficas LAT y LONG obtenidas con el *script* de PHP deberán ir colocadas en la siguiente línea de código de la API de Google Maps:

```
map.setCenter(new GLatLng(37.4419, -122.1419), 13);
```



Programa principal (geolocacion.php)

```
<?php

// IP pasada manualmente de la URL
# p.e. http://host/geolocation.php?ip=213.73.38.93
$ip = "";
if(isset($_GET['ip']) && !empty($_GET['ip']) ) {

   # Compruebo con una expresión regular que la IP es correcta
      if(preg_match("/^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$/", $_GET['ip']) )
      {
         $ip = $_GET['ip'];
   }
}
```

```
// IP automática obtenida automáticamente
if(empty($ip) )  {
   $ip =  $_SERVER['REMOTE_ADDR'];
}

// Consulto la base de datos para resolver la IP Address Look Up
$geoIpURL = "http://host/geoip.php?ip=$ip";

# Abro el contenido del URL para sólo lectura
if($GeoIpFP = fopen($geoIpURL,r)) {

   # Habilito el uso de búferes de salida
   ob_start();

   # Leo el flujo de datos y lo cargo en el búfer de salida
   fpassthru($GeoIpFP);

   # Cargo en una variable los contenidos del búfer de salida, sin borrarlo
   $GeoIpQueryData = ob_get_contents();

   # Deshabilito los búferes de salida
   ob_end_clean();

   # Cierro el apuntador al URL abierto
   fclose($GeoIpFP);

   # Cargo las parejas de datos en un array
   $parejas = array();
   $parejas = explode('&', $GeoIpQueryData);

   # Cargo los datos de cada pareja en un array asociativo
   $GeoIPData = array();
   for($i=0; $i < count($parejas); $i++) {
      list($k, $v) = split(":",$parejas[$i]);
      $GeoIPData["$k"] = $v;
   }

}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
   <title>Geoposicionamiento</title>

      <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAvqU-
      yHDNLzzic0kICu0UmBRA4kvN07udzPbkMjf7Gs6xqBycohQgma4w6SDtmk4tjEo4x0HuIpV96w"
      type="text/javascript"></script>

      <script type="text/javascript">
         //<![CDATA[
         function load() {
            if (GBrowserIsCompatible()) {
               var map = new GMap2(document.getElementById("map"));
               map.setCenter(new GLatLng(<?=$GeoIPData['LAT']?>, <?=$GeoIPData['LONG']?>),
               13);
            }
         }
         //]]>
      </script>
</head>
```

```
<body onload="load()" onunload="GUnload()">
   <h3>Geoposicionamiento</h3>
   <p>Dirección IP: <?=$GeoIPData['LAT']?></p>
   <p>Ciudad: <?=$GeoIPData['CITY']?></p>
   <p>País: <?=$GeoIPData['COUNTRY']?></p>
   <p>Longitud: <?=$GeoIPData['LAT']?></p>
   <p>Latitud: <?=$GeoIPData['LONG']?></p>
   <h3>Google Map</h3>
   <div id="map" style="width: 500px; height: 300px"></div>
</body>
</html>
```

Anexos

Piero Berni Millet
Dídac Gil de la Iglesia

PID_00155709

Universitat Oberta de Catalunya
www.uoc.edu

Índice

1. Anexo 1: Uso del servidor remoto de la UOC como SGBD y servidor de Internet

GMMD pone a disposición de los estudiantes matriculados en alguna de las asignaturas relacionadas con el Laboratorio de PHP y MySQL un espacio de trabajo en un servidor de la UOC. El objetivo de dicho servidor es permitir a los estudiantes publicar aplicaciones web accesibles desde Internet y publicar bases de datos de manera remota. Este servidor es una máquina Linux con un SGBD MySQL (ver. 5.x) y un servidor Apache 2.0 que permite ejecutar páginas web dinámicas escritas en PHP 5.x.

A lo largo de la asignatura realizaréis las actividades en dos entornos de trabajo. El **entorno de desarrollo** es vuestro sistema local, donde aprenderéis a instalar y configurar el SGBD en vuestro propio PC. Este entorno será vuestro taller de pruebas para diseñar, montar y probar las prácticas de la asignatura. El **entorno de producción** es la máquina Linux de la UOC. Podréis acceder a vuestra cuenta de usuario en el servidor de la UOC, donde publicaréis el trabajo definitivo. El entorno de producción está pensado para que adquiráis experiencia en el alojamiento de sitios web dinámicos desarrollados con este tipo de aplicaciones.

