

*Guía Práctica  
para usuarios*

# Desarrollo Web con PHP y MySQL

José Antonio Gallego Vázquez



**ANAYA**  
MULTIMEDIA

+ . 5 - 6 - 2003  
No. Reg. 02138  
GUÍAS PRÁCTICAS

681.3  
Ball  
D

INFORMÁTICA  
DISEÑO WEB  
PHP  
SQL

Responsable editorial:  
Eugenio Tuya Feijoo

Diseño de cubierta:  
Narcís Fernandez

Realización de cubierta:  
Gracia Fernandez-Pacheco

Reservados todos los derechos. El contenido de esta obra está protegido por la Ley, que establece penas de prisión y/o multas, además de las correspondientes indemnizaciones por daños y perjuicios, para quienes reprodujeren, plagiaren, distribuyeren o comunicaren públicamente, en todo o en parte, una obra literaria, artística o científica, o su transformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier medio, sin la preceptiva autorización.

© EDICIONES ANAYA MULTIMEDIA (GRUPO ANAYA) S. A., 2003  
Juan Ignacio Luca de Tena, 15. 28027 Madrid  
Depósito legal: M. 14.294-2003  
ISBN: 84-415-1525-5  
Printed in Spain  
Impreso en Anzós, S. L. - Fuenlabrada (Madrid)

*Esta guía, como toda **mi** vida,  
esta dedicada a Idoia.*

# Índice

Introduccion .....	15
Como usar este libro .....	19
1. Introduccion .....	23
1.1. Páginas Web dinamicas vs. estaticas .....	24
1.2. Sobre el PHP .....	26
1.2.1. Historia del PHP .....	28
1.3. MySQL .....	28
1.4. Servidor Apache .....	29
1.5. Un enfoque práctico .....	30
2. Instalacion y configuración .....	31
2.1. Introduccion .....	31
2.2. Licencia GPL .....	32
2.3. Instalacion .....	32
2.4. PHP Home Edition 2 .....	33
2.4.1. Instalacion PHP Home Edition 2 .....	33
2.4.2. Configuración y utilización .....	35
2.5. Nuestro primer script en PHP .....	39

3. Introduccion a PHP .....	41
3.1. Introduccion .....	41
3.2. Nuestro primer paso en PHP .....	41
3.2.1. Apertura y cierre en PHP .....	42
Otros metodos de indicarlo .....	43
3.2.2. Como se comenta el codigo .....	43
3.2.3. Como indicar la fecha en PHP .....	44
3.3. Variables .....	46
3.3.1. Primer contacto con variables .....	46
3.3.2. Mayusculas y minusculas .....	46
3.3.3. Adicion de variables .....	47
3.4. Operadores .....	47
3.4.1. Operadores aritméticos .....	47
3.4.2. Operadores de texto .....	47
3.5. HTML y PHP. Formularios .....	48
3.5.1. bienvenida.html .....	48
3.5.2. saludo.php .....	49
3.6. Estructuras de control .....	51
3.6.1. If-else .....	51
3.6.2. clase.php .....	51
3.6.3. While .....	53
3.6.4. bucle.php .....	53
3.6.5. Elseif .....	54
3.7. Ejercicio practico .....	56
4. Introduccion a MySQL .....	57
4.1. Introduccion a las bases de datos .....	57
4.2. ¿Qué es y para que sirve una base de datos? .....	57
4.2.1. Ordenemos la biblioteca .....	58
4.3. Primeros pasos con MySQL .....	59
4.3.1. Crear una base de datos .....	60
4.3.2. Crear tablas en nuestra base de datos .....	61
Hagamos la prueba con un ejemplo práctico .....	62
4.3.3. Sentencias SHOW y DESCRIBE .....	63
4.3.4. Sentencia DROP .....	64
4.3.5. Insertar datos en las tablas .....	65
4.3.6. Examinar los datos de nuestra base: .....	65
Comando SELECT .....	66

4.3.7. Modificar elementos en nuestra base de datos .....	68
4.4. Repaso a las sentencias básicas .....	69
5. Acceder a los datos en MySQL a traves de la Web gracias a PHP .....	71
5.1. Como funciona una pagina Web dinamica .....	71
5.2. Como conectar PHP con MySQL .....	72
5.2.1. Conexión .....	73
5.3. Seleccionar una base de datos .....	75
5.4. Busquedas en la base de datos .....	75
5.5. Mostrar los datos en pantalla .....	76
5.5.1. Aplicacion practica .....	76
5.6. ColegioMaravillas (primera version) .....	78
5.6.1. apellido.php .....	78
5.6.2. alta.php .....	80
5.6.3. buscador.php .....	82
5.6.4. queridos.php .....	84
6. Control y proceso de la información en nuestra Web ..	89
6.1. Funciones de control de texto en PHP .....	90
6.1.1. Funcion trim(),ltrim() y chop().....	90
Soluciones alternativas son las funciones ltrim() y chop().....	90
6.1.2. Funcion nl2br() .....	90
6.1.3. Funcion htmlspecialchars() .....	91
6.1.4. Funcion strtoupper(), strtolower(), ucfirst() y ucwords() .....	92
6.1.5. Funcion AddSlashes() y StripSlashes() .....	94
6.2. Como buscar y reemplazar palabras y simbolos en las cadenas de texto .....	95
6.2.1. Como identificar cadenas de texto .....	95
6.2.2. Sustituir una cadena de texto por otra .....	96
6.2.3. Clases de caracteres .....	96
Aplicacion practica :validación del campo de correo electrónico .....	97
7. MySQL avanzado .....	99
7.1. Por que necesitamos copias de seguridad .....	99
7.1.1. Como realizar copias de seguridad de nuestras bases de datos .....	100

7.2. Privilegios para acceder a nuestra base de datos .....	103
7.2.1. Tipos de privilegios en MySQL .....	104
Privilegios para usuarios .....	104
Privilegios para administradores .....	105
Privilegios especiales .....	105
7.2.2. Conceder y quitar privilegios: funciones GRANT y REVOKE .....	105
7.2.3. Conceder y quitar privilegios. Ejemplo práctico .....	106
7.3. Trabajar con varias tablas de datos .....	107
7.3.1. Por que utilizar diferentes bases de datos .....	107
7.4. Consultar diversas bases de datos .....	109
8. PHP avanzado .....	125
8.1. Por que reutilizar el codigo .....	125
8.1.1. Sentencias require() e include() .....	125
8.1.2. Diferencias entre require() e include() .....	127
8.1.3. Utilizacion de plantillas gracias a include()....	127
8.2. Creación de una lista de correo .....	129
9. PostNuke .....	135
9.1. Que es PostNuke .....	135
9.2. Características de PostNuke .....	137
9.3. Documentación de PostNuke .....	138
9.4. Instalación de PostNuke .....	139
9.5. Configuración de PostNuke .....	141
9.5.1. Opciones del menu Administración .....	141
Submenu Settings .....	141
Submenu Polls .....	143
Submenú Admin .....	143
9.6. Utilización de PostNuke .....	145
9.6.1. Validación de mensajes .....	145
9.6.2. Control de estadísticas .....	145
9.6.3. Lista de miembros .....	146
9.7. Conclusion .....	146
10. osCommerce: solución Open Source de Comercio Electrónico .....	147
10.1. Introducción .....	147
10.2. osCommerce .....	148

10.3.Características fundamentales de osCommerce ....	149
10.4. Instalacion de osCommerce .....	152
10.4.1.Requisitos de instalacion .....	152
10.4.2. Como subir los archivos via FTP al servidor ..	152
10.4.3.Ejecutar el script de instalacion .....	153
10.5. Configuracion de osCommerce .....	154
10.5.1. Opciones Administrador .....	154
10.5.2. Opciones Cliente (estructura de la tienda).....	157
10.5.3.Compra simulada .....	158
10.5.4.Gestion de los pedidos .....	159
10.6. Conclusion .....	160
 11. phpBB: solución Open Source para la creación de foros personalizados .....	161
11.1.Introduccion .....	161
11.2.phpBB .....	161
11.3.Caracteristicas de phpBB .....	162
11.4. Instalacion de phpBB .....	162
11.5.Configuracion de phpBB .....	163
11.6.Creación de un foro .....	166
11.7. Conclusion .....	167
 12. Cookies y sesiones .....	169
12.1. <b>Que</b> es una cookie .....	169
12.2. Funcionamiento de las cookies .....	170
12.3.Cómo crear nuestras propias cookies .....	171
12.4. Conclusion .....	174
12.5. <b>Que</b> son las sesiones .....	174
12.6. Como crear las sesiones .....	175
12.6.1.session_start() .....	175
12.6.2.session_register() .....	176
12.6.3.session_is_registered(); .....	176
12.7. Cerrar las sesiones .....	177
12.8. sesion.php .....	177
12.9. duracion.php .....	178
12.10.Autentificación de usuarios mediante sesiones ....	179
 13.eMule: Solución Open Source de intercambio de archivos .....	185
13.1. Introduccion .....	185
13.2. <b>¿Qué</b> es eMule? .....	186

13.3. Caracteristicas fundamentales de eMule .....	187
13.4. Instalacion de eMule .....	188
13.5. Configuracion e instalacion de eMule .....	189
13.5.1. Conexion a un servidor .....	189
13.5.2. Configuracion de las diversas opciones .....	191
Opcion General .....	191
Opcion Conexion .....	192
Opcion Busqueda .....	193
Opcion Tráfico .....	193
13.6. Ayuda con eMule .....	193
14. Programacion orientada a objetos .....	197
14.1. Introduccion .....	197
14.2. Concepto .....	197
14.3. Clases y objetos .....	198
14.4. Caracteristicas de clases y objetos .....	199
14.5. Creacion de clases, atributos y funciones en PHP .....	200
14.5.1. Creacion de una clase .....	200
14.5.2 Constructores .....	201
14.5.3. Creacion de objetos dentro de una clase .....	201
14.6. Como utilizar los atributos de la clase .....	202
14.6.1. Funciones de acceso .....	203
14.6.2. Acceder a las operaciones dentro de las clases .....	204
14.7. Herencia en PHP .....	205
14.8. Ejemplo. Programacion orientada a objetos vs. Programación convencional .....	206
15. Trabajar con multiples bases de datos .....	213
15.1. Introduccion .....	213
15.2. Diseñar las relaciones entre nuestras bases de datos .....	213
15.2.1. Relaciones de "uno a uno" .....	213
15.2.2. Relaciones de "uno a varios" .....	214
15.2.3. Relaciones de "varios con varios" .....	216
15.2.4. Conclusion .....	218
15.3. Creación de un foro. Ejemplo práctico .....	218
15.3.1. Caracteristicas del foro .....	218
15.3.2. Bases de datos .....	219

15.3.3. Scripts .....	221
15.3.4. Conclusion .....	237
<b>16. Como hacer nuestras aplicaciones seguras .....</b>	<b>239</b>
16.1. Introducción .....	239
16.2. Autentificacion con PHP y MySQL .....	239
16.2.1. Sencillo mecanismo de control .....	240
16.2.2. Mecanismo de control que almacena las contraseñas en una base de datos .....	243
16.2.3. Mecanismo de autentificacion mediante sesiones .....	245
16.3. Encriptacion de contraseñas .....	249
16.4. Proteger multiples páginas .....	249
16.4.1. Autentificacion basica .....	250
16.4.2. Como utilizar la autentificacion basica con PHP .....	250
16.5. Como establecer una política de seguridad en nuestra pagina Web .....	252
16.5.1. Interrupción en la transmision de datos confidenciales .....	252
Establecer una conexión segura en nuestro servidor (SSL) .....	255
Confiar todo el tema de nuestros cobros a una empresa especializada .....	256
<b>A. Funciones de PHP que actuan con MySQL .....</b>	<b>259</b>
A.1. Funciones de PHP que trabajan con MySQL .....	259
mysql_affected_rows .....	259
mysql_close .....	260
mysql_connect .....	260
mysql_create_db .....	260
mysql_data_seek .....	260
mysql_dbname .....	261
mysql_drop_db .....	261
mysql_error .....	261
mysql_fetch_array .....	261
mysql_fetch_field .....	262
mysql_fetch_object .....	263
mysql_fetch_row .....	263
mysql_field_seek .....	263

mysql_fieldflags .....	264
mysql_fieldlen .....	264
mysql_fieldname .....	264
mysql_fieldtable .....	264
mysql_fieldtype .....	264
mysql_free_result .....	265
mysql_list_dbs .....	265
mysql_list_fields .....	265
mysql_list_tables .....	265
mysql_num_fields .....	266
mysql_num_rows .....	266
mysql_pconnect .....	266
mysql_query .....	266
mysql_regcase .....	267
mysql_result .....	267
mysql_select_db .....	267
mysql_tablename .....	267
<b>B. Tipos de columnas en MySQL .....</b>	<b>269</b>
Parametros .....	269
Atributos .....	270
Valores numéricos .....	270
Valores de fecha y tiempo .....	272
Valores de caracteres .....	273
<b>C. Guía de referencia rápida de HTML .....</b>	<b>275</b>
<b>C.1. Etiquetas de HTML .....</b>	<b>275</b>
<b>Etiquetas básicas .....</b>	<b>275</b>
Etiquetas de atributos .....	276
Etiquetas de texto .....	276
Etiquetas de links .....	277
Etiquetas de formato .....	278
Etiquetas de elementos gráficos .....	279
Etiquetas de tablas .....	279
Atributos de tabla .....	280
Frames .....	280
Atributos de los Frames .....	281
Formularios .....	282
<b>Índice alfabético .....</b>	<b>285</b>

# Introducción

Ya no cabe ninguna duda de que el fenómeno Internet ha revolucionado definitivamente el modo en como los seres humanos nos comunicamos y relacionamos. Cuando los historiadores del futuro analicen el final del siglo XX y el principio del XXI, seguro que valoraran el fenómeno Internet como un impacto comparable al que tuvieron el descubrimiento de la imprenta a finales del XV y la posterior alfabetización masiva que convirtió el conocimiento en algo universal, o la revolución industrial de finales del XIX, con los consiguientes cambios sociales, económicos y demográficos que transformaron la faz de la sociedad.

A falta de una perspectiva temporal que nos impide calibrar el impacto a largo plazo de la "Red de redes" en nuestra sociedad, ya podemos encontrar algunas diferencias de peso entre estos fenómenos:

Si tomamos la fecha de invención de la imprenta en 1450, a finales de ese siglo solo se habían impreso 6000 obras diferentes, y su campo de acción se reducía casi exclusivamente a lo que hoy conocemos como Italia. En España, por ejemplo, esta invención no se popularizaría hasta bien entrado el siglo XVI.

En la revolución industrial nos encontramos un fenómeno parecido. En 1769 Watt patenta una máquina de vapor cuya misión era achicar el agua de las minas, pero realmente no es hasta el siglo XX cuando la producción industrial se convierte en un fenómeno verdaderamente global.

Sin embargo, la velocidad con que Internet ha pasado de ser un simple experimento de laboratorio a ser un elemento imprescindible de nuestra vida cotidiana es sencillamente increíble.

Los primeros intentos de crear una red de ordenadores descentralizada que se comunicasen entre sí no surgió hasta los años 60, cuando en plena guerra fría J.C.R. Licklider, psicólogo e informático que trabajaba para el Pentágono, presentó sus primeras investigaciones acerca de cómo la comunicación entre ordenadores podía ayudar al desarrollo del pensamiento humano.

A finales de los 60, y tras una serie de estudios previos por parte de prestigiosos científicos del MIT (Instituto Tecnológico de Massachusetts), se retoman las propuestas de Licklider y se crea ARPANET, una red de ordenadores comunicados entre sí, con un uso fundamentalmente militar de seguridad.

El primer programa para enviar correo electrónico llegó hasta 1971, desarrollado por Ray Tomlinson, y combinaba el envío de correo electrónico con un programa de transferencia de ficheros.

Se podría decir que Internet, tal y como hoy la conocemos, es creada en 1974, cuando Vinton Cerf y Bob Kahn publican *Protocolo de Intercomunicación de Redes Por Paquetes*, donde especifican detalladamente las bases de un nuevo protocolo llamado TCP (*Transmission Control Protocol*, Protocolo de control de transmisiones), que con el tiempo se convirtió en el estándar de lo que a partir de entonces se llamaría INTERNET.

En 1983 es cuando ARPANET se desvincula del Pentágono y, poco a poco, universidades, instituciones, empresas y particulares por todo el mundo comienzan a unirse, produciendo un crecimiento sin precedentes del fenómeno INTERNET que, en este amanecer del siglo, es un elemento imprescindible en cualquier campo de la actuación humana, ya sea la economía, el arte, la filosofía, etc. Incluso el crecimiento demográfico en determinadas ciudades empieza a estar influenciado por el acceso a banda ancha de Internet.

Las estadísticas muestran una clara relación directa entre el nivel de acceso y conocimientos de Internet de una persona con su nivel de renta y calidad de vida.