1.1. Vuestra información de usuario

Después del proceso de alta, recibiréis en vuestra cuenta personal del Campus un mensaje de correo que incluye las claves de acceso y el nombre del servidor remoto:

```
usuario: (el mismo que el del Campus)
contraseña: (una cadena alfanumérica de 8 caracteres)
host: comoras.uoc.edu (el nombre de la máquina)
```

1.2. Acceso a vuestra cuenta de usuario en Comoras

Cada alumno tiene en Comoras un espacio de disco donde subir los ficheros del sitio web que ha trabajado en el entorno de desarrollo. Este espacio está limitado a 20 Mb, aunque se puede ampliar si fuese conveniente. Cabe destacar que dicho servicio no caduca y, por tanto, podrá seguir usándose indefinidamente.

Para acceder al espacio remoto deberéis usar un programa cliente FTP (no hay posibilidad de conectarse mediante SSH).

Existen multitud de clientes web que se pueden usar para acceder al servidor. A continuación explicamos cómo hacerlo usando el programa Filezilla.

1.3. ¿Qué es Filezilla y dónde encontrarlo?

FileZilla es un cliente FTP, gratuito, libre (GNU), multiplataforma y de código abierto. Soporta FTP, SFTP y FTP sobre SSL. Está disponible para Windows, Linux y MacOS X, entre otros.

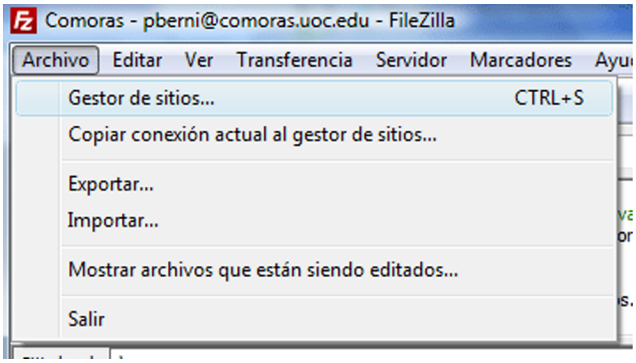
FileZilla en español

Se puede descargar la última versión del cliente FileZilla en español desde la versión española del sitio web de FileZilla.

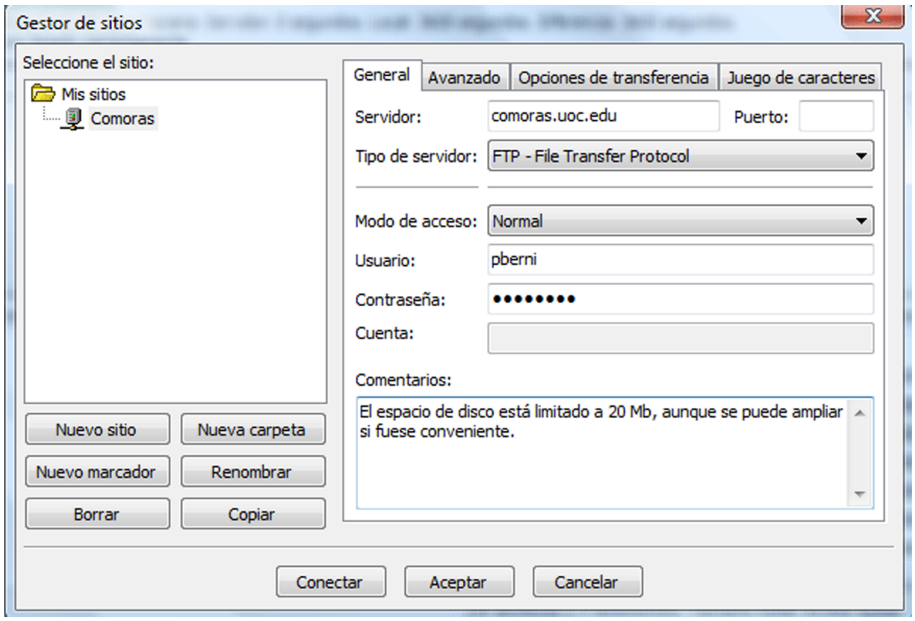
También encontraréis un mini manual de uso de FileZilla en español en Webcindario.

1.4. Acceso al espacio web usando Filezilla

Tras haber descargado el programa en el ordenador, procederemos a su instalación. A continuación vamos a configurar una nueva conexión para acceder a Comoras. Para ello vamos a Archivo -> Gestor de sitios.



En el Administrador de sitios pulsamos “Nuevo sitio”. Nos sale un icono y la opción de poner un texto; ponemos “Comoras”. En los detalles del Sitio hay que poner la dirección del servidor ftp; el puerto lo dejamos por defecto en 21. Ahora en “Modo de Acceso” elegimos “Normal” y se activa el campo “Usuario y Contraseña”, donde pondremos las claves que recibidas por correo.



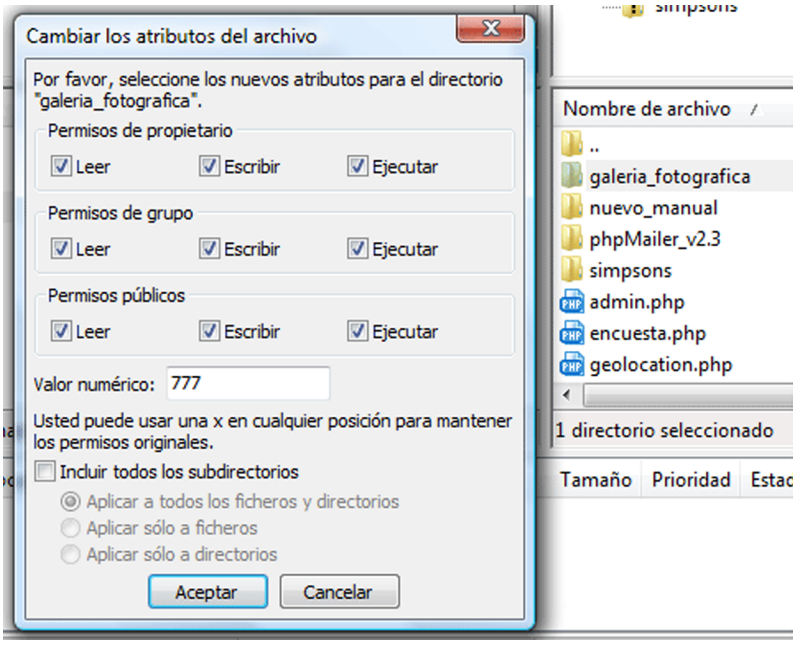
Tras darle al botón "aceptar" se establece la conexión a Comoras. A la izquierda tenemos el sistema de archivos del Sitio Local (nuestro PC) y a la derecha la ubicación de nuestro sistema de archivos en el Sitio Remoto (Comoras).

El directorio que aparece por defecto al entrar al Sitio Remoto es la carpeta raíz del sitio web en Comoras. Todos los ficheros HTML, PHP, CSS, JS, GIF, JPEG, etc. que se suban al servidor serán accesibles desde Internet.

Es importante tener en cuenta que los sistemas de ficheros basados en Linux hacen distinción entre mayúsculas y minúsculas, con lo cual form.php es distinto a Form.php. Para evitar conflictos con los nombres de los ficheros, es recomendable nombrarlos siempre en minúscula y evitar los acentos o las letras especiales como la *ñ*. Igualmente, es conveniente no nombrar los ficheros con espacios en blanco (por ejemplo: "validador usuario"); una buena solución es sustituir el espacio en blanco por un guión (por ejemplo, "validador_usuario").

Algunas aplicaciones web requieren que los archivos tengan determinados permisos. Por ejemplo, podría darse el caso de querer añadir una imagen a una galería fotográfica. Para ello, será necesario dar a la carpeta que contendrá las fotografías los permisos de lectura y de escritura.

Si queremos dar todos los permisos a todos los archivos de un directorio con Filezilla, creamos una nueva carpeta y la llamamos, por ejemplo, "galería_fotografica". Después, seleccionamos la carpeta con el puntero del ratón y abrimos su menú contextual, luego elegimos la opción "Permisos de Achivo". Finalmente, marcamos las casillas "Escribir" en las secciones de Permisos de Grupo y Permisos públicos.