Internet ha dejado de ser un hobby, un fenómeno o un elemento de diversión, para convertirse en una obligación. Nuestro nivel de integración y de prosperidad en la sociedad actual depende de un dominio adecuado de las nuevas tecnologías, y es nuestra responsabilidad estar a la altura de los tiempos y asegurarnos de que las nuevas generaciones tienen un acceso directo y preferente a este torrente de conocimiento.

Por ello, hay que tener claro que, en el caso de esta guia, iniciarnos en el diseño y creación de páginas Web dinámicas es algo más que un capricho o una afición para los ratos libres.

Dominar lenguajes como el PHP, MySQL, o aprender a configurar y manejar servidores Apache, puede convertirnos en profesionales de mayor éxito y remuneración, ciudadanos más informados y mejores padres, puesto que podremos orientar a nuestros hijos en el campo de las nuevas tecnologías y ayudarles en materias esenciales para su desarrollo personal y profesional.

Por ello, hemos querido orientar esta guia al ciudadano común y corriente, al usuario de Internet al que le gusta navegar y desea dar un paso adelante y diseñar sus propias páginas Web, al emprendedor que quiere utilizar Internet para sus negocios, al estudiante, al oficinista, etc.

Hemos huido de lenguajes dogmáticos y disertaciones excesivamente teóricas que puedan echar para atrás al lector no especializado. Por el contrario, hemos utilizado un lenguaje directo, didáctico, lleno de ejemplos prácticos fácilmente realizables y aplicables a la página Web del lector.

## **Las ventajas de las páginas Web dinámicas respecto a las estáticas**

Una de las divisiones que podemos realizar entre todos los tipos de páginas Web existentes podría ser entre estáticas y dinámicas.

Una página Web estática presenta las siguientes características:

- Ausencia de movimiento y funcionalidades.
- Absoluta opacidad a los deseos o busquedas del visitante a la página.
- Realizadas exclusivamente mediante lenguaje HTML.
- Para cambiar los contenidos de la página, es imprescindible acceder al servidor donde está alojada la página.
- El usuario no tiene ninguna posibilidad para seleccionar, ordenar o modificar los contenidos o el diseño de la página a su gusto.
- El proceso de actualización es lento, tedioso, y esencialmente manual.
- No se pueden utilizar funcionalidades tales como bases de datos, foros, etc.

Por el contrario, una pagina Web dinamica tiene las siguientes características:

- Gran numero de posibilidades en su diseño y desarrollo.
- El visitante puede alterar el diseño, contenidos o presentación de la pagina a su gusto.
- En su realización se utilizan diversos lenguajes y técnicas de programación.
- El proceso de actualización es sumamente sencillo, sin necesidad de entrar en el servidor.
- Permite un gran numero de funcionalidades tales como bases de datos, foros, etc.
- Pueden realizarse íntegramente con *software* de libre distribución.
- Existe una amplia comunidad de programadores que brinda apoyo desinteresado
- Cuenta con un gran numero de soluciones prediseñadas de libre disposición.

En definitiva, el concepto de pagina Web dinamica se ha impuesto en el mundo del diseño y de la empresa en Internet. Páginas como Yahoo, Google, Amazon, etc., son excelentes ejemplos de páginas dinámicas que permiten interactuar al visitante y le ofrecen posibilidades realmente sorprendentes: carritos de compra, posibilidad de incluir sus propias críticas en libros y discos, buscar en base a criterios determinados, participar en foros de discusion ...

Puede parecerle increíble, pero con la guía que ahora tiene en las manos, sera capaz de desarrollar herramientas Web profesionales que en nada envidiaran a las mencionadas anteriormente. Asimismo, le explicaremos en profundidad algunos de los programas realizados en PHP y MySQL, y que usted podra utilizar y adaptar libremente para crear, por ejemplo, su propia tienda *on line* con una calidad que le permitira rivalizar con El Corte Inglés, Carrefour o cualquier otra gran cadena comercial.

Es un viaje apasionante que solo requiere un poco de su esfuerzo y atención, y muchísima curiosidad. ¿Está preparado? Solo tiene que seguir leyendo.

Si desea contactar con el autor de esta guía, escriba a [jose.gallego@idoweb.net](mailto:jose.gallego@idoweb.net).

# Como usar este libro

Cada vez son mas las personas que se interesan por el diseño de paginas Web y que, sin carecer de experiencia previa en informática o lenguajes de programacion, intentan adquirir ellos mismos esos conocimientos por su cuenta.

Para el especialista informatico, es facil estar al tanto de las nuevas tendencias en diseño y programacion. Sin embargo, el neofito y autodidacta a menudo se desanima entre la gran maraña de opciones que se le presentan: academias, cursos *on line* e infinidad de libros y revistas que venden las virtudes de lenguajes de programacion "magicos" que permiten hacer practicamente todo y que son una garantia de trabajo y de exito profesional.

Es a ellos a quien va dirigida esta guia. Les presentamos un manual eminentemente práctico, dirigido a un público mayoritario (aunque el especialista en otros lenguajes lo encontrara util si desea reciclarse), que utiliza un lenguaje directo y asequible sin renunciar al rigor.

He querido mostrar las enormes ventajas que presenta el lenguaje PHP en combinación con la base de datos MySQL a la hora de diseñar nuestras paginas Web, y mostrar las enormes ventajas que suponen las paginas Web dinamicas frente a las estaticas tradicionales.

Para ello, acompañaremos a un personaje imaginario, Ricardo, que desea crear una pagina Web que sirva de punto de reunion a los antiguos alumnos de su colegio, y le ayudaremos a convertir su *site* en una verdadera pagina Web dinamica llena de funcionalidades, con ejemplos prácticos que pueden ser facilmente extrapolables a otro tipo de pagina Web.

Partiendo desde cero, terminaremos esta guia siendo capaces de crear una pagina Web que incluya base de datos, foro de discusion, envío de boletines, validación de usuarios, creación de *cookies*, manejo de sesiones, etc.

Todo ello, como hemos dicho, dirigido al lector principiante. Serán útiles, sin embargo, a la hora de asimilar mejor esta guía, conocimientos de Internet a nivel de usuario, conocimientos básicos del lenguaje HTML (en ningún caso imprescindibles), y (como en tantos otros ordenes de la vida) será muy útil cierta desenvoltura con el idioma inglés. Aunque la comunidad de programadores y usuarios de PHP y MySQL de habla hispana es muy activa, el inglés sigue siendo la lengua fundamental en el mundo de Internet.

¿Y eso es todo? ¿No necesitamos costosos programas y potentes ordenadores? En absoluto. Todos los lenguajes y programas a los que nos referimos a lo largo del libro son completamente gratuitos (no por 30 días, sino para siempre) y vienen incluidos en el CD adjunto.

### **Esta guía viene distribuida de la manera siguiente:**

- **Capítulo 1.** Presenta una introducción al concepto de página Web dinámica en contraposición al de página Web estática, delimitando claramente las diferencias entre una y otra. Asimismo, tendremos nuestro primer contacto con los protagonistas de este libro, PHP y MySQL, así como el *software* de servidor Apache. Veremos su historia, sus posibilidades, y su enorme y creciente implantación.
- **Capítulo 2.** Nos enseñará cómo instalar estos componentes en nuestro ordenador, evitándonos quebraderos de cabeza, puntos muertos, y el desánimo general que suele acompañar a este tipo de operaciones, gracias a la aplicación *PHP Home Edition 2* (incluida en el CD adjunto).
- **Capítulo 3.** Nos obligará a ponernos el traje de faena, y nos introducirá las sentencias y funciones básicas del lenguaje PHP a través de sencillos y divertidos ejemplos.
- **Capítulo 4.** Será la introducción a la potente base de datos MySQL, de la que aprenderemos sus posibilidades, sentencias básicas, crearemos nuestra primera base de datos, la organizaremos, e introduciremos nuestros primeros registros y aprenderemos a mostrarlos en pantalla según determinados criterios.
- **Capítulo 5.** Haremos un breve repaso de los dos anteriores y entraremos en aquello que verdaderamente configura una página Web dinámica: la interconexión

entre el lenguaje PHP y una base de datos MySQL, y como podemos insertar, borrar y modificar datos en la base de datos a través de la Web.

- **Capítulo 6.** Nos mostrará unas utilísimas sentencias que nos permitirán controlar y dar formato a toda aquella información que se almacene en nuestra base de datos, evitando faltas de ortografía, espacios en blanco, palabras inapropiadas, etc.
- **Capítulo 7.** Profundizaremos aún más en el conocido lenguaje de manejo de base de datos SQL, que nos servirá para controlar, entre otras, bases de datos como MySQL. Aprenderemos a realizar copias de seguridad, conceder privilegios, o trabajar con varias bases de datos a la vez.
- **Capítulo 8.** Tras haber adquirido una base de conocimientos del lenguaje PHP, seguiremos avanzando en su estudio a lo largo de este capítulo, realizando algunos utilísimos casos prácticos tales como la creación de una lista de correo con boletín.
- **Capítulo 9.** Cambiaremos un poco el tono del libro y pasaremos a estudiar en profundidad un programa de libre disposición llamado *PostNuke*, creado íntegramente en PHP y MySQL, y que podremos utilizar para realizar de manera sencilla nuestra página Web.
- **Capítulo 10.** Sigue mostrándonos otra aplicación de características verdaderamente impresionantes, *osCommerce*, programa de comercio electrónico que nos permitirá crear nuestra propia tienda vía Internet.
- **Capítulo 11.** Estudiaremos la última de las aplicaciones prediseñadas que nos ayudarán a hacer crecer nuestra página Web: *phpBB*, solución *Open Source* (es decir, escrita en código abierto) para la creación de foros personalizados, que utiliza exclusivamente el lenguaje PHP y la base de datos MySQL.
- **Capítulo 12.** Aprovechando el amplio bagaje de conocimientos adquiridos a lo largo del libro, afrontaremos una de las posibilidades más útiles y a la vez más complejas del lenguaje PHP: las *cookies* y sesiones. A través de diversos ejercicios prácticos aprenderemos a manejar estos conceptos, imprescindibles a la hora de crear, por ejemplo, el carrito de compra de una tienda *on line*, o en temas de seguridad informática, para comprobar si el usuario está correctamente registrado en nuestra base de datos.

- **Capítulo 13.** Revision de una de las mas populares herramientas para el intercambio de archivos, eMule, que ha venido a remediar el vacio dejado por Napster.
- **Capítulo 14.** Introducción a una de las técnicas mas innovadoras de programacion: la programacion orientada a objetos (POO).
- **Capítulo 15.** Como diseñar aplicaciones que utilicen diversas bases de datos. Enfoque teórico y práctico. Creación de un foro para integrar en cualquier pagina Web.
- **Capítulo 16.** Dedicado a la seguridad en Internet. Como hacer nuestras aplicaciones seguras y establecimiento de politicas de seguridad en nuestro *site*.

La parte final de esta guia esta formada por una serie de anexos destinados al lector mas especializado, o a aquel que, una vez terminada la guia, desea proseguir su aprendizaje en el mundo de las páginas Web dinamicas, o simplemente como elemento de consulta para tener siempre sobre nuestra mesa de trabajo.

- **Anexo A.** Es una exhaustiva recopilacion de todas aquellas funciones de PHP que trabajan con MySQL. Es decir, aquellas que nos sirven para interactuar en nuestra base de datos a través de la Web. Aunque a lo largo de la presente guia se han estudiado las mas importantes de manera exhaustiva y con ejemplos, es aconsejable poseer una documentación que las recopile todas.
- **Anexo B.** Estudiaremos mas en detalle los distintos tipos de columnas que admiten nuestras tablas en MySQL y los distintos formatos de datos que admiten, ya sean valores de texto, de fecha, etc.
- **Anexo C.** De suma utilidad para aquellos que no sean expertos en HTML, incluye una recopilacion de las etiquetas fundamentales de este lenguaje, su funcion, y el contexto en el que se usan.

## Introducción

Atendiendo a las ultimas estadisticas, se calcula que hay alrededor de 4000 millones de paginas Web existentes. Tantas, que casi superan el numero de habitantes en nuestro planeta. Es evidente que si tenemos planeado crear una pagina Web, ya sea simplemente informativa, o planeamos crear un *site* de comercio electronico, la competencia a la que nos enfrentamos es sencillamente incontable.

¿Cómo hacer que nuestra pagina Web sea lo bastante atractiva como para llamar la atencion de los internautas que saltan de *site* en *site* con un simple clic de ratón, como abejas en un campo de flores?

Es evidente que el diseño de la pagina influye. Un diseño claro y atractivo (que no recargado) nos ayudara a atraer visitantes a nuestro sitio Web y mantenerlos allí. Para ello, el dominio (esencial) del lenguaje HTML, conocimientos de JavaScript, y el dominio del programa *Flash* pueden bastarnos para crear una pagina Web que guste a los internautas, pero en ningun caso sera una garantia de que vuelvan una y otra vez a visitarla. El "limbo virtual" de Internet esta plagado de paginas que impresionan por su diseño, pero de las que sus habituales visitantes se cansaron, debido a la ausencia de actualizaciones.

Entonces, ¿cuáles son las claves para que una pagina Web obtenga un flujo continuo de visitantes satisfechos, ansiosos de entrar en nuestra pagina y comprar nuestro producto, visitar a nuestros anunciantes, o simplemente participar en nuestra comunidad?

Fundamentalmente, Contenidos Personalizados. Esto es lo que se conoce como Pagina Web Dinamica: una pagina que interactua con el usuario, recogiendo la informacion que realmente busca y necesita.

Esa es la clave. Unos contenidos actualizados, interesantes, y adaptados a las necesidades de cada visitante.

Pensemos en las paginas Web con mayor trafico: tiendas como Amazon, buscadores como Google, portales como AOL o Terra, periodicos como The New York Times o El País...

¿Qué tienen en comun todos estos sitios entre sí?

Uno podra pensar: la gran cantidad de medios, tanto economicos como humanos, invertidos en su realización; y seria verdad, pero solo en parte. Conozco (y de hecho, he trabajado en alguno de ellos) ciclopeos proyectos de Internet donde se malgastaron cantidades ingentes de dinero y talento que hoy son estudiados en las escuelas de negocios como ejemplos de fracaso empresarial.

Y tambien he colaborado en muchos otros donde con pocos medios, pero con mucha imaginación y talento, se han logrado exitos rotundos y negocios plenamente rentables.

Y si tuviera que encontrar un factor presente en todos los exitos (y ausente en todos los fracasos) es el de ofrecer a los visitantes contenidos (comerciales, educativos, economicos o de servicios) realmente utiles, con actualizaciones continuas y personalizadas para cada cliente. Si en la naturaleza solo sobreviven los mas fuertes, en Internet solo salen adelante aquellas paginas Web dinamicas capaces de satisfacer las necesidades del usuario, por sencillas o peculiares que parezcan.

Y este mismo factor se puede aplicar tanto a proyectos de decenas de personas, como al sencillo proyecto que pretendamos llevar adelante nosotros mismos.

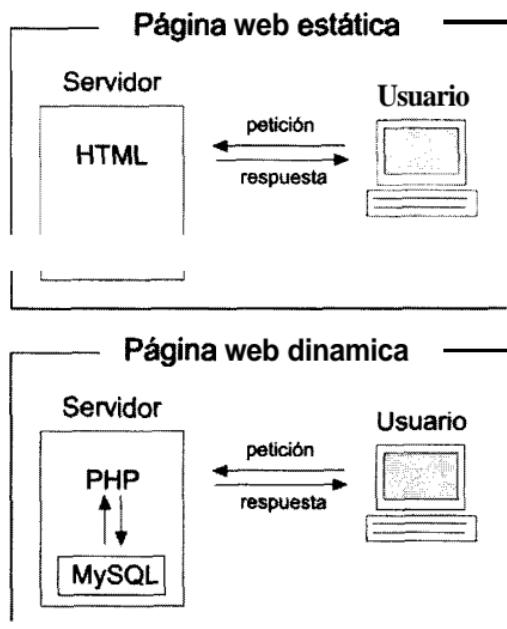
La clave es conocer los mecanismos que nos permitiran crear nuestra propia Pagina Web Dinamica, la unica pequeña garantia de tener exito en este mundo de Internet. Para ello existe esta *Guía práctica: Desarrollo Web con PHP y MySQL*.

## 1.1. Páginas Web dinamicas vs. estáticas

A la vista del grafico anterior, podemos ver la diferencia básica entre una pagina Web dinamica y otra estatica.

Cuando un usuario entra en una pagina estatica, su navegador (*Microsoft Explorer* o *Netscape* en un 99 por 100 de los casos) solicita al servidor donde esta se aloja autorizacion para ver dicha pagina, escrita en lenguaje HTML. La

pagina sera exactamente igual para todos los usuarios que entren en ella, y su contenido no variara hasta que su *webmaster* realice algún cambio. Los cambios en las paginas Web son bastante engorrosos, pues requieren conocer el lenguaje HTML e introducir dichos cambios en el servidor via FTP (*File Transfer Protocol*, Protocolo de transferencia de archives), lo cual requiere tiempo y puede dar lugar a errores, por lo que cualquier actualización en una pagina Web estática debe realizarse solo por personal cualificado.