Normalmente, el valor numérico de los permisos recomendados es 644 para que el "grupo" y el "resto" de usuarios tengan sólo acceso de lectura y así se puedan visualizar las páginas web.

1.5. Probando PHP en el servidor remoto

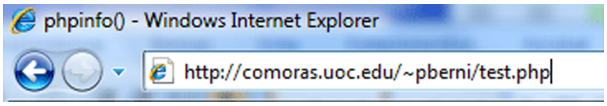
Lo primero que hay que hacer para la puesta en marcha del entorno de producción es comprobar si tenemos habilitado PHP en nuestro espacio web. El proceso es muy sencillo y rápido de llevar a cabo. Creamos con un editor de texto simple un fichero test.php con la siguiente porción de código:

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

Tras salvar el fichero lo subimos a la carpeta raíz de Comoras con Filezilla. Luego abrimos el navegador y escribimos la siguiente ruta (recordad poner en "usuario" el *login* que os ha sido asignado):

```
http://comoras.uoc.edu/~usuario/test.php
```

Observad cómo la ruta http://comoras.uoc.edu/~usuario/ apunta a la carpeta raíz en Comoras donde se almacenan los archivos del sitio. El carácter '~' sirve para indicar a Apache que debe servir la página desde el *home* del usuario.



La función phpinfo() del *script* test.php imprime en pantalla una gran cantidad de información sobre los parámetros de configuración y el estado actual de PHP. También podemos comprobar las versiones de Apache, PHP y MySQL instaladas en Comoras, así como las extensiones de PHP habilitadas. Por ejemplo, para MySQL se muestra la siguiente información:

mysql

MySQL Support	enabled
Active Persistent Links	0
Active Links	0
Client API version	5.0.38
MYSQL_MODULE_TYPE	external
MYSQL_SOCKET	/var/run/mysqld/mysqld.sock
MYSQL_INCLUDE	-I/usr/include/mysql
MYSQL_LIBS	-L/usr/lib -lmysqlclient

1.6. Probando MySQL en Comoras

Cada alumno tiene en Comoras su cuenta personal para poder utilizar la base de datos MySQL a través de páginas web. Las mismas claves de usuario y contraseña se utilizan para administrar la cuenta remota de MySQL y, también, para conectarse a la base de datos desde un *script* de PHP.

MySQL utiliza el puerto 3306 por defecto para que el servidor sea accesible desde Internet con una herramienta cliente de administración. Este puerto está abierto en el cortafuegos de Comoras, por lo que es posible conectarse remotamente con el servidor para acceder a la base de datos o realizar tareas administrativas. Existen muchos programas clientes con interfaz gráfica de usuario para utilizar el servidor de bases de datos MySQL. Los desarrolladores de MySQL también suministran tres de estos programas:

- MySQL Administrator: esta herramienta se emplea para la administración de servidores, bases de datos, tablas y usuarios de MySQL.

- MySQL Query Browser: esta herramienta gráfica distribuida por MySQL AB permite crear, ejecutar y optimizar consultas dirigidas a bases de datos MySQL.

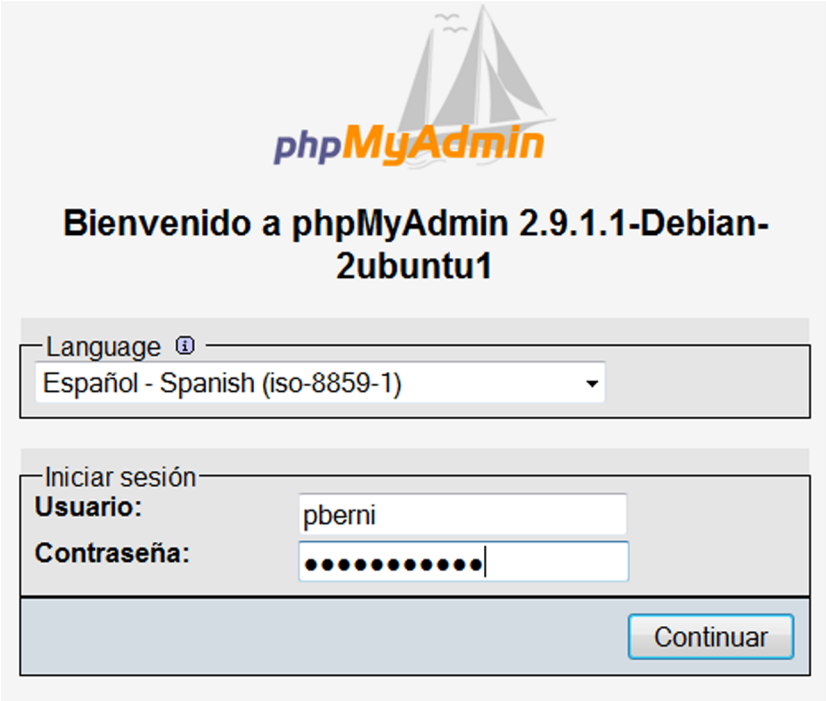
- MySQL Migration Toolkit: esta herramienta está orientada a brindar asistencia en el proceso de migración de esquemas y datos desde otros sistemas gestores de bases de datos relacionales hacia MySQL.

Los usuarios de Comoras pueden gestionar su cuenta de MySQL desde el mismo servidor de la UOC, usando la herramienta phpMyAdmin. Se trata de un proyecto de código abierto, escrito con PHP, para administrar la base de datos a través de una interfaz web intuitiva que permite acceder a todas las funciones típicas de MySQL.

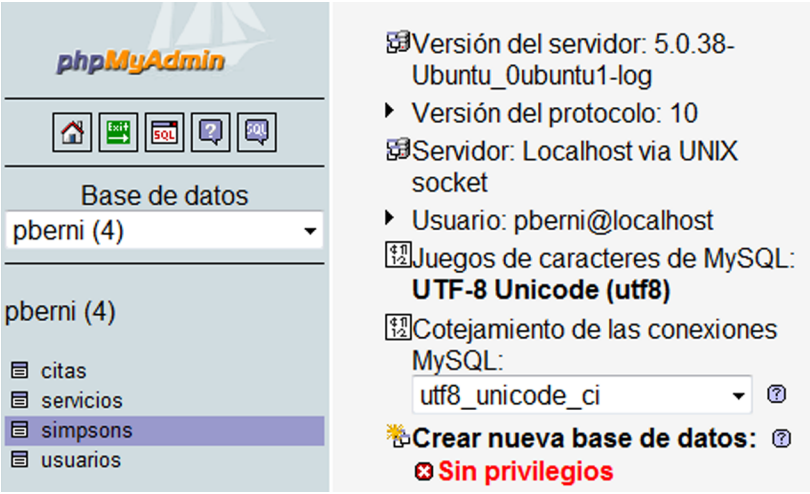
Para acceder a phpMyAdmin, basta con poner en el navegador la dirección de phpMyAdmin:

`http://comoras.uoc.edu/phpmyadmin/`

La página de inicio del programa nos pedirá las dos claves de acceso para entrar en la aplicación (las mismas del FTP). Las *cookies* deben estar habilitadas en el navegador o, de lo contrario, dará un aviso y no podremos entrar.



Al entrar en la aplicación, vemos a la izquierda la base de datos que tenemos creada por defecto en Comoras y que se llama igual a nuestro nombre de usuario. Junto al nombre de la base de datos aparece un valor entre paréntesis que se corresponde con el número de tablas contenidas en esa base de datos (el valor es 0 si está vacía).



En el ejemplo de la ilustración anterior vemos la base de datos pberni de un usuario que tiene ese mismo nombre por *login*. Debajo, los nombres de las cuatro tablas creadas para esa base de datos. A la derecha aparece el mensaje en rojo "Sin privilegios" bajo la opción de crear nuevas bases de datos. Debido a ello, no es posible crear otras bases de datos en nuestra cuenta de MySQL de Comoras, lo que nos obligará a colocar todas las tablas que hagamos en un único contendedor.

Con phpmyadmin se puede crear, copiar, borrar y modificar tablas; borrar, editar y añadir campos a cada tabla; ejecutar sentencias SQL, así como importar y exportar bases de datos salvadas en archivos de texto con extensión .sql. La documentación oficial del programa se encuentra en uno de los enlaces de la misma página principal.

1.7. Importar una base de datos MySQL a la cuenta en Comoras

Vamos a ver a continuación cómo importar a Comoras una base de datos de nuestro sistema local, para seguidamente hacer un test de conexión a los datos desde un *script* de PHP. Tomemos como ejemplo práctico la base de datos de los Simpsons que habéis trabajado durante la puesta en marcha del SGBD local.