**Figura 1.- ■Diferencias entre paginas Web estaticas y dinamicas.**

Si, por el contrario, nuestra pagina Web es dinamica, la petición de pagina por parte del navegador del visitante sera procesada por el lenguaje PHP instalado en el servidor que, dependiendo de factores como las preferencias del usuario, su país de procedencia o su idioma natal, solicitara unos contenidos específicos a la base de datos MySQL, con lo que la pagina que le sera mostrada sera absolutamente personalizada y adaptada a las características del visitante. Asimismo, las modificaciones y actualizaciones en la página Web dinámica pueden ser realizadas de manera sumamente sencilla, casi como escribiren un procesador de textos, sin necesidad de saber HTML ni utilizar ningun gestor de

FTP, por lo que cualquier persona (a la que nosotros autorizemos) podra realizar los cambios.

¿Convencido de las enormes ventajas que suponen las paginas Web dinamicas? En esta guía aprendera cómo crearlas, modificarlas e, incluso, a utilizar otras aplicaciones creadas por la comunidad de programadores de *software* libre, que facilitaran enormemente su trabajo.

Ahora, permitame presentarles a los dos actores protagonistas de nuestro libro: el lenguaje de *scripts* “del lado del servidor” PHP, la base de datos MySQL; y a un secundario de lujo, el *software* de servidor mas popular del mundo, Apache.

## 1.2. Sobre el PHP

¿Qué es PHP?

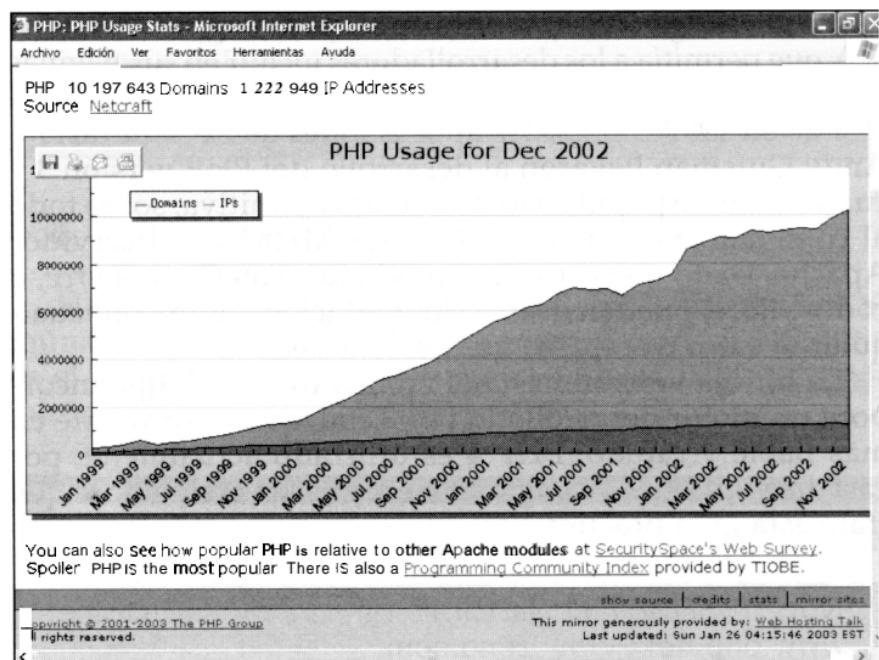
PHP (acronimo de *Hypertext Preprocessor*) es un lenguaje ”del lado del servidor” (esto significa que PHP funciona en un servidor remoto que procesa la pagina Web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de paginas Web dinamicas. Puede ser incluido con facilidad dentro del codigo HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo.

Combinado con la base de datos MySQL, es el lenguaje estandar a la hora de crear sitios de comercio electrónico o paginas Web dinamicas.

Entre sus caracteristicas fundamentales estan:

- **Gratis.** Al tratarse de *software* libre, puede descargarse y utilizarse en cualquier aplicacion, personal o profesional, de manera completamente libre.
- **Gran popularidad.** Existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su codigo, y que en muchos casos estaran encantados de echarnos una mano cuando nos enfrentemos a algún problema. Baste decir que en el momento de escribir este libro son casi diez los millones de paginas Web desarrolladas con PHP.
- **Enorme eficiencia.** Con escaso mantenimiento y un servidor gratuito (en nuestro caso, Apache), puede soportar sin problema millones de visitas diarias.

- **Sencilla integracion con multiples bases de datos.** Esencial para una pagina Web verdaderamente dinamica, es una correcta integracion con base de datos. Aunque MySQL es la base de datos que mejor trabaja con PHP (y la que, por tanto, estudiaremos en nuestra guia), puede conectarse tambien a PostgreSQL, Oracle, dbm, filePro, interbase o cualquier otra base de datos compatible con ODBC (*Open Database Connectivity Standard*).



**Figura 1.2.** Popularidad del PHP. Fuente: Netcraft ([www.netcraft.com](http://www.netcraft.com))

- **Versatilidad.** PHP puede usarse con la mayoria de sistemas operativos, ya sea basados en UNIX (*Linux, Solares, FreeBSD...*), como con *Windows*, el sistema operativo de Microsoft.
- **Gran numero de funciones predefinidas.** A diferencia de otros lenguajes de programacion, PHP fue disenado especialmente para el desarrollo de paginas Web dinamicas. Por ello, esta dotado de un gran numero de funciones que nos simplificaran enormemente tareas habituales como descargar documentos, enviar correos, trabajar con *cookies* y sesiones, etc.

## 1.2.1. Historia del PHP

Este lenguaje fue creado en 1994 por Rasmus Lerdorf como un complemento para el lenguaje PERL. Lo incorpo-ro por primera vez en su propia página Web para monitorizar las visitas que recibia.

Fue tanta su popularidad que infinidad de usuarios le pidieron a Rasmus poder utilizar estos comandos en sus paginas, por lo que Rasmus se decidió a lanzar la primera versión completa de PHP, conocida por entonces como *Personal Home Page Tools* (Herramientas para Páginas Personales), que permitia a los desarrolladores incluir en sus páginas funcionalidades como libros de visitas, foros o contadores.

Fue en 1997 cuando las aportaciones de Zeev Suraski y Andy Gutsman llevaron al desarrollo del PHP versión 3 y cuando este se popularizo de manera definitiva, sobre todo al combinarse con la base de datos MySQL y el servidor Apache. Dado que estos tres productos son *Open Source*, o código libre, pueden usarse sin limitación alguna por cualquier usuario o empresa que así lo desee.

La ultima versión hasta el momento es la 4, que incorpora un motor desarrollado por Zend, que lo convierte en más fiable y rapido. PHP 4 es utilizado actualmente por casi diez millones de páginas Web, y es la versión de la que trata esta guia práctica.

**Nota:** Para más información, descargas y documentación:  
[www.php.net](http://www.php.net) y [www.zend.com](http://www.zend.com).

## 1.3. MySQL

MySQL es, sin duda, la base de datos más popular y utilizada a la hora de desarrollar páginas Web dinámicas y sitios de comercio electrónico. Se suele trabajar en combinación con PHP, y comparte con este algunas de las características que lo convierten en una elección segura. Entre ellas:

- **Gratis.** También se trata de *software libre* que puede ser utilizado sin limitación alguna.
- **Popularidad.** Son innumerables las páginas donde encontrar información, y las listas de correo donde podrán ayudarnos desinteresadamente con nuestros proyectos.

- Rapidez. La velocidad de proceso de MySQL es legendaria. En esta página se pueden encontrar comparativas con otras bases de datos: [www.mysql.com/information/benchmarks.html](http://www.mysql.com/information/benchmarks.html).
- Versatilidad. Trabaja tanto con sistemas operativos basados en *Unix* como con el sistema operativo *Windows*, de Microsoft.
- Sencillez de manejo. Al utilizar el lenguaje estandar SQL, el tener conocimientos de otras bases de datos nos ayudara enormemente. Y aunque no sea así, con un poco de esfuerzo puede llegar a dominarse en poco tiempo.

*Nota:* Para más informacion: [www.mysql.com](http://www.mysql.com).

## 1.4. Servidor Apache

He aquí nuestro tercer guía en el apasionante viaje al mundo de las páginas Web dinámicas. El servidor Apache es el complemento perfecto para nuestras páginas dinámicas desarrolladas con PHP y MySQL.



**Figura 1.3. Popularidad del Servidor Apache. Fuente: Netcraft ([www.netcraft.com](http://www.netcraft.com))**

Comparte con estos muchas de sus características, como son la gratuidad (también se trata de *software libre*), su popularidad, su sencillez de manejo y su versatilidad, ya que podemos instalarlo sobre *Unix* o sobre Windows.

Es por ello que lo recomendamos y lo utilizaremos en este libro. De hecho, es utilizado por el 60 por 100 de los sitios activos actualmente en el mundo.

Su sencillez de manejo lo hace ideal para instalarlo en nuestro ordenador para hacer todo tipo de pruebas y ejercicios. Y cuando necesitemos contratar espacio en servidor en alguna empresa de alojamiento Web, veremos que son mayoritarias (y más económicas) las empresas que utilizan Apache en sus servidores que, por ejemplo, las que usan aplicaciones Microsoft.

*Nota: Para más información: <http://www.apache.org/>*

## 1.5. Un enfoque práctico

En esta guía aprenderemos los fundamentos de las páginas Web dinámicas con un enfoque eminentemente práctico, lleno de ejemplos claramente explicados que usted podrá realizar y modificar de acuerdo a sus propias necesidades.

Para facilitar esta tarea, nos ayudaremos de Ricardo. Ricardo es un personaje de ficción creado a la medida de esta guía. Desde hace tiempo acaricia la idea de crear una página Web para los antiguos alumnos del ya desaparecido Colegio Maravillas (también de ficción).

Ricardo quiere crear una página que sirva de contacto para todos aquellos que alguna vez pasaron por sus aulas, un punto de encuentro para charlar y recuperar viejas amistades, donde todos puedan participar e intercambiar recuerdos.

Ricardo ha estudiado el lenguaje HTML, y se ha dado cuenta que es claramente insuficiente para la idea que tiene en mente. Pero un día alguien le habló de las páginas Web dinámicas y de una guía práctica llamada *Guía práctica: Desarrollo Web con PHP y MySQL*, y no dudó un minuto en comprarla y ponerse a trabajar con ella.

A lo largo de los siguientes capítulos acompañaremos a Ricardo en su proceso, y aprenderemos cómo convertir la idea que tenemos en mente en una página Web dinámica verdaderamente profesional.

¿Nos acompañas?

# Instalación y configuración

## 2.1. Introducción

En este capítulo aprenderemos como configurar e instalar, y dar nuestros primeros pasos con los protagonistas estelares de nuestro libro: PHP y MySQL. Tambien aprenderemos a instalar y manejar a nuestro secundario de lujo: el *software* del servidor Apache.

No debemos olvidar, sin embargo, que la instalacion que realicemos en nuestro ordenador personal servira unicamente como banco de pruebas en el que realizar los ejercicios propuestos en el libro. Una vez que tengamos terminada la pagina Web y pretendamos "presentarla en sociedad", es decir, publicarla en Internet, tendremos que optar por una de las siguientes opciones:

- 1. Adquirir nuestro propio servidor.** Opción nada aconsejable a menos que seamos una gran empresa, debido al elevado precio de estos aparatos y los enormes costes que conlleva su mantenimiento (instalaciones especiales a determinada temperatura, corriente eléctrica 24 horas, etc.).
- 2. Contratar espacio en servidor en alguna empresa de alojamiento de páginas Web.** Sin duda, la opcion mas aconsejable. Dependiendo de nuestras necesidades, alquilamos la cantidad necesaria de espacio (generalmente medido en Mb) en los servidores de una empresa que se encarga de su alojamiento y mantenimiento. El coste mensual no suele ser muy elevado e, incluso, algunas empresas ofrecen alojamiento gratuito a cambio de incluir su pu-

blicidad en nuestras páginas. En buscadores como Google ([www.google.com](http://www.google.com)) u ODP ([www.dmoz.org](http://www.dmoz.org)) encontrarás una exhaustiva lista de empresas que se dedican al alojamiento de páginas Web. Simplemente, deberemos especificarles que queremos utilizar PHP y MySQL en un servidor Apache, y ellos se encargará de su instalación y configuración.

## 2.2. Licencia GPL

Como vimos en el capítulo anterior, tanto PHP como MySQL o Apache son *software libre*, es decir, podemos descargarlos y utilizarlos libremente siempre que respetemos la licencia que rige este tipo de *software*, llamada GPL (GNU General Public License, Licencia Pública General). Dicha licencia se creó para proteger la integridad del *software libre* y evitar que nadie restrinja o utilice su circulación.

**Nota:** Las cláusulas completas de esta licencia las podemos encontrar en: <http://www.gnu.org/copyleft/gpl.html>.

Como resumen, diremos que en todo programa o código sujeto a esta licencia se nos permite:

1. Descargar, utilizar y copiar libremente dicho programa, siempre que se incluya el *copyright* correspondiente y todo el texto correspondiente a la licencia.
2. Incluir modificaciones y mejoras en dicho *software* y distribuirlas libremente, siempre que se especifique que cambios se han hecho en el *software* original y la fecha de realización.
3. Dicho *software* debe distribuirse libremente, sin coste alguno. Como mucho, se permite cobrar una cuota que costee los gastos de reproducción (por ejemplo, el CD en el que se graba).

## 2.3. Instalación

Podemos descargar el código completo de PHP, MySQL y Apache, así como completas instrucciones de instalación y diversa documentación, en las siguientes páginas:

- PHP: <http://www.php.net/downloads.php>
- MySQL: <http://www.mysql.com/downloads/index.html>
- Apache: <http://httpd.apache.org/download.cgi>

En todos los casos deberemos elegir la version de instalacion compatible con nuestro sistema operativo, ya sea *Linux* o *Windows*. Es un proceso sumamente sencillo y completamente documentado. Cualquier usuario acostumbrado a trabajar con *Linux* puede terminar el proceso en poco menos de 20 minutos.

Quizá pueda resultar un poco mas complicado para los usuarios que trabajen con sistema operativo *Windows*, dado que los programas diseñados para entornos Microsoft suelen tener interfaces de instalacion mas intuitivas. Afortunadamente para ellos, existe la solucion perfecta: *PHP Home Edition 2*.

## 2.4. ***PHP Home Edition 2***

*PHP Home Edition 2* (incluido en el CD adjunto) es una utilisima aplicacion que instala y configura el lenguaje PHP, la base de datos MySQL y el servidor Apache en ordenadores con el sistema operativo *Windows* en apenas unos minutos, dejandolo perfectamente listo para usar.

Esta aplicacion ha sido desarrollada por Milan Gacik, programador de la Repùblica Checa y apasionado defensor del *software libre*. *PHP Home Edition 2* es utilizada por miles de personas en todo el mundo dada su sencillez. Esta traducida a numerosos idiomas, entre ellos el castellano.