Para migrar la base de datos de los Simpsons de local a Comoras, el primer paso es hacer un *backup* de ésta. Existen varias alternativas para hacer copias de seguridad, como, por ejemplo, la herramienta gráfica MySQL Migration Toolkit. No obstante, vamos a ver el proceso de una manera más sencilla y rápida. Para ello, usaremos la utilidad mysqldump de MySQL.

Reflexión

Podéis ver más información sobre cómo usar mysqldump en el módulo de MySQL.

La utilidad mysqldump

mysqldump permite hacer la copia de seguridad de una o múltiples bases de datos. Además, permite que estas copias de seguridad se puedan restaurar en distintos tipos de gestores de bases de datos, sin la necesidad de que sean MySQL. Esto se consigue creando unos ficheros que contienen todas las sentencias SQL necesarias para poder restaurar la tabla y su contenido. Este fichero incluye, para cada tabla, la sentencia de creación de la tabla y una sentencia *insert* por cada uno de los registros que forman parte de ésta. Las

opciones que se pueden especificar a la hora de realizar la copia de seguridad las encontraréis detalladas en el manual de referencia de MySQL 5.0.

Antes de proceder al *backup*, conviene tener presente que la base de datos de los Simpsons se encuentran en una tabla llamada mitabla y dentro de una base de datos llamada ejemplo.

El comando para ejecutar el *backup* desde Linux es (tras darle a "Enter", nos pedirá introducir la contraseña de root):

```
mysqldump --opt -u root -p ejemplo > simpsonsbackup.sql
```

Desde la línea de comando de Windows, el comando es muy parecido a Linux. Por ejemplo, si nuestro SGBD se ha instalado con WAMPSERVER, bastará con escribir:

```
C:\wamp\bin\mysql\mysql5.0.51b\bin\mysqldump.exe --opt -u
root –p ejemplo > simpsonsbackup.sql
```

La copia de seguridad se salva en un archivo de texto (simpsonsbackup.sql). Si editamos este fichero, veremos cómo el volcado contiene comandos SQL para crear la tabla y/o rellenarla.

Al comienzo del fichero hay dos sentencias SQL con el nombre de la base de datos local ("ejemplo") que deberemos modificar, o de lo contrario no se podrán importar las tablas ni sus datos a nuestra base de datos remota:

```
CREATE DATABASE IF NOT EXISTS ejemplo;
USE ejemplo;
```

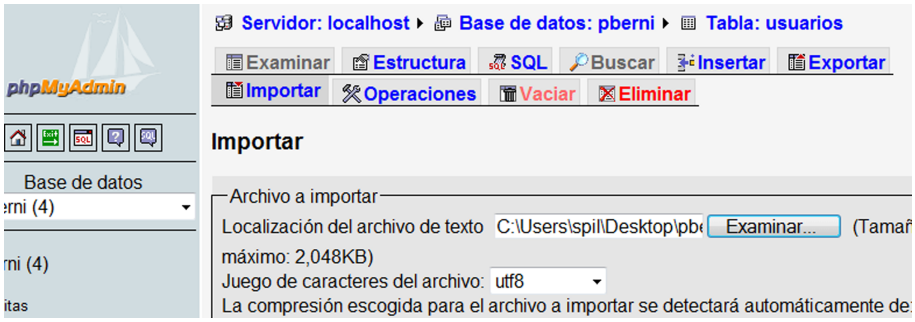
Cambiamos el nombre de la base de datos por el que ya tenemos asignado en Comoras.

```
CREATE DATABASE IF NOT EXISTS pberni;
USE pberni;
```

Guardamos los cambios y abrimos phpmyadmin en Comoras con el navegador.

Seleccionamos la base de datos con nuestro nombre de usuario en la columna izquierda de la página y, seguidamente, hacemos clic en el enlace "Importar". A continuación, la ventana de importar nos permite subir a Comoras el fichero del *backup* (simpsonsbackup.sql). En la selección de "Juego de caracteres" debemos seleccionar la codificación de los caracteres de la base de datos original (habitualmente se utiliza Latin1 o UTF-8); esta opción se puede especificar en el momento de exportar la base de datos con mysqldump.

Ya sólo nos queda hacer clic en "Continuar" y esperar a que se cargue el archivo.