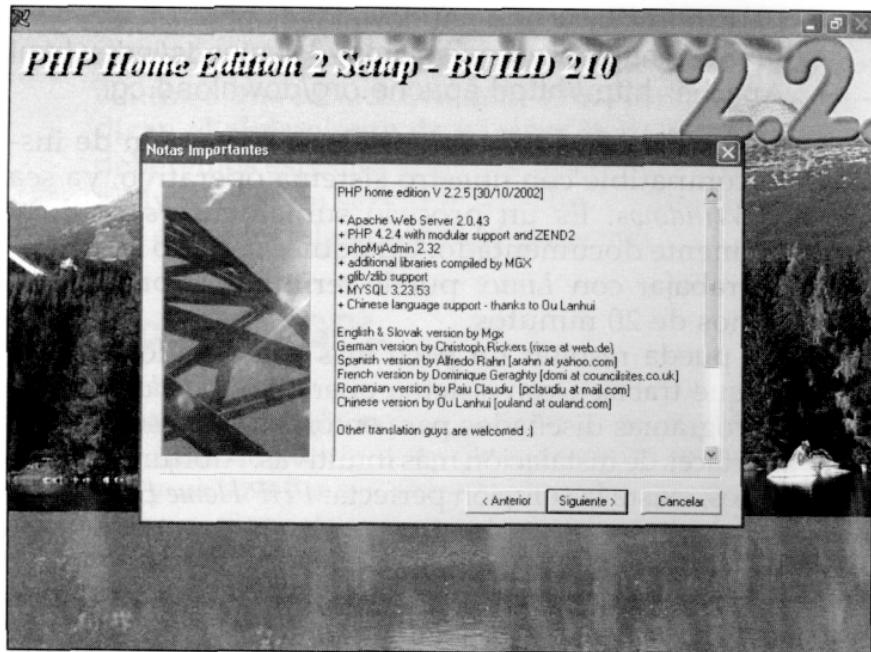
Esta sujet a la licencia GPL, y su pagina Web es: <http://sourceforge.net/projects/phphome/>

### 2.4.1. ***Instalación PHP Home Edition 2***

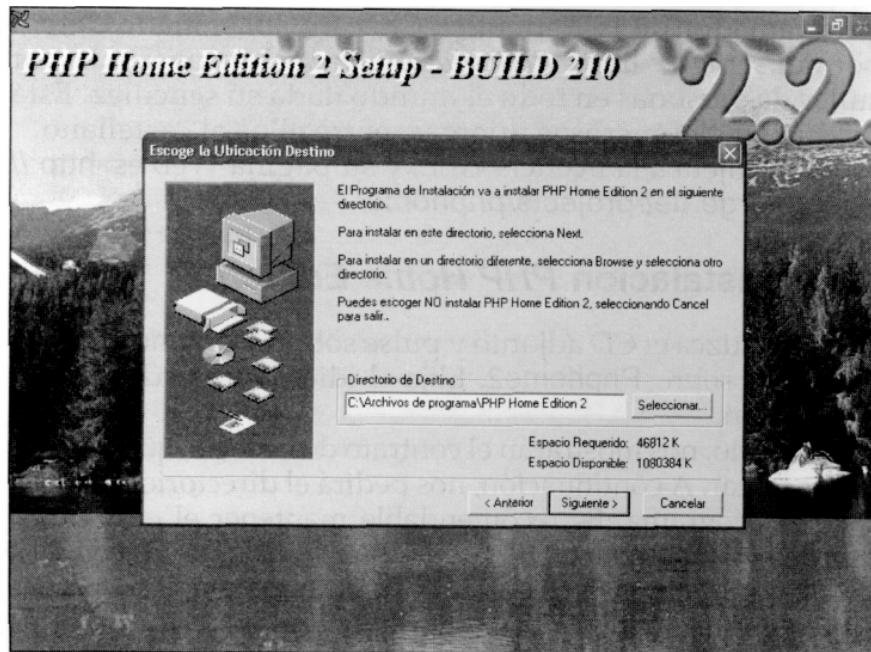
Introduzca el CD adjunto y pulse sobre **Phphome2.2.5full**. Haga clic sobre **Phphome2**. Elija el idioma deseado para la instalacion.

Tras ello, nos mostraran el contrato de licencia, que debemos aceptar. A continuación, nos pedira el directorio para instalar el programa. Es recomendable mantener el que indica por defecto **C:\Archivos de programa\PHP Home Edition 2**.

El siguiente paso sera elegir **Tipo de instalacion** que deseamos. Pulse **Típica**. Tras ello, deberemos verificar todos los datos anteriores. Empezara el proceso de instalacion propiamente dicho.



**Figura 2.1.** Nos aparecerá una pantalla especificando el contenido del programa y las personas que han ayudado a traducirlo.



**Figura 2.2.** Directorio de instalación de PHP Home Edition 2.

*Nota: Importante. Durante el proceso de instalacion se nos pedird el nombre del servidor, de la conexion, y de la contraseña. Apúntelo en lugar seguro, pues nus será necesario en numerosas ocasiones.*

El nombre del servidor deberemos dejarlo tal cual (es decir, localhost). El nombre de la conexion y de la contraseña es a nuestra elección. Apunte cuidadosamente los nombres elegidos en lugar seguro, pues deberemos emplearlos muy a menudo de ahora en adelante.

A continuación, se nos pedira el Nombre del servidor y el Puerto a través del cual se comunica, que dejaremos con los valores por defecto (127.0.0.1 y 80), y nuestra Dirección de correo electronico.

Con ello, practicamente habremos terminado el proceso de instalacion.

El siguiente paso sera configurar el Nombre de usuario y la Contraseña en la interfaz de MySQL. Lo aconsejable, para evitar errores, es utilizar los mismos que utilizamos durante la instalacion. Éste sera el ultimo paso en la instalacion.

## 2.4.2. Configuración y utilización

Tras reiniciar el ordenador, veremos que algunos iconos nuevos han aparecido en nuestro escritorio. Veamos cuáles son y cómo utilizarlos.

- **Iconos del Servidor Apache**  
Encontraremos 3 iconos: uno para iniciar, otro para cerrar y otro para reiniciar el servidor Apache. El servidor se iniciara cada vez que iniciemos el ordenador, asi que no deberemos utilizarlos salvo que haya algun fallo, en cuyo caso, con pulsar el icono de reiniciar bastara para arreglar el problema.
- **Icono Test my PHP**  
Pulsando este icono, se abrirá la ventana del navegador (a partir de ahora, cada vez que abramos el navegador, por defecto, se abrirá la misma pantalla), donde veremos la configuración del PHP. Probablemente, no entienda nada. ¡No se desanime! Nuestro viaje al mundo de las páginas Web dinámicas no ha hecho mas que empezar.

- Icono **phpMyadmin**

Pulsemos en dicho vínculo. Tras introducir el Nombre de usuario y la Contraseña (que establecimos durante el proceso de instalacion), entraremos en el programa propiamente dicho. Esta aplicación, llamada *phpMyAdmin*, es la que utilizarernos para crear y gestionar las bases de datos MySQL.

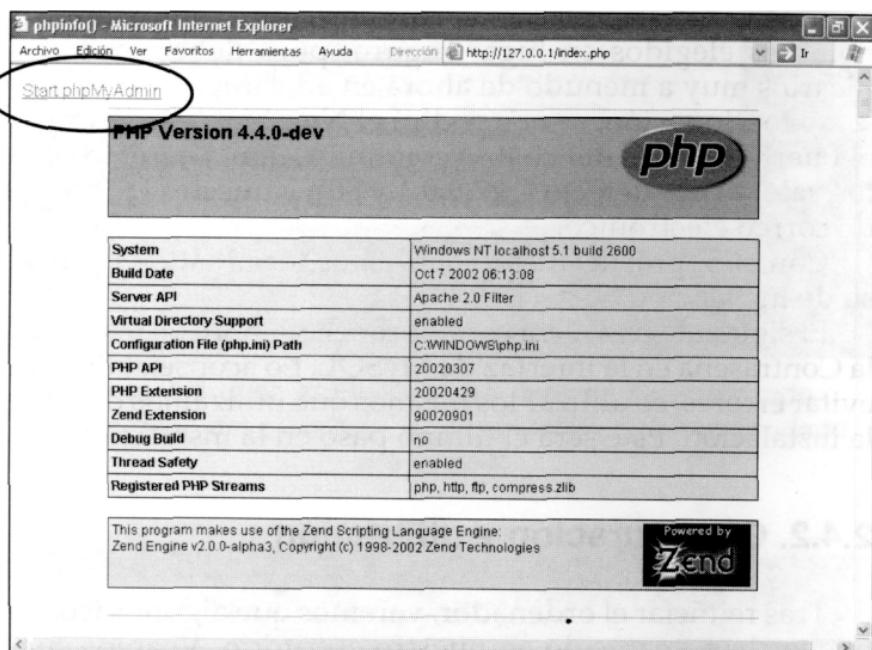


Figura 2.3. Aplicacion phpMyAdmin.

Se trata de otra aplicación de *software libre* que ofrece un entorno mas amigable a la hora de trabajar con las bases de datos. Permite trabajar bien con SQL (el lenguaje propio de muchas bases de datos, entre ellas MySQL), o bien con otro mas sencillo e intuitivo.

En esta guía aprenderemos los fundamentos del lenguaje SQL, lo que nos permitira actuar sobre las bases de datos desde la propia pagina Web, utilizando el lenguaje PHP.

Tras pulsar en la izquierda de la pantalla, donde pone **Bases de datos**, apareceran las dos bases de datos que vienen por defecto: **MySQL** y **Test**. Pulsemos sobre la base de datos **MySQL>SQL**.

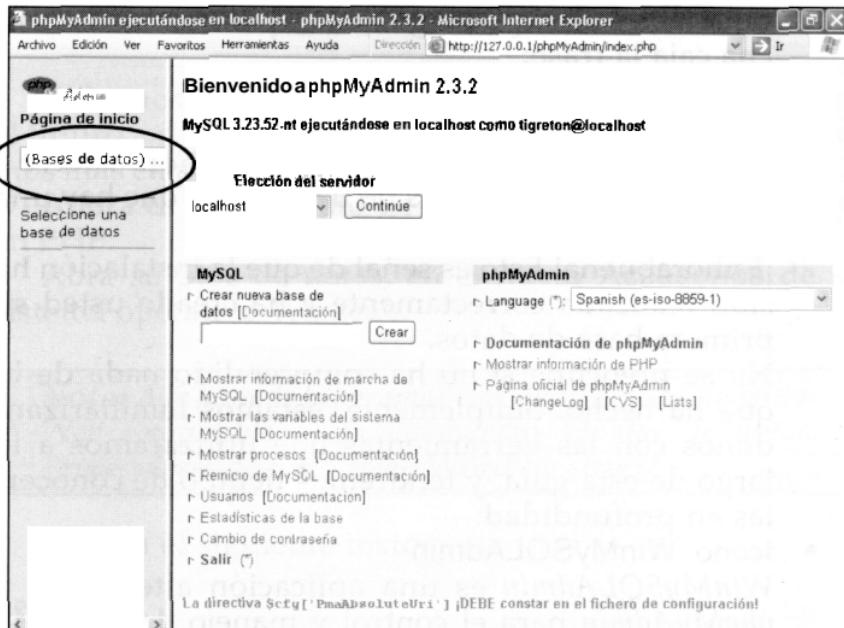


Figura 2.4. Bases de datos en phpMyAdmin.

Tabla	Acción	Campos	Tipo	Tan
<input type="checkbox"/> <b>alumnos</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	21	MyISAM	2
<input type="checkbox"/> <b>authors</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	2	MyISAM	2
<input type="checkbox"/> <b>categories</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>columns_priv</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>db</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	1	MyISAM	3
<input type="checkbox"/> <b>func</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>host</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>jokelookup</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>jokes</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>odiados</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	3	MyISAM	2
<input type="checkbox"/> <b>odiados2</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>populares</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	3	MyISAM	2
<input type="checkbox"/> <b>tables_priv</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
<input type="checkbox"/> <b>user</b>	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	5	MyISAM	2

Figura 2.5. Trabajar con MySQL.

Aparecerá una amplia caja de textos, precedido por la frase **Ejecute la/s consulta/s SQL en la base de datos MySQL [Documentación]**; es aquí donde

podemos utilizar el lenguaje SQL. Escribamos en dicha caja la frase:

**create database alumnos**

Y a continuación, pulse **Continue**. Si ahora vuelve a pulsar en el menu **Bases** de datos, verá que hay una nueva base llamada Alumnos.

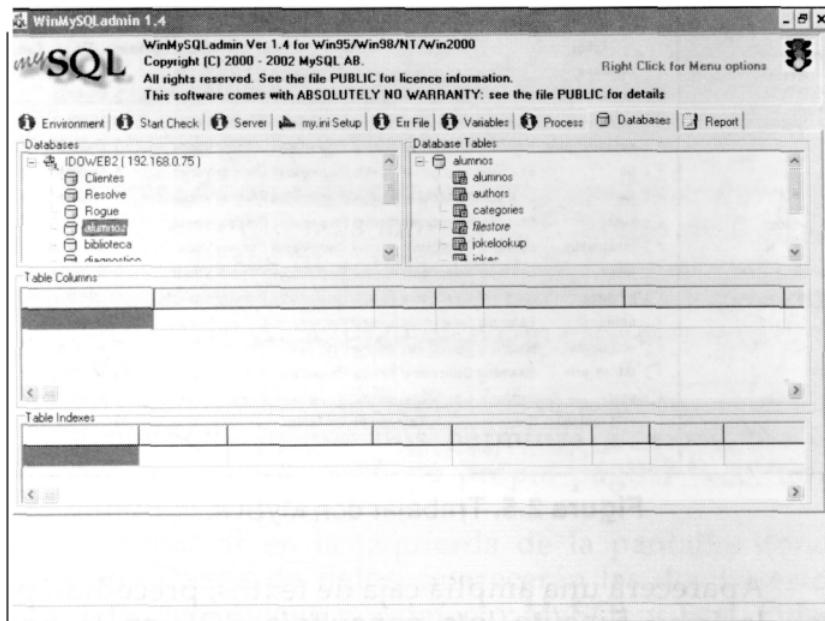
¡Enhorabuena! Esto es señal de que la instalación ha sido realizada correctamente y ha creado usted su primera base de datos.

No se preocupe si no ha comprendido nada de lo que ha hecho. Simplemente, estamos familiarizándonos con las herramientas que utilizaremos a lo largo de esta guía, y tendremos tiempo de conocerlas en profundidad.

- **Icono WinMySQLAdmin**

*WinMySQLAdmin* es una aplicación alternativa a *phpMyAdmin* para el control y manejo de bases de datos MySQL. Aunque es sumamente completa y útil, a efectos didácticos, y durante la presente guía, emplearemos siempre la aplicación *phpMyAdmin*.

A modo de prueba, pulse en el menú superior en la pestafia Databases. Allí encontrara la base de datos Alumnos que creó hace unos instantes.



**Figura 2.6. Localización de la base de datos Alumnos.**

## 2.5. Nuestro primer *script* en PHP

Ya hemos visto las herramientas que vamos a utilizar en el futuro, en especial la aplicación *phpMyAdmin*, que veremos más en profundidad en el capítulo 4.

Ahora es el momento de escribir nuestro primer *script* en PHP.

Abra su bloc de notas, en el menú Accesorios de su sistema operativo *Windows*.

---

*Nota: A la hora de programar con PHP, emplee un editor Web compatible o, en su defecto, el bloc de notas. Procesadores de texto como Word no sirven.*

---

Escriba el siguiente texto:

```
<?
echo "Bienvenido al mundo de las páginas Web
dinámicas"
?>
```

Grabe el documento con el nombre *bienvenido*, con la extensión *.php*. En la pestaña **Tipo**, señale **Todos los archivos**, y en **Codificación** señale **ANSI** en el directorio **C:\Archivos de programa\PHP Home Edition 2\www**.

---

*Nota: Todos los scripts en PHP deben guardarse con la extensión *.php* para que funcionen correctamente.*

---

Abra su navegador e introduzca la siguiente dirección Web: <http://127.0.0.1/bienvenido.php>.

Voila! Este es su primer *script* en PHP. No es demasiado impresionante, ni algo que no se pueda hacer simplemente con HTML, pero es el primer paso de un viaje maravilloso que le llevará a una nueva dimensión en el campo del diseño Web.

# Introducción a PHP

## 3.1. Introducción

Dado que esta guía es eminentemente práctica, daremos un vistazo a las funciones y estructuras esenciales en PHP a través de ejemplos.

Si tiene experiencia en programación con ASP, Java u otros, verá que muchas de ellas son parecidas, cuando no idénticas. En caso de ser neófito, se asombrará de lo sencillo que es crear páginas interactivas con este lenguaje.

Si desea un enfoque más teórico del tema, o simplemente quiere profundizar aún más, recuerde que en [www.php.net](http://www.php.net) encontrará completa información y gente deseosa de ayudarle y compartir experiencias con usted. Asimismo, podrá descargar el manual oficial de PHP en castellano.

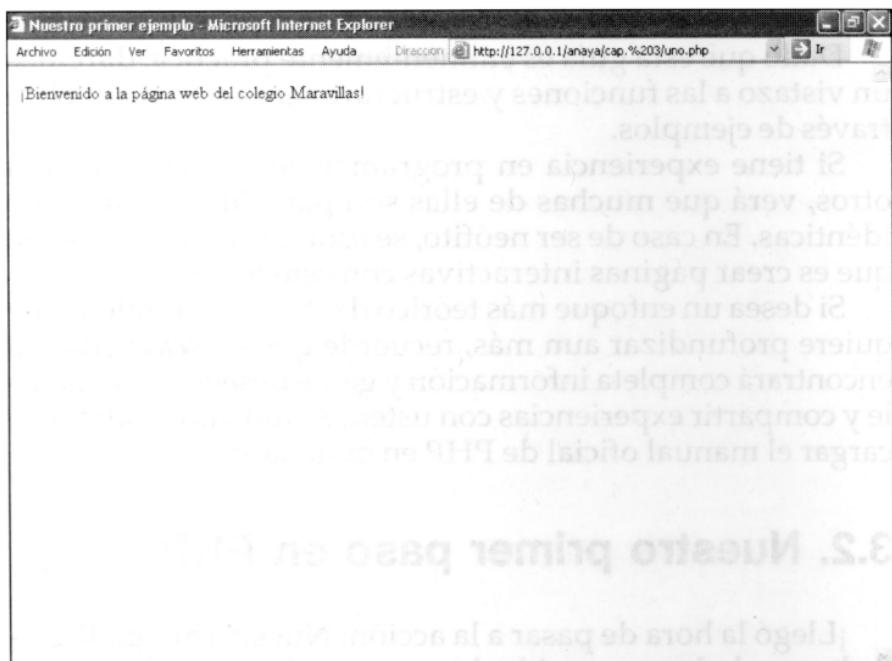
## 3.2. Nuestro primer paso en PHP

¡Llegó la hora de pasar a la acción! Nuestro amigo Ricardo ha quedado convencido de que una página dinámica realizada mediante PHP y MySQL es lo que realmente necesita para su Web de antiguos alumnos del colegio Maravillas.