1.8. Conectando a MySQL desde un *script* de PHP en Comoras

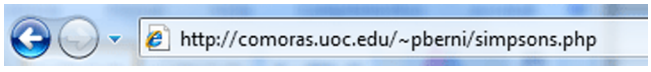
Vamos a probar la base de datos de los Simpson en Comoras como test de conexión a nuestra cuenta de MySQL en Comoras mediante un *script* de PHP. Abrimos el editor de texto y creamos un nuevo *script* "simpsons.php".

Siempre que especifiquemos una conexión al servidor MySQL desde Comoras usaremos en *hostname* el valor de "localhost" como servidor (*localhost* es un nombre reservado que tienen todas las computadoras para referirse a sí mismas. El nombre *localhost* es traducido como la dirección IP de *loopback* 127.0.0.1 en IPv4).

Las claves de usuario que recibisteis por correo se colocan en los *argumentos usuario* y *password* de la función mysql_connect().

Una vez establecida la conexión con éxito, seleccionamos la base de datos en Comoras con la función mysql_select_db(). Recordad que nuestra base de datos en Comoras se llama igual a nuestro nombre de usuario. Este detalle es importante, sobre todo, si se trata de ejecutar un *script* que hemos subido a Comoras desde nuestro sistema local, lo que nos obliga a renombrar la base de datos, o de lo contrario fallará la selección.

Finalmente, enviamos una sencilla consulta MySQL:



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
       <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
       <title>List&iacute;n telef&oacute;nico familia Simpsons</title>
</head>
<body>
       <h4>List&iacute;n telef&oacute;nico familia Simpsons</h4>
       <?php
       # Abro una conexión al servidor MySQL de Comoras
       $link = mysql_connect('localhost', 'pberni', '******');
       if (!$link) {
              die('No puedo concetar: ' . mysql_error());
       }

       # Selecciono la base de datos de los Simpson
       $db_selected = mysql_select_db('pberni', $link);
       if (!$db_selected) {
              die ('No puedo seleccionar pberni : ' . mysql_error());
       }

       # Envío una consultaMySQL
       $result = mysql_query('SELECT * from mitabla');
       if (!$result) {
              die('Consulta inválida: ' . mysql_error());
        }

       # Extraigo las filas del resultado como un objeto
       while ($row = mysql_fetch_object($result)) {
              $id = $row->id;
              $nombre = $row->nombre;
              $telefono = $row->telefono;
       ?>
       <p>
              ID: <?=$id;?><br />
              Nombre: <?=$nombre;?><br />
              Tel&eacute;fono: <?=$telefono;?><br />
       </p>

       <?php
          }
       # Cierro el enlace abierto con Comoras
       mysql_close($link);