**Nota:** *Tal y como vimos en el capítulo 2, deberemos escribir nuestros scripts en PHP sobre un editor compatible con PHP, o sencillamente, con la utilidad Bloc de notas de nuestro sistema operativo Windows. Dichos script deberán guardarse con la extensión .php, y mientras trabajemos en nuestro propio ordenador, grabarlo en el directorio.*

### 3.2.1. Apertura y cierre en PHP

```
c!- apertura.php ->
<html>
<head>
<title>Apertura y cierre</title>
</head>
<body>
<?
echo "¡Bienvenido a la página Web del
colegio Maravillas!";
?>
</body>
</html>
```



**Figura 3.1. Primeros pasos con PHP.**

Examinemos el siguiente codigu:

```
<?php echo "¡Bienvenido a la página Web del
colegio Maravillas!" ; ?>
```

Esta es la única línea de código que contiene lenguaje PHP. El resto es una sencilla estructura en HTML. Si no está familiarizado con HTML, recuérdela, pues trabajaremos con ella muy a menudo.

Los caracteres al inicio (<?) y al final de la linea (?) indican donde empieza y donde acaba el codigo PHP. Es imprescindible colocar estos caracteres de apertura y cierre al principio y final de la parte del *script* escrita en PHP para que el navegador sea capaz de distinguir e interpretar la parte escrita en HTML de la escrita en PHP.

---

*Nota: Nuestro navegador interpreta el código de la siguiente manera: <? = aqui empieza el código PHP. ?> = aqui termina el código PHP.*

---

Esta manera de indicar el comienzo (<?) y el cierre (?) del código en PHP se conoce como "estilo abreviado", y es el mas habitual entre los programadores.

La instruccion echo es la que se utiliza en PHP para indicar que se debe mostrar un texto en pantalla. El texto a mostrar, en este caso "¡Bienvenido a la pagina Web del colegio Maravillas!", debe ir escrito entre comillas y la sentencia finalizada con un punto y coma.

## Otros metodos de indicarlo

- **Estilo XML:** <?php . . . . ?>  
Si tiene pensado implementar XML (*Extensive Markup Lenguaje*) en su sitio Web, deberia usar este tipo de simblos.
- **Estilo ASP:** <% . . . . %>  
Llamado asi por ser idéntico al usado en ASP (*Active Server Pages*). Utilicelo solo si usa un editor orientado hacia ASP o si, por haber programado ya en ese lenguaje, le resulta mas sencillo acordarse.
- **Estilo Script:** <script language='php'> . . . . </script>  
Les resultara familiar a aquellos que hayan trabajado con JavaScript. Utilicelo solo si trabaja con un editor de HTML que le de problemas con el resto de estilos.

### 3.2.2. Como se comenta el codigo

Otro elemento a tener en consideración cuando trabajemos con PHP es la posibilidad de comentar nuestro código. Es decir, incluir frases aclaratorias para cualquier persona (o nosotros mismos) que pueda leer el código, pero

que, al ejecutarlas el intérprete de PHP, las obvie y no nos dé error.

La forma mas comun, que sera muy familiar a los usuarios acostumbrados a trabajar con C++, consiste en incluir dos barras ( // ) al final de la linea y, a continuación, el comentario que deseemos hacer.

Veamoslo con un ejemplo:

```
echo "¡Bienvenido!"; // esta sentencia muestra  
en pantalla "Bienvenido".
```

De esta manera podremos hacer mas entendible el código a cualquiera que lo lea, sin interferir en su ejecucion.

### 3.2.3. Como indicar la fecha en PHP

Hasta ahora no hemos hecho nada que no pueda hacerse solamente con HTML. Es hora de que empecemos a explorar las nuevas caracteristicas de PHP.

Añadamos la siguiente sentencia al script anterior:

```
<!-- fecha.php -->  
<html>  
<head>  
<title>Fecha</title>  
</head>  
<body>  
    c? echo "¡Bienvenido a la página Web del  
colegio Maravillas!.  
    La fecha actual es"; <br>  
    echo date ("H:i, jS F Y");  
    ?>  
</body>  
</html>
```

date () es otra de las funciones predefinidas en PHP. "H:i" indica la hora en formato de 24 horas. "jS" indica los minutos en formato de dos cifras, "F" indica el dia del mes en formato de una cifra, e "Y" es el año en formato de 4 cifras. Desgraciadamente, todo ello escrito en ingles.

**Nota:** Si quiere ver la lista de todos los formatos soportados por la función date(), mirar en [www.php.net](http://www.php.net).



**Figura 3.2. Resultado del script.**

Hay un par de cosas a tener en cuenta con este *script*:

1. Muestra un resultado verdaderamente dinamico. La hora y el dia cambian en funcion del reloj y el calendario. Esto es algo que no se puede hacer solamente con HTML.
2. La fecha y la hora que muestra es indicada por el servidor en donde alojemos esta página. Es por ello que al PHP se le llama un "lenguaje del lado del servidor", en contraposicion a otros como JavaScript, conocidos como "lenguaje del lado del cliente". En nuestro caso, la fecha mostrada en pantalla es independiente de la fecha y hora que el usuario tenga en su ordenador. Si un *script* similar se hubiese realizado con JavaScript, la hora y fecha que se mostraria seria la que el usuario tenga en su ordenador.

Veamoslo más claro con un ejemplo:

Imaginemos el caso de un nostálgico que, conmocionado por la desaparicion de Elvis, ha configurado su ordenador para que la fecha del mismo sea siempre la de la muerte del Rey del Rock: 16de Agosto de 1977.

Cuando este individuo entre en una pagina que indique la fecha mediante un *script* en JavaScript, este le mostrara la

fecha de su ordenador (**16** de agosto de 1977). Mientras que si ese *script* hubiese sido realizado mediante PHP, lo que aparecería sería la fecha indicada en nuestro servidor, es decir, la correcta (sobre todo, si tenemos en cuenta que ¡Elvis vive!).

### 3.3. Variables

Las variables son un concepto matemático fundamental en cualquier lenguaje de programación.

Si no está familiarizado con ellas, podríamos representarlas con una caja vacía. La variable es la caja, inicialmente vacía, y en ella podemos meter el valor que queramos.

Veamos un ejemplo:

```
$nombre = Pedro
```

En PHP toda variable debe ir precedida por el signo del dolar \$. En este caso, hemos definido la variable Nombre (nuestra caja vacía), y le hemos dado el valor de *Pedro* (hemos metido *Pedro* en la caja).

Veamos el siguiente *script*:

#### 3.3.1. Primer contacto con variables

```
<? $nombre = Pedro;
echo $nombre; ?>
```

Este sencillo *script* presentará en pantalla la palabra *Pedro*. Podemos sustituir *Pedro* por cualquier otra palabra o, incluso, por un número, y será éste el que se vea en pantalla. Eso es debido a que la variable es una simple caja que depende del valor que nosotros le asignemos.

---

**Nota:** El nombre de la variable es sensible a mayúsculas y minúsculas.

---

#### 3.3.2. Mayúsculas y minúsculas

Veamos más características de las variables con un nuevo ejemplo:

```
<? $a = "Pedro ";
$A = "Juan";
echo $a . $A ;?>
```

Este código mostrará en pantalla *Pedro Juan*, es decir, ha considerado \$a y \$A como dos variables absolutamente diferentes. Hay que tenerlo muy en cuenta cuando escribamos código para evitar errores.

La variable puede contener cualquier tipo de dato, ya sea cadena de texto, números o cualquier otro valor, y como tal, se puede operar con ellas.

### 3.3.3. Adición de variables

```
c? $a = 4;  
$b= 9;  
echo $a + $b; ?>
```

Este *script* mostrará en pantalla el valor **13**.

## 3.4. Operadores

Cuando trabajemos con variables, usaremos fundamentalmente dos tipos de operadores: aritméticos y de texto.

### 3.4.1. Operadores aritméticos

Los mismos que utilizamos en cualquier operación matemática sencilla.

Nombre	Operador	Ejemplo
Adición	+	\$a + \$b
Sustracción	-	\$a - \$b
Multiplicación	*	\$a * \$b
División	/	\$a / \$b

Tabla 3.1. Operadores aritméticos.

### 3.4.2. Operadores de texto

Utilizado para unir dos cadenas de texto. También se le conoce como Operador de concatenación.

```
<? $a = "Bienvenido al ";  
$b = "Colegio Maravillas";  
$c = $a.$b;  
echo $c; ?>
```

Mediante este concatenado, hemos asignado a la variable \$c el contenido de las dos cadenas de texto \$a y \$b. Por ello, la instrucción **echo \$c;** nos mostrara en pantalla la frase "Bienvenido al Colegio Maravillas".

## 3.5. HTML y PHP. Formularios

Con los conocimientos que hemos adquirido hasta ahora, estamos en disposición de comenzar con nuestro proyecto de pagina Web.

Ricardo quiere colocar en su pagina un formulario de entrada escrito en HTML que recoja los datos del usuario y pasar esos datos a una pagina escrita en PHP que de una bienvenida personalizada al visitante.

El problema se le presenta a la hora de pasar los datos que el visitante introduce a la pagina en PHP.

### 3.5.1. bienvenida.html

Veamos cómo podemos ayudarle:

```
<!-- bienvenida.html -->
<form action="saludo.php" method=post>
<html>
<head>
<title>Bienvenida</title>
</head>
<body bgcolor="#FFFFFF">
    <div align="center">
        <p>&nbsp;</p>
        <p><b>Bienvenido a mi p&aacute;gina Web</b></p>
        <form name="form1">
            <p>Nombre : <input type="text"
name="nombre"></p>
            <p>Apellido: <input type="text"
name="apellido"></p>
            <p>A&ntilde;o en que naciste: <input type=
"text"
name="nacimiento"></p>
            <input type="submit" value="enviar"></p>
        </form>
    </div>
</body>
</html>
```

The screenshot shows a Microsoft Internet Explorer window with the title 'Untitled Document - Microsoft Internet Explorer'. The address bar displays the URL 'http://127.0.0.1/anaya/cap.%203/bienvenida.html'. The page content is a simple HTML form with the following text:  
Bienvenido a mi página web  
Nombre  
Apellido  
Año en que naciste

Figura 3.3. Formulario de bienvenida.

### 3.5.2. saludo.php

```
<!-- saludo.php -->
<html>
<head>
<title>Saludo</title>
</head>
<body>
    C?
    echo "¡Bienvenido $nombre $apellido a la
página de antiguos
    alumnos del colegio Maravillas! ";
    $año= 2002;
    $edad = $año - $nacimiento;
    echo " Tienes $edad años. ¡Una edad estupenda!";
    ?
</body>
</html>
```

Probemos el funcionamiento de estos dos programas. El visitante introduce su nombre, apellidos y año de nacimiento en el formulario bienvenida.html. Esto crea tres variables distintas: **Nombre**, que es donde (lógicamente) va almacenado

el nombre de nuestro visitante; **Apellido**, donde se almacena el apellido (¡quién lo hubiera dicho!); y **Nacimiento**, que almacena el año de nacimiento de la persona.

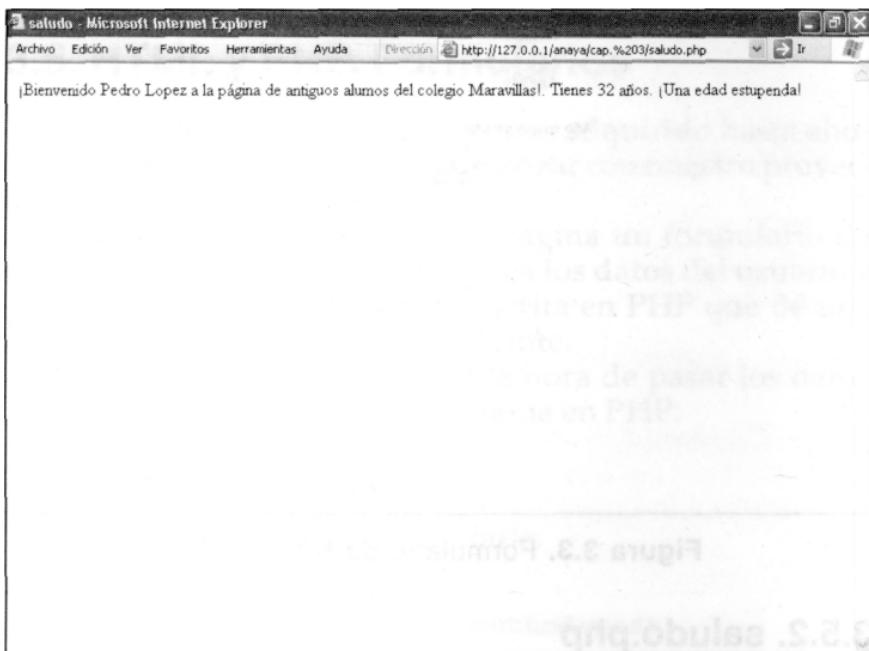


Figura 3.4. Aplicacion dinamica para nuestra pagina.

Miremos mas detenidamente la primera linea del código:

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

Para el ordenador del usuario, esta linea vendría a decir algo así: "Cuando pulse el botón **Enviar**, debo abrir el *script* llamado **saludo.php**, y enviarle las variables **Nombre**, **Apellido** y **Nacimiento**".

Todo ello viene indicado por `form action="saludo.php"`. (Es bien sabido que los ordenadores son parcos en palabras.)

La otra mitad de la frase `method=post`, viene a decir la manera en que estas variables son enviadas al *script* **saludo.php**.

POST viene a significar que se debe mandar mediante un paquete independiente, en contraposición a GET, que indica que se deben mandar como un añadido al final de la URL.

¿Cómo interpreta "saludo.php" esta información? Como hemos visto, en PHP todas las variables deben ir precedidas por el signo `$`. Dado que el valor de estas variables (el contenido de la caja) viene definido vía

POST desde el *script* anterior, lo unico que tenemos que hacer es mostrarlo en pantalla (usando **echo**), junto con un pequeño mensaje de bienvenida.

Por otro lado, tambien hemos practicado con el tema de los operadores aritmeticos: definimos la variable \$año con el valor del año actual (en el momento de escribir este libro, 2003). Simplemente, restamos al año actual la fecha de nacimiento del visitante, y ya tenemos su edad.

De manera sencilla, hemos visto como personalizar las paginas para cada usuario definiendo unas variables (en este caso con su nombre, pero pueden incluir sus preferencias, su dirección...), y hemos aprendido a enviarlas a una pagina en PHP que las reutiliza para crear un mensaje absolutamente personal y unico para cada visitante.

Alguien podría decir: "Estupendo, pero sería todavía mejor si pudiese almacenar toda esa información en alguna parte, para llevar un registro de la gente que me visita". No se impacienten. Todo llegará. En este caso, en el capítulo 5, cuando veamos como PHP y MySQL pueden trabajar juntos.

## 3.6. Estructuras de control

Nos será muy útil, ahora que sabemos como funcionan las variables, conocer un tipo de estructuras que nos permitirán personalizar aún más nuestras páginas y hacerlas todavía más dinámicas. Esto se conoce como estructuras de control. Si tiene conocimientos de algún otro lenguaje de programación, ya sea Java, Perl, C, C++, verá que en PHP funcionan de manera casi idéntica. Para los que no, con unos sencillos ejemplos veremos como funcionan.

### 3.6.1. If-else

La más frecuente es la conocida como **if-else**. Para los que no dominen el inglés, *if* significa un sí condicional, y *else* es la opción alternativa. "Si hace sol me voy a la playa, y si no me quedo en casa" sería un perfecto ejemplo de la estructura **if-else** aplicado a la vida real. Apliquemos esta estructura a nuestro ejemplo anterior.

### 3.6.2. clase.php

Supongamos que Ricardo estuviese muy interesado en contactar con sus compañeros de curso, que previsiblemente

nacieron el mismo año que el, 1970. (Claro que esta el tema de los repetidores, pero como Ricardo siempre fue un poco empollon, no quiere saber nada de ellos.)