       ?>
</body>
</html>
```

y mostramos el resultado de la consulta en una página del navegador:

Listín telefónico familia Simpsons

ID: 1
Nombre: Homer Simpson
Teléfono: 555-1234

ID: 2
Nombre: Bart Simpson
Teléfono: 555-4321

ID: 3
Nombre: Lisa Simpson
Teléfono: 555-3214

ID: 4
Nombre: Marge Simpson
Teléfono: 555-2314

ID: 5
Nombre: Maggie Simpson
Teléfono: 555-3142

2. Anexo 2: Máquinas virtuales

Los requisitos necesarios para el funcionamiento de un servidor web dinámico, usando Apache como servidor web, PHP como módulo dentro de Apache para generar código HTML dinámicamente y con MySQL como gestor de bases de datos, son relativamente bajos si los comparamos con las prestaciones que los ordenadores de hoy en día ofrecen.

Los requisitos son aún menores si el servidor web no va a necesitar una complejidad excesiva en la creación de páginas web dinámicas ni un *throughput*[1] de conexiones muy elevado.

(1) *Throughput* es la cantidad conexiones simultáneas para ofrecer los servicios web.

En Internet, uno podrá encontrar gran cantidad de artículos que animan a reciclar ordenadores "presuntamente" obsoletos para alojar páginas web, ya que con 300 MHz de CPU, 256 MB de RAM, el espacio en disco suficiente para alojar los contenidos y conexión a la Red, se puede crear un servidor de páginas web totalmente funcional.

Un ordenador actual podría perfectamente equipararse a diez ordenadores como el descrito para alojar un portal. El uso de máquinas virtuales permite esto exactamente. Con una aplicación de virtualización de máquinas podemos hacer que un ordenador contenga varias máquinas virtualizadas, que compartirán el procesador, la RAM, el disco duro y las conexiones de red.

VMWare es una herramienta de virtualización de máquinas y tiene una versión gratuita para Microsoft Windows en el sitio web de VMWare.

VirtualBox

Otra de las herramientas de virtualización gratuitas más extendidas es el VirtualBox.

Esta versión gratuita no permite la creación de nuevas máquinas virtuales, pero permite cargar imágenes de máquinas virtuales creadas.

Existen comunidades y organizaciones en Internet, como VMware, donde se ofrece la descarga de máquinas virtuales para diferentes propósitos, tales como el testeo de diferentes sistemas operativos (máquinas limpias, simplemente con el sistema operativo, para uso genérico a las que se les deberá añadir el software deseado) y servidores (máquinas con software específico instalado, por ejemplo, para creación de servidores web, de servicios de monitorización, herramientas de seguridad, etc.).

Para nuestro caso, podremos descargar una versión de Ubuntu, distribución Linux que ha sido elegida dentro de la UOC debido a su equilibrio entre facilidad de uso y gran difusión. Ubuntu 8.10 Server Virtual Appliance (LAMP).

Podemos disponer de una imagen de máquina virtual LAMP con unos requisitos de hardware incluso más reducidos que para la máquina Ubuntu, pero penalizando la facilidad de uso del sistema operativo Ubuntu.

Las máquinas virtuales en VMWare están compuestas como mínimo de un fichero de configuración y un fichero de imagen con el contenido del disco duro de la máquina que vamos a emular.

El fichero de configuración es un fichero con extensión .vmx. Este fichero contiene los parámetros necesarios para crear la máquina virtual en lo que a hardware se refiere. Es decir, define la cantidad de RAM de la máquina virtual, la cantidad de CPU, la existencia de USB, CD/DVD, tarjetas de red y su configuración, el tamaño del disco duro principal (y de los secundarios si existen), etc.

El fichero de disco duro tiene una extensión .vmdk. Simula el contenido del disco duro de la máquina virtual. Por lo tanto, contendrá el sistema operativo, las aplicaciones y los datos que se hayan almacenado en él. Como características avanzadas, se pueden hacer congelaciones del disco duro. La congelación del disco consiste en el almacenamiento del estado actual del disco para que, toda modificación que se realice sobre éste, quede anulada después de un reinicio. Esta característica puede ser interesante en procesos que puedan hacer peligrar la información existente dentro de la máquina virtual. Por ejemplo, un programador web, después de haber creado un portal web con contenidos dinámicos alojados dentro de una base de datos, validaciones de parámetros y establecimiento de sesiones, podrá realizar una congelación del disco y posteriormente lanzar aplicaciones de testeo de seguridad sobre su portal web que podrían destruir el trabajo realizado.

Como mencionamos anteriormente, en webs de comunidades de desarrollo, tales como VMware, podemos obtener máquinas virtuales ya preparadas para ser usadas en un cliente VMWare de máquinas virtuales.

Para poder trabajar con estas imágenes, será necesario que copiéis los ficheros de la máquina virtual descargada en una carpeta local en el ordenador, es decir, deberán copiarse los ficheros en el disco duro del ordenador que alojará la máquina virtual. Recomendamos hacer una carpeta en "C:\GMMD\Maquinas Virtuales" donde se hará la copia en local de los ficheros de la máquina virtual.

A continuación mostramos un ejemplo de una máquina virtual Ubuntu 8.10 Desktop en ejecución a través de la aplicación VMWare-Player.

Abriendo la máquina virtual con VMWare-Player, puede observarse que se trata de una máquina Ubuntu en su totalidad. Es posible cambiar ciertas preferencias de la máquina virtual, tal como su cantidad de memoria RAM, agregarle una unidad lectora de DVD/CD, una unidad lectora de disquetes, etc. Para activar o desactivar los periféricos, simplemente se debe presionar sobre el icono correspondiente (CD, disquete, tarjeta de red o audio).

Para apagar la máquina virtual, existe la opción correspondiente en:

Player → Troubleshood → Reset o **Power Off and Exit**

Cabe mencionar que, para que la máquina virtual pueda acceder a Internet, es recomendable que la tarjeta de red esté en modo NAT.