Solo habria que modificar el *script clase.php* de la siguiente manera:

```
<!-- clase.php -->
<html>
<head>
<title> saludo </title>
</head>
<body>
<?
echo ";Bienvenido $nombre $apellido a la
pagina de antiguos
alumnos del colegio Maravillas!. ";
$ano= 2003;
$edad = $ano - $nacimiento;
if ( $edad == 33)
{
echo "Tú y yo fuimos a la misma clase. Por
favor, escribeme a
ricardo@maravillas.com. ";
}
else
{
echo " Tienes $edad años. ¡Una edad estupenda!";
}
?>
</body>
</m>
```

Con este cambio, incluimos la condición de que si el visitante nacio en 1970, el mensaje que le salga sea el de ponerse en contacto con Ricardo. En caso contrario, le aparecera el mensaje habitual.

Habrá notado que hemos duplicado el signo de igualdad en la linea

```
if ( $edad == 33)
```

Esto debe hacerse cada vez que establezcamos una condicion (si la variable edad es igual a 33). Si solo hubiésemos puesto un signo de igualdad, lo unico que hubiesemos logrado seria asignar a \$edad el valor de 33, independientemente del que tuviese antes.



Figura 3.5. Aplicación para encontrar compañeros de curso.

### 3.6.3. While

La sentencia **while**, en castellano, mientras, es otra de las que debemos aprender a la perfección, pues la emplearemos muy a menudo a lo largo de este libro.

Con esta sentencia, obligamos a realizar una acción mientras determinada condición, establecida por **while**, se cumpla. Esto se conoce como "bucle", y es empleada frecuentemente en cualquier lenguaje de programación, entre ellos, por supuesto, PHP.

### 3.6.4. bucle.php

Veamos el ejemplo más habitual y usado de bucle:

```
<?
$cuenta = 1;
while ($cuenta <=5)
{ echo" $count ";
$cuenta++;
}
?>
```

El resultado de este bucle muestra en pantalla, de manera sucesiva, los numeros del 1 al 5.

Examinémoslo con detenimiento.

```
$cuenta=1; // Creamos una variable llamada cuenta y
le asignamos el valor "1"
while ($cuenta <=5)
{ echo" $count "; // mientras la variable cuenta
sea menor o igual de 5, se muestra en pantalla.
$count++ // El doble signo de adición significa
en PHP (sumar uno).
```



**Figura 3.6.** Utilización de bucles en nuestras páginas.

De esta manera, nuestro bucle particular va sumando una unidad mas a la variable cuenta,, y es mostrada en pantalla. Cuando la variable cuenta se hace mayor de cinco, deja de cumplirse la condición **while**.

### **3.6.5. Elseif**

Una ampliacion de la estructura de control **if-else** que nos permite dar cabida a mas de dos posibilidades.

Veamoslo con un ejemplo. Anteriormente, Ricardo ha creado un aviso personalizado a todos los alumnos del colegio

que nacieron en su mismo año. Dado que el colegio Maravillas se cerro en 1980, cabe suponer que todo el que haya nacido despues de ese año es imposible que haya ido al mismo colegio. Asimismo, Ricardo siempre se llevo muy mal con los del curso inmediatamente superior al suyo, los nacidos en 1969, y no quiere volver a saber nada de ninguno de ellos. Para ofrecer un mensaje personalizado a cada uno de ellos, vamos a emplear la estructura `elseif`.

Añadamos algunas líneas al *script* `saludo.php`.

```
<!-- saludo.php -->
<html>
<head>
<title>Saludo</title>
</head>
<body>
  <?
  echo "¡Bienvenido $nombre $apellido a la
  pagina de antiguos
  alumnos del colegio Maravillas!. ";
  $año= 2003;
  $edad = $año - $nacimiento;
  if ( $edad == 33)
  {
    echo "Tú y yo fuimos a la misma clase. Por
    favor, escribeme a
    ricardo@maravillas.com. ";
  }
  elseif ($edad <=23)
  {
    echo " Eres demasiado joven para haber
    asistido al colegio
    Maravillas. Lo siento, esta página no es de
    tu interés";
  }
  elseif ($edad ==34)
  {
    echo "¡Tú y todos los de tu curso siemgre
    fuisteis un atajo ae
    matones. Fuera de mi pagina inmediatamente!";
  }
  ?>
</body>
</html>
```

### 3.7. Ejercicio práctico

¿Se ve capacitado a mejorar lo anterior? Modifique los *scripts* **bienvenida.html** y **saludo.php** de tal manera que introduzca nuevas preguntas al visitante, como el telefono o la dirección de correo electronico. Si tiene conocimientos de HTML, modifique ambos codigos para hacerlos mas atractivos visualmente.

# Introducción a MySQL

## 4.1. Introducción a las bases de datos

En el capítulo anterior hemos visto alguna de las características que convierten a PHP en uno de los lenguajes más populares a la hora de crear páginas Web dinámicas, pero también hemos visto algunas de sus carencias, sobre todo a la hora de almacenar y recuperar información.

Es por ello que sus posibilidades máximas se obtienen empleandolo junto con una base de datos como MySQL.

En este capítulo estudiaremos via ejemplos que es una base de datos y las estructuras y funciones básicas en MySQL.

## 4.2. ¿Qué es y para qué sirve una base de datos?

Aunque el concepto pueda asustarnos un poco, las bases de datos informáticas tienen por objeto hacernos las cosas mucho más sencillas. Su objetivo es almacenar la información que le suministramos y ordenarla en base a criterios que nos harán más útil consultarla cuando nos haga falta.

Veamoslo con un ejemplo práctico:

Supongamos que por fin nos hemos decidido a ordenar nuestros libros. Durante años hemos ido almacenandolos sin orden ni concierto en la biblioteca y perdemos un tiempo precioso cada vez que queremos encontrar un título determinado.

Así que hemos decidido invertir una cantidad de tiempo en ordenarlo de tal manera que en el futuro sepamos como encontrar cada título.

El primer problema al que nos enfrentamos es ¿qué criterio utilizamos para ordenarlos?

Un criterio lógico sería el del orden alfabetico por título. De esta manera se podría encontrar cualquier libro en cuestión de segundos. Pero ¿qué sucede si lo que necesitamos es encontrar la bibliografia completa de, por ejemplo, Benito Perez Galdos? Entonces tendremos que buscar por toda la biblioteca, y en caso que desconozcamos alguno de sus titulos, nos sera casi imposible encontrarlo.

Entonces quiza optemos por ordenarlos por autor. Pero, ique sucede si necesitamos encontrar un libro del que solo recordamos el titulo y no el autor? Si la biblioteca es muy grande, nos volveremos locos. ¿Y si el libro esta escrito por varios autores? ¿Dónde lo colocamos entonces? Para colmo, resulta que algunos estantes de la biblioteca son muy estrechos y solo caben libros de bolsillo. Por otro lado, dado que nos encanta la literatura del Siglo de Oro espaiiol, nos planteamos incluso ordenar los libros cronologicamente y por país...

Para volverse realmente loco. Estamos a punto de tirar la toalla, de dejar los libros en un perenne desorden y nos preguntamos: ¿por qué no existira una biblioteca que se ordene automaticamente en base al criterio que le pidamos? Es decir, que cuando busquemos un autor, se ordene automaticamente por autores.

Si al dia siguiente lo buscamos por titulo, que se ordene por orden alfabetico... ¿Imposible? Nada es imposible en el mundo de la informática. Una base de datos como MySQL puede, en nuestro ordenador, presentar la informacion en base a los criterios que nosotros le pidamos: alfabeticamente, por orden de entrada, por tamaño, por precio, etc.

Cuando trabajemos con cantidades grandes de informacion (y un sitio Web dinamico suele tenerlas), es imprescindible guardar todos esos datos en una base de datos, de tal manera que cada persona que visite nuestra Web pueda encontrar la informacion que realmente necesita de la manera en que **el** la necesita. Es por eso que necesitamos dominar y entender este concepto.

## 4.2.1. Ordenemos la biblioteca

Hemos llegado a la conclusion de que lo mejor sera utilizar nuestro ordenador y una base de datos para ordenar los libros.

El primer paso sera determinar que criterios utilizaremos para consultar la información. Esto lo haremos definiendo una tabla como la del dibujo.

<i>ID</i>	<i>Título</i>	<i>Autor</i>	<i>Editorial</i>	<i>Idioma</i>
1	<i>Cruzando el paraíso</i>	Shepard, Sam	Anagrama	Castellano
2	<i>La tabla de Flandes</i>	Pérez-Reverte, Arturo	Plaza & Janés	Castellano
3	<i>Todos los nombres</i>	Saramago, José	Santillana	Castellano
4	<i>Flash MX (edición especial)</i>	Hernández, Claudio	Anaya Multimedia	Castellano
...				

**Tabla 4.1.** Ordenacion lógica de la biblioteca.

Esta tabla esta dividida en filas y columnas. En cada fila va colocado un titulo diferente. En cada columna se indica un tipo de dato de cada libro que posteriormente podremos utilizar para consultar los libros (autor, titulo, fecha). La columna ID no es más que un identificador unico que asignamos a cada titulo para evitar confusiones con otro del mismo autor o con mismo titulo. Aunque parezca un tema baladí, la columna ID sera imprescindible en cualquier tabla que realicemos.

¿Siguiente paso? Pegar en el lomo de cada libro una etiqueta con su correspondiente y exclusivo ID, y ordenarlos de manera correlativa. La próxima vez que busquemos un libro, solo tendremos que consultar la base de datos de nuestro ordenador, y nos indicara cual es el numero que busquemos.

## 4.3. Primeros pasos con MySQL

Como hemos visto en el capítulo dos, podemos manejar nuestra base de datos MySQL a traves del programa *phpMyAdmin*.

Gracias a él podriamos, de manera sencilla e intuitiva, crear bases de datos, modificarlas o borrarlas simplemente haciendo clic. Desgraciadamente, cuando PHP se comunica con MySQL lo hacen en un lenguaje particular, conocido

como SQL (*Structured Query Language*, lenguaje de búsqueda estructurado), por lo tanto deberemos conocer los fundamentos de este lenguaje para ser capaces de escribir *scripts* en PHP que recojan o modifiquen la información almacenada en la base de datos.

SQL es un lenguaje estandar que se utiliza con la mayor parte de las bases de datos, como por ejemplo *Microsoft SQL Server*.

Aunque el nombre sea tan parecido, son terminos bien distintos: MySQL es un tipo de base de datos que alojamos en nuestro servidor. SQL es el lenguaje que esta y muchas otras bases de datos utilizan.

#### 4.3.1. Crear una base de datos

En la esquina superior izquierda encontrara la frase **Selecciona Una Base de Datos**, y una pestaña encima. Si la pulsamos, encontraremos la base de datos existentes actualmente en nuestro servidor.

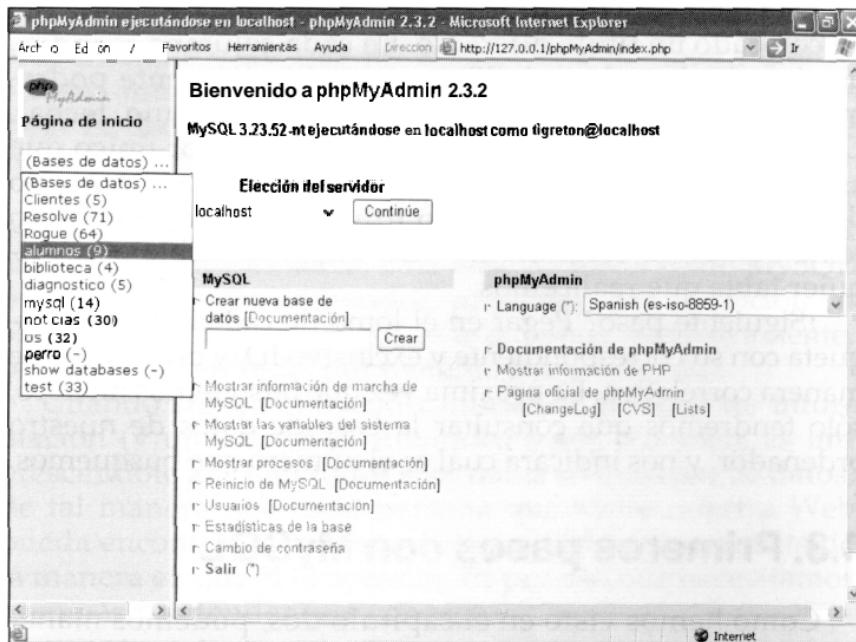


Figura 4.1. Imagen de la interfaz SQL.

Si el servidor es nuestro y esta recién instalado, aparecerán dos nombres: **MySQL** y **Test**. Si hemos contratado un servidor a alguna empresa de alojamiento, serán ellos quienes nos hayan creado la base de datos, y no podremos

crear bases nuevas. Si ese es su caso, puede saltarse este paso e ir directamente a Crear tablas en nuestra base de datos. Cualquier nueva base de datos deberá encargarlo a la empresa que aloja su Web.

Pulse entonces en la base Test. Le aparecerá una pantalla como ésta:



**Figura 4.2.** Tablas en la base de datos.

Pulse en la segunda pestaña superior por la izquierda **SQL**, y llegará a la interfaz donde podrá escribir el código SQL.

Crear una base de datos es así de sencillo. Escriba:

```
CREATE DATABASE alumnos;
```

Si ahora miramos de nuevo en la pestaña de la esquina superior izquierda, encontraremos una nueva palabra, Alumnos.

Pulse, y estará en nuestra nueva base de datos, Alumnos.

### 4.3.2. Crear tablas en nuestra base de datos

Una vez dentro de nuestra base de datos, podemos crear infinitas tablas (aunque no es aconsejable).

En el capítulo 7 veremos más a fondo como el crear diversas tablas puede ayudarnos a procesar mejor la información. De momento, incluiremos todos los datos en una sola tabla.

La estructura a la hora de crear una tabla es la siguiente:

```
CREATE TABLE nombre_de_la_tabla (
  nombre_columna1 tipo_columna1 detalles_columna1,
  nombre_columna2 tipo_columna2 detalles_columna2,
  nombre_columna3 tipo_columna3 detalles_columna3,
);
```

## Hagamos la prueba con un ejemplo práctico

Ricardo quiere crear una tabla en la que ira introduciendo el nombre, primer apellido, fecha de nacimiento y dirección de correo electrónico de los alumnos del colegio Maravillas.

<i>ID</i>	<i>Nombre</i>	<i>Apellido</i>	<i>Fecha</i>	<i>Correo</i>
1				
2				
3				
...				

Y el codigo en SQL que debemos emplear es:

```
CREATE TABLE alumnos (
  ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nombre TEXT,
  apellido TEXT,
  nacimiento DATE,
  email TEXT
);
```

Veamos detenidamente cada una de las líneas:

- CREATE TABLE alumnos (. Indica el nombre de la tabla que queremos crear. No importa que tenga el mismo nombre que la base, aunque puede crear confusiones.
- ID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,. La primera columna se llamara ID, que como hemos visto es el identificador. INT indica que se trata de un numero entero positivo. NOT NULL obliga a que siempre tenga un valor, no admitiendose el cero. AUTO\_INCREMENT es una util sentencia que nos librara de tener que asignar este valor cada vez que introduzcamos un nuevo alumno en la tabla, ya que le asignara el anterior ID +1, y PRIMARY KEY

indica que ese ID debe ser unico para cada alumno, y nunca repetirse.

- **nombre TEXT.** Esta es mas sencilla. Creamos una nueva columna llamada Nombre que contendrá texto (TEXT, en inglés).
- **nacimiento DATE.** Una nueva columna llamada Nacimiento. En este caso, se almacenaran fechas (DATE, en inglés).
- **email TEXT.** Sin mayor complicación. La columna Email también contendrá texto.

### 4.3.3. Sentencias SHOW y DESCRIBE

Es fácil llevar el control de nuestra base de datos cuando, como en este caso, solo hemos creado una tabla. Sin embargo, a medida que vayamos adentrandonos en esta guia, iremos creando mas tablas, y necesitaremos sentencias para saber cuantas y de qué tipo son las tablas que hemos creado. Para ello existen SHOW y DESCRIBE.

En la zona de la interfaz destinada para el SQL, escribamos la siguiente sentencia.

SHOW TABLES;

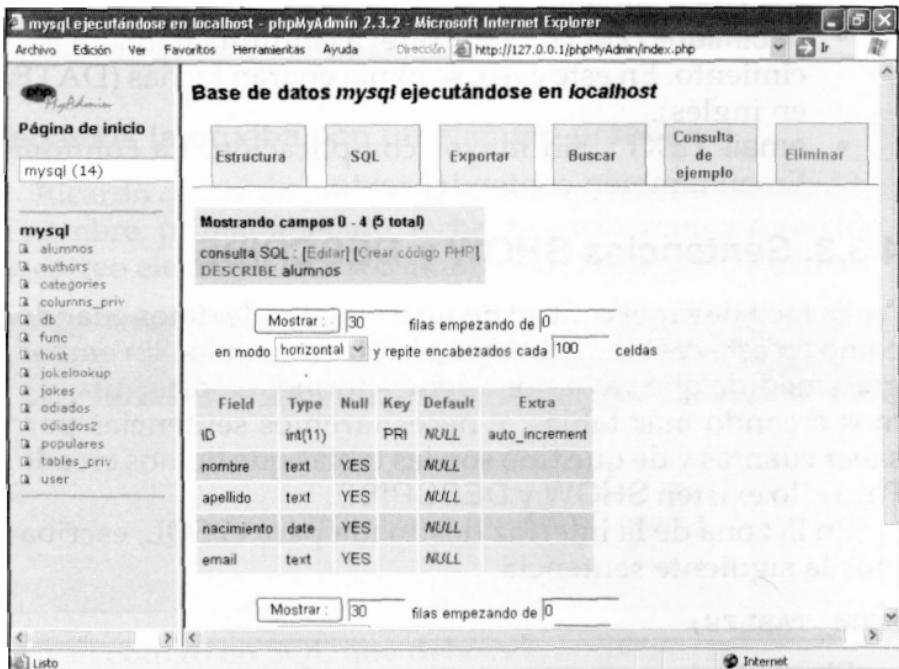
El resultado sera este:

The screenshot shows the phpMyAdmin interface for MySQL. The left sidebar shows the 'mysql' database with 14 tables: alumnos, authors, categories, columns\_priv, db, func, host, jokelookup, jokes, odiodos, odiodos2, populares, tables\_priv, and user. The main content area has a 'consulta SQL' (SQL query) box containing 'SHOW TABLES'. Below it, a table titled 'Tables\_in\_mysql' lists the same 14 tables. The 'db' table is currently selected in the list.

Figura 4.3. Ejemplo de sentencia SHOW.

Y, si queremos saber algo mas de esta tabla, escribamos la siguiente sentencia:

```
DESCRIBE ALUMNOS;
```



Field	Type	Null	Key	Default	Extra
ID	int(11)		PRI	NULL	auto_increment
nombre	text	YES		NULL	
apellido	text	YES		NULL	
nacimiento	date	YES		NULL	
email	text	YES		NULL	

Figura 4.4. Ejemplo de sentencia DESCRIBE.

Esta sentencia indica cuantos son los campos en nuestra base de datos y cuales son sus caracteristicas. Cuando uno se enfrenta por primera vez a una base de datos no diseñada por uno mismo, conviene utilizar estas sentencias para conocer su estructura y caracteristicas.

#### 4.3.4. Sentencia DROP

Se utiliza esta sentencia para eliminar las tablas en nuestra base de datos. Si escribiersemos (¡pero no lo haga todavía!):

```
DROP ALUMNOS;
```

La tabla desapareceria en un santiamén. Así de sencillo ¡y así de peligroso! A diferencia de otros programas, MySQL no muestra un mensaje de confirmación a la hora de borrar una tabla. La borra sin mas. ¿Imaginan que en dicha tabla tuviésemos almacenados cientos de datos de **alumnos**? Por ello es conveniente realizar copias de seguridad de nuestras tablas, cosa que veremos en el capítulo 7.

### 4.3.5. Insertar datos en las tablas

Conociendo las sentencias básicas para crear, examinar y borrar tablas, es el momento de insertar datos en ellas (pues para eso han sido creadas).

Imaginemos que Ricardo quiere utilizar esta tabla para introducir los datos de todos los antiguos alumnos de su colegio con los que vaya contactando. De momento, solo tiene sus datos. ¿Cómo hacemos para meterlos en la tabla? El comando a emplear es **INSERT** y puede emplearse de dos maneras muy similares.

La más frecuente es:

```
INSERT INTO nombre-tabla SET
columnal = 'valor1',
columna2 = 'valor2',
columna3 = 'valor3';
```

Otra manera alternativa e igualmente efectiva es:

```
INSERT INTO nombre-tabla
(columnal, columna2, ...)
VALUES (valor1, valor2, ...);
```

Veamos como aplicamos ambos métodos al caso de Ricardo:

```
INSERT INTO alumnos SET
nombre = 'Ricardo',
apellido = 'Manzano',
nacimiento = '1970-12-11',
email='ricardo@maravillas.com';
```

Obien:

```
INSERT INTO alumnos
(nombre, apellido, nacimiento, email) VALUES(
'Ricardo', 'Manzano', '1970-12-11',
'ricardo@maravillas.com');
```

---

*Nota: No ha sido necesario dar ningún valor al campo ID. Tal y como lo hemos configurado, le asignara siempre un valor único por nosotros. MySQL enfiende la fecha en formato anglosajón (año-mes-día), en contraposición al que estamos acostumbrado (día-mes-año).*

---

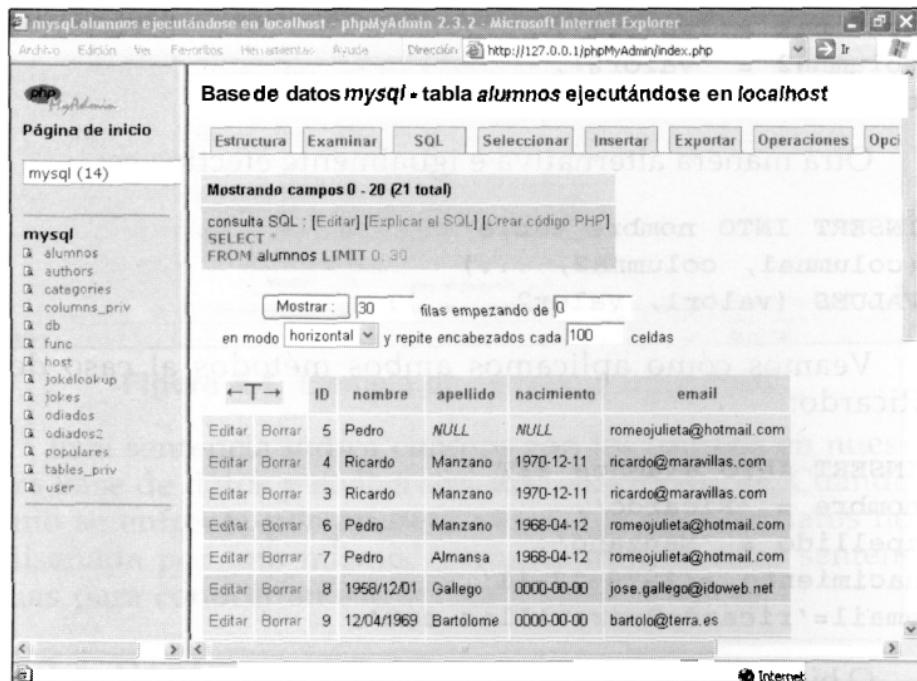
#### 4.3.6. Examinar los datos de nuestra base: Comando SELECT

Dada la gran cantidad de opciones del comando **SELECT**, se convierte en uno de los más útiles y a la vez complicados de entender.

Veamos un ejemplo. Escribamos la siguiente línea en nuestra interfaz de SQL:

```
SELECT * FROM alumnos;
```

El signo “\*” significa toda la base de datos. Es por ello que al pulsar **Sigue**, nos aparecerá la siguiente pantalla:



The screenshot shows the phpMyAdmin interface for a MySQL database named 'mysql'. The left sidebar lists tables: 'alumnos', 'authors', 'categories', 'columns\_priv', 'db', 'func', 'host', 'joke lookup', 'jokes', 'odiodos', 'odiodos2', 'populares', 'tables\_priv', and 'user'. The main area is titled 'Base de datos mysql - tabla alumnos ejecutándose en localhost'. It shows a table with 21 rows. The SQL query in the 'SQL' tab is:

```
SELECT *  
FROM alumnos LIMIT 0, 30
```

The table has columns: ID, nombre, apellido, nacimiento, and email. The data is as follows:

ID	nombre	apellido	nacimiento	email
5	Pedro	NULL	NULL	romeojulieta@hotmail.com
4	Ricardo	Manzano	1970-12-11	ricardo@maravillas.com
3	Ricardo	Manzano	1970-12-11	ricardo@maravillas.com
6	Pedro	Manzano	1968-04-12	romeojulieta@hotmail.com
7	Pedro	Almansa	1968-04-12	romeojulieta@hotmail.com
8	1958/12/01	Gallego	0000-00-00	jose.gallego@idoweb.net
9	12/04/1969	Bartolome	0000-00-00	bartolo@terra.es

Figura 4.5. Ejemplo de sentencia SELECT.

Hagamos una búsqueda más específica de la siguiente manera:

```
SELECT ID, nombre FROM alumnos;
```

En este caso, la información contenida en pantalla se restringe a estos dos campos.

Esta es la utilización más sencilla y común de este comando. De manera genérica, la estructura de **SELECT** es la siguiente:

```
SELECT campos
FROM tabla/s
[WHERE condicion]
[GROUP by tipo]
[HAVING definición]
[ORDER BY criterio]
[LIMIT limite] ;
```

Introduzcamos algunos datos mas y hagamos algunos ejemplos.

Primero insertamos mas datos tal y como ya sabemos con el comando **INSERT**.

```
INSERT INTO alumnos SET
nombre = 'Pedro',
apellido = 'Almansa',
nacimiento ='1968-04-12',
email='almansa15@hotmail.com';
```

Pruebe usted a introducir mas datos alternando las dos maneras de utilizar la sentencia **INSERT**.

Ahora pidamosle que nos muestre los datos segun el nombre por orden alfabetico.

```
SELECT * FROM alumnos
ORDER BY nombre;
```

Si quisieramos hacer una busqueda especifica, haríamos lo siguiente:

```
SELECT * FROM alumnos
WHERE apellido= Almansa;
```

Si, por ejemplo, quisiesemos saber el nombre y el apellido de aquellos alumnos que nacieron a partir de determinada fecha:

```
SELECT nombre, apellidos FROM alumnos
WHERE nacimiento >='1968-01-01';
```

¿Ve lo util que resulta esta sentencia? Con algo asi se podría ordenar la biblioteca de la que hablabamos al principio del capitulo en un santiamen. ¡Y esto no es mas que una introducción! El comando **SELECT** nos va a resultar esencial a la hora de preparar nuestra pagina Web dinamica.

Otra sentencia que utilizaremos muy a menudo es el comando **COUNT**.

Gracias a COUNT (contar, en inglés), podremos saber cuantas son las entradas en nuestra base de datos que cumplen el criterio de búsqueda que hemos solicitado.

Hagamos la primera prueba. Escribamos en la interfaz de SQL:

```
SELECT COUNT(*) FROM alumnos;
```

Como respuesta, nos dará el número de entradas que hemos metido en nuestra base de datos.

Y si quisieramos saber cuantos alumnos en nuestra base de datos se llaman Ricardo, utilizamos:

```
SELECT COUNT(*) FROM alumnos WHERE nombre='Ricardo';
```

#### 4.3.7. Modificar elementos en nuestra base de datos

Cuando se trabaja con bases de datos grandes, es increíblemente fácil cometer errores. A veces se nos olvida una letra, equivocamos las filas, erramos en los acentos... Es por ello que existen sentencias como UPDATE (en inglés, actualizar), que nos sirve para seleccionar y definir con precisión los campos que debemos cambiar.

Su estructura es la siguiente:

```
UPDATE nombre_tabla SET  
nombre_columna=nuevo_valor,  
WHERE condiciones;
```

Imaginemos que Pedro Almansa ha cambiado su dirección de correo electrónico. Pensaba que su anterior dirección `almansa15@hotmail.com` dice poco sobre su forma de ser, y ha optado por `romeobuscajulieta@hotmail.com`. ¿Cómo procedemos a cambiarlo?

Introduzcamos el siguiente código:

```
UPDATE alumnos SET email=  
'romeobuscajulieta@hotmail.com' WHERE nombre=  
'Pedro';
```

¿Qué problema nos puede dar esta sentencia? Que cambiará el correo electrónico de todos aquellos cuyo nombre sea Pedro, y es sencillo que en una base de datos muy amplia haya varias personas con ese nombre. Podríamos restringir la búsqueda indicando WHERE nombre='Pedro'

**AND apellido='Almansa';** pero ni de esa manera podemos estar seguros que no haya existido otro alumno con ese nombre y apellido.

Es por ello que hemos creado el campo ID. Por sus características (recordemos que al crearlo indicamos que era PRIMARY KEY, es decir, Génico), sabemos que es un campo que no se repite nunca, y que es propio y exclusivo de cada alumno. Por ello, a la hora de hacer cualquier cambio, actualización o eliminación, debemos regirnos por este campo.

Si, por ejemplo, sabemos que el ID de Pedro es el 3, escribiremos la frase de la siguiente manera:

```
UPDATE alumnos SET email=
'romeobuscajulieta@hotmail.com' WHERE ID='3';
```

## 4.4. Repaso a las sentencias basicas

Demos un repaso a las sentencias basicas que hemos visto hasta ahora:

<i>Sentencia</i>	<i>Características</i>
CREATE	Permite crear nuevas bases de datos o tablas
SHOW	Nos muestra las bases de datos o tablas existentes.
DESCRIBE	Nos muestra la estructura de las bases de datos y tablas existentes
DROP	Permite eliminar las tablas y bases de datos existentes.
SELECT	Permite seleccionar determinadas entradas de las tablas.
INSERT	Permite insertar nuevos registros en las tablas.
UPDATE	Permite modificar valores ya existentes en las tablas.
DELETE	Permite eliminar valores ya existentes en las tablas.

**Tabla 4.2.** Sentencias basicas.

Vistas ya las sentencias basicas en PHP y MySQL, ha llegado el momento que venimos esperando durante todo el libro: utilizar ambos lenguajes de manera conjunta para crear páginas Web dinamicas.

# Acceder a los datos en MySQL a traves de la Web gracias a PHP

¡Por fin llegamos al momento que todos esperabamos!

Hasta ahora, hemos aprendido como instalar y configurar correctamente PHP y MySQL. Conocemos las sentencias fundamentales de ambos, y nos disponemos a crear nuestra primera pagina Web dinamica.

Pero antes, conviene hacer una pequena recapitulación.

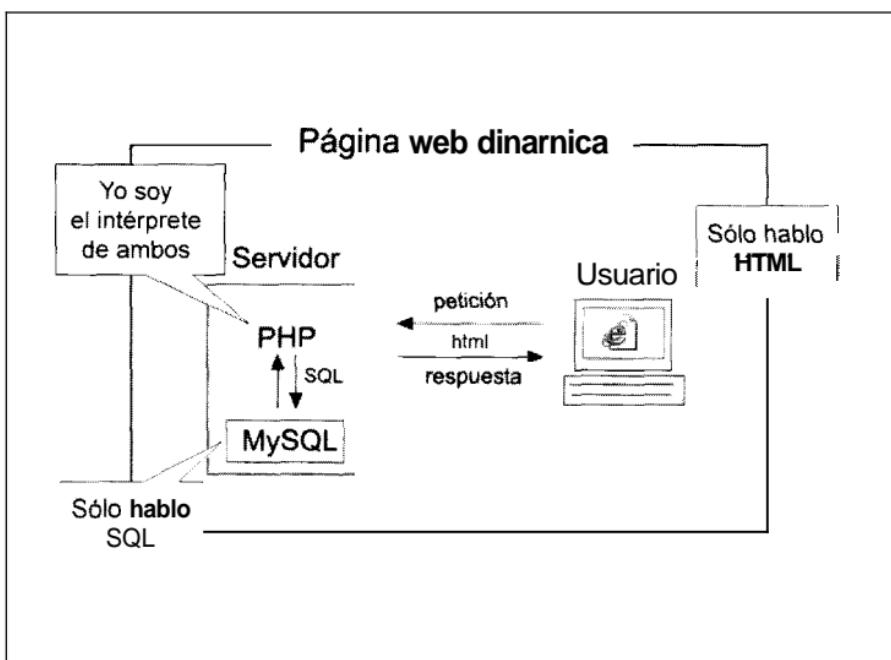
## 5.1. Como funciona una pagina Web dinamica

Tenemos claro el funcionamiento de una pagina Web dinamica. La informacion se encuentra almacenada de manera ordenada en una base de datos conocida como MySQL. Esta base de datos se aloja en un servidor, en nuestro caso Apache. Sabemos que esta base de datos es la manera ideal de ordenar la informacion, y en ella esta todo el contenido de nuestra pagina Web. Conocemos tambien las sentencias basicas de PHP, un potente lenguaje "del lado del servidor" especialmente diseñado para la creación de paginas Web y que puede incluirse con suma sencillez dentro del codigo HTML.

Por otro lado, tenemos un usuario que, via Internet, entra en nuestra pagina. Para ello utiliza un navegador (*Internet Explorer* y *Netscape* en su mayor parte) preparado para recibir paginas Web en HTML.

El problema esta en como "hacer que se entiendan" ambos. Sabemos que nuestra base de datos MySQL utiliza un lenguaje llamado SQL, que no tiene nada que ver

con la pagina HTML estandar que nuestro explorador necesita para mostrar al usuario.



**Figura 5.1.** Funcionamiento de una pagina **Web** dinamica.

Para intermediar entre ambos es para lo que existe PHP, un potente lenguaje "del lado del servidor" especialmente diseñado para la creación de páginas Web y que puede incluirse con suma sencillez dentro del código HTML.

Dado que puede integrarse fácilmente en una página HTML, de tal manera que "engaña" al explorador, tenemos solucionado el problema por ese lado. Ahora solo nos queda saber cómo PHP puede "entenderse" con MySQL. Y eso es lo que vamos a aprender en este capítulo.

## 5.2. Como conectar PHP con MySQL

Afortunadamente, entre las funciones incluidas en PHP hay una que hace esta tarea sumamente fácil.

Su estructura es la siguiente:

```
mysql_connect(dirección, nombre_de_usuario,  
contraseña);
```

Veamos cada uno de los terminos por separado:

- **direccion.** Indica la direccion IP o el dominio en el que la base de datos MySQL esta alojada. En el caso de que hayamos instalado tanto la base de datos como el servidor en nuestro propio ordenador para hacer pruebas, esta palabra deberá sustituirse por localhost. En el caso de que hayamos contratado espacio en servidor en alguna empresa de alojamiento Web, deberán darnos este dato, aunque lo mas normal (de hecho es asi en el 90 por 100 de los casos) es que tambien ellos instalaran MySQL en el mismo servidor en que este alojada nuestra pagina, por lo que tambien sera localhost.
- **nombre\_de\_usuario.** Si lo tenemos instalado en nuestro propio ordenador, el nombre de usuario lo habremos definido durante el proceso de instalacion. En caso de que hayamos contratado espacio Web en alguna empresa, deberan ser ellos quienes nos faciliten este dato.
- **contraseña.** Lo mismo que en el caso anterior.

En algunos casos, esta funcion se representa asi:

```
mysql_pconnect(direccion, nombre_de_usuario,  
contraseña);
```

La unica diferencia es que `mysql_pconnect()` crea una conexion persistente, y `mysql_connect()` crea una conexion normal, es decir, que se cierra cuando el *script* finaliza su ejecucion o utilizamos la funcion `mysql_close()` para cerrar la conexion. Una conexion persistente permaneceria abierta en cualquiera de los casos anteriores.

### 5.2.1. Conexión

Veamos un *script* completo de PHP que realiza una conexion con MySQL:

```
$bd=@mysql_connect ("localhost", "ricardo",  
"maravillas");  
if (!$bd) {  
echo ("Error. No se pudo conectar con la base de  
datos en este  
momento. Inténtelo más tarde);  
exit();  
}
```

Veamos este *script* con mayor detenimiento:

```
$bd=@mysql_connect("localhost", "ricardo",
"maravillas");
```

¿Recuerdan cuando hablamos en el capítulo 2 de las variables en PHP? Dijimos que eran semejantes a una caja vacía en la que podíamos incluir cualquier cosa, ya fuese un número, una cadena de texto... Pues, en este caso, hemos ido un poco más lejos: hemos definido una variable (con el signo \$) y le hemos dado el valor de la conexión a la base de datos. Ello nos va a simplificar enormemente la tarea de escribir código.

Otra cosa a destacar es la inclusión del signo "@" justo delante de `mysql_connect()`. Ello es debido a que si la conexión falla, aparecería un mensaje de error confuso y en inglés, y con el signo @ delante evitamos esto, y podemos a continuación escribir nuestro propio mensaje de error en castellano.

```
if (!$bd) {
echo ("Error. No se pudo conectar con la base de
datos en este
momento. Intentelo más tarde");
exit();
}
```

El comando `if` ya lo estudiamos en el capítulo 3. La única diferencia es incluir el signo "!" delante de la variable. Esto sirve para invertir el operador condicional.

Veámoslo con un ejemplo:

```
if(x)=3 {
echo ("x=3");
}
```

En este caso estamos indicando que si `x` tiene un valor de tres, debe aparecer en pantalla "x=3".

Pero si escribimos:

```
if(!x)=3 {
echo ("x no es igual a 3");
}
```

En este caso estamos indicando que si `x` no es igual a 3, debe aparecer en pantalla "x no es igual a 3".

En el *script* de nuestro ejemplo, `if(!$bd)`, lo que indica es lo que hay que hacer en el caso de que no funcione la conexión con la base de datos, es decir, mostrar en pantalla el texto: "Error. No se pudo conectar con la base de datos en este momento. Intentelo mas tarde", y terminar el programa gracias al comando `exit()`.

## 5.3. Seleccionar una base de datos

En caso de que la conexión sí haya funcionado, el siguiente paso sería seleccionar una base de datos entre todas las que hayamos creado.

Para ello, emplearemos la sentencia:

```
mysql_select_db(nombre_base_de_datos);
```

Podriamos continuar el *script* anterior, seleccionando nuestra base de datos de la siguiente manera:

```
mysql_select_db("alumnos");
if(! @mysql_select_db("alumnos") )
{
echo( "Error. No se puede acceder a la base de
datos en este momento. Intentelo más tarde");
exit();
}
```

## 5.4. Búsquedas en la base de datos

Para seleccionar los datos de nuestra base de datos en base a determinado criterio, utilizamos la función:

```
mysql_query(criterio_de_búsqueda)
```

Sigamos completando nuestro *script* anterior añadiendo las siguientes líneas:

```
$busqueda= @mysql_query ("SELECT nombre AND apellido
FROM alumnos")
if (!busqueda) {
echo ("Error al seleccionar los elementos de la
base de datos. Intentelo más tarde");
exit();
}
```

Hemos creado la variable **\$búsqueda**, y le hemos asignado el valor de una búsqueda (¡era de esperar!) mediante **mysql\_query()**, en el que seleccionamos los campos nombre y apellido de la tabla **alumnos**.

En caso de que por cualquier motivo dicha búsqueda falle, nos aparecerá el mensaje "Error al seleccionar los elementos de la base de datos. Intentelo mas tarde".

## 5.5. Mostrar los datos en pantalla

Una vez conectados a la base de datos, seleccionada la tabla. Hecha la búsqueda, el siguiente paso es recoger dichos datos en la página Web. Para ello, empleamos dos estructuras bien definidas:

1. Un bucle (como ya estudiamos en el capítulo 3), empleando el operador condicional **while**.
2. La sentencia **mysql\_fetch\_array()**, cuya misión es seleccionar todos (y solamente) los campos seleccionados por la búsqueda que realizamos anteriormente con **mysql\_query()**.

El funcionamiento sería el siguiente: **mysql\_fetch\_array()** selecciona todos los campos definidos mediante la variable **\$busqueda**, y los muestra en pantalla gracias a la sentencia **echo**. Cuando se terminan los campos seleccionados, es cuando termina el bucle, y así evitamos que aparezcan en pantalla los datos no seleccionados.

Para presentar mejor los resultados al usuario, hemos incluido dentro de la sentencia **echo** los caracteres **<p>** y **</p>**, que nos sirven para crear saltos de línea, y un pequeño espacio " ", para evitar que nombre y apellido salgan juntos.

```
while ($row = mysql_fetch_array($busqueda))  
{  
echo("<p>" . $row["nombre"] . " " . $row ["apellido"]  
. "</p>");  
}
```

### 5.5.1. Aplicación práctica

Ahora que conocemos el significado de cada línea, veamos el código en su integridad.

```
<html>  
<head>
```

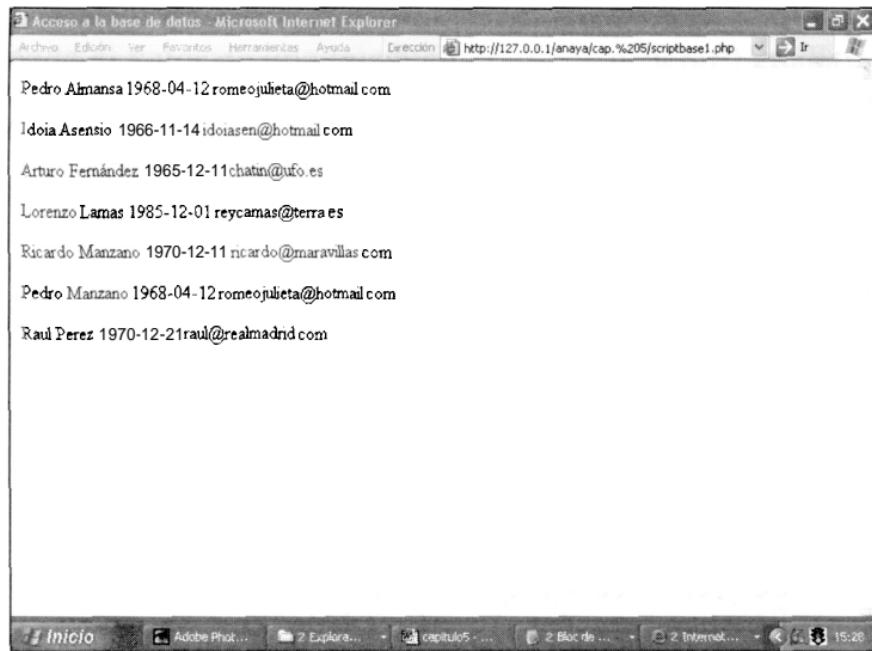
```

<title>Acceso a la base de datos</title>
</head>
<body>
<?php
$bd=@mysql_connect("localhost", "ricardo",
"maravillas");
if (!$bd) {
echo ("Error, No se pudo conectar con la
base de datos en este
momento, Intentelo más tarde");
exit();
}
$sel=@mysql_select_db("alumnos");
if (!$sel)
{
echo( "Error, No se puede acceder a la base
de datos en este
momento, Intentelo más tarde");
exit();
}
$busqueda= @mysql_query("SELECT nombre,
apellido FROM
alumnos");
if (!$busqueda)
{
echo ("Error al seleccionar los elementos
de la base de datos,
Intentelo más tarde");
exit();
}
while ($row = mysql_fetch_array($busqueda))
{
echo("<p>" . $row["nombre"] ." ". $row
[ "apellido"] . "</p>");
}
?>
</body>
</html>

```

Grabelo con la extensión .php, y acceda a él a través de tu navegador.

¡Enhorabuena! Ha creado su primera página Web dinámica empleando PHP y MySQL. Sin embargo, aun nos queda mucho por aprender.



Pedro Almansa 1968-04-12 romeojulieta@hotmail.com  
Idoia Asensio 1966-11-14 idoiasen@hotmail.com  
Arturo Fernández 1965-12-11 chatin@ufo.es  
Lorenzo Lamas 1985-12-01 reycamas@terra.es  
Ricardo Manzano 1970-12-11 ricardo@maravillas.com  
Pedro Manzano 1968-04-12 romeojulieta@hotmail.com  
Raul Perez 1970-12-21 raul@realmadrid.com

**Figura 5.2.** Ejemplo de consulta en la base de datos.

## 5.6. Colegio Maravillas (primera version)

Con lo que ha aprendido hasta ahora, Ricardo se siente preparado para realizar una primera versión de su página Web.

Las funcionalidades que quiere implementar en la página son las siguientes:

1. Un *script* que muestre la lista de antiguos alumnos ordenados por orden alfabético, o por año de nacimiento.
2. Un formulario donde el antiguo alumno introduzca sus datos y estos queden almacenados en la base de datos.
3. Una herramienta de búsqueda donde cualquiera pueda buscar compañeros introduciendo su apellido.
4. Una votación con los profesores más populares de aquella época.

### 5.6.1. apellido.php

*Script* que muestre la lista de antiguos alumnos ordenados por orden alfabético o por año de nacimiento.

```
<!-- apellido.php -->
<html>
<head>
<title>Acceso a la base de datos</title>
</head>
<body>
<?php
$bd=@mysql_connect("localhost", "ricardo",
"maravillas");
if (!$bd) {
echo ("Error, No se pudo conectar con la
base de datos en este
momento, Intentelo más tarde");
exit();
}
$sel=@mysql_select_db("alumnos");
if (!$sel)
{
echo( "Error, No se puede acceder a la
tabla "alumnos" en este
momento. Intentelo más tarde");
exit();
}
$busqueda= @mysql_query("SELECT nombre,
apellido, nacimiento,
email FROM alumnos order by apellido");
if (!$busqueda)
{
echo ("Error al seleccionar los elementos
de la base de datos.
Intentelo más tarde");
exit();
}
while ($row = mysql_fetch_array($busqueda))

echo("<p>" . $row["nombre"] . ". $row
["apellido"] . " ". $row
["nacimiento"] . " ". $row ["email"] .
"</p>");
}
?>
</body>
</html>
```

Este *script* no es mas que una simple aplicacion de todas las sentencias que hemos visto hasta el momento.

La unica linea que incorpora novedades es esta:

```
$busqueda= @mysql_query("SELECT nombre, apellido,
nacimiento, email FROM alumnos order by apellido");
```

Gracias a ella, a la vez que realizamos la busqueda, ordenamos los datos por apellido siguiendo el orden alfabetico (**order by apellido**). Si, por ejemplo, quisiesemos ordenarlo por fecha de nacimiento, lo sustituiriamos por **order by nacimiento**.

## 5.6.2. alta.php

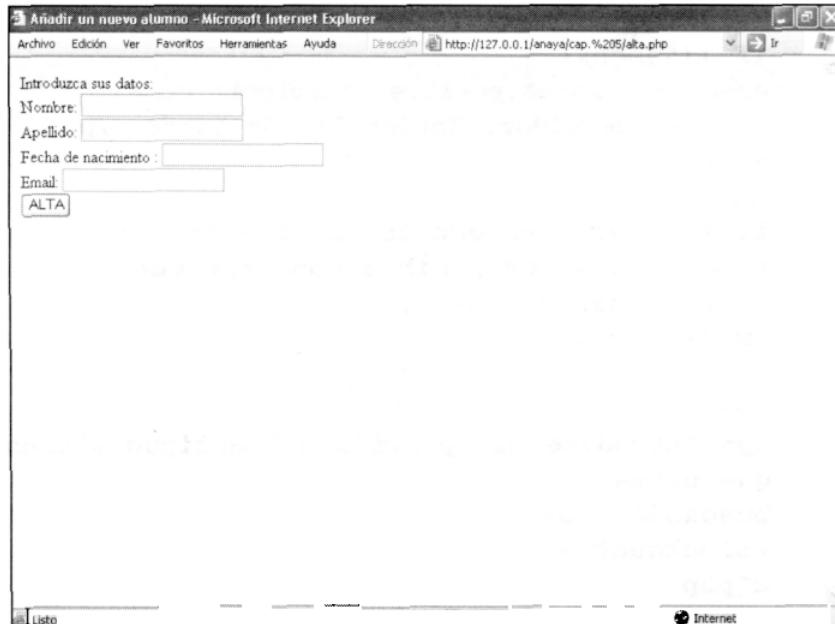
Un formulario donde el antiguo alumno introduzca sus datos y estos queden almacenados en la base de datos.

```
<!-- alta.php -->
<html>
<head>
<title> Añadir un nuevo alumno </title>
</heads
<body>
<?php
if ($submit) :
$dbcnx = mysql_connect("localhost", "ricardo",
"maravillas");
mysql_select_db("alumnos");
$sql = "INSERT INTO alumnos SET
nombre='$nombre',
apellido='$apellido',
nacimiento='$nacimiento',
email='$email'      ";
if (@mysql_query($sql)) {
echo("<p>Alta dada satisfactoriamente</p>");
} else {
echo("<p>Error al darse de alta.: " .
mysql_error() . "</p>");}
?>
<p><a href=<?=
$PHP_SELF?>">Añadir un nuevo alumno.</a></p>
<?php
else:
?>
```

```

<form action="<?=$PHP_SELF?>" method="post">
<p>Introduzca sus datos:<br />
Nombre: <input type="text" name=
"nombre" size="20"
maxlength="255" /><br />
Apellido: <input type="text" name=
"apellido" size="20"
maxlength="255" /><br />
Fecha de nacimiento (aaaa/mm/dd):
<input type="text"
name="nacimiento" size="20" maxlength=
"255" /><br />
Email: <input type="text" name=
"email" size="20" maxlength="255"
/><br />
<input type="submit" name="submit" value=
"ALTA" /></p>
</form>
<?php endif; ?>
</body>
</html>

```



**Figura 5.3.** Ejemplo de formulario de alta.

Este script consta de un pequeño formulario en HTML que recoge los datos introducidos por el usuario, los convierte en

variables, y se los envía a si mismo (<form action=""-<?=\$PHP\_SELF?>" method="post">), para que la parte en PHP ejecute el alta. El siguiente paso es: cuando el usuario pulse el botón de alta (if(\$submit):), conectar con la base de datos, con la tabla "alumnos", e insertar dichos valores en ella. Finalmente, creamos un vínculo de nuevo a la misma página (<p><a href="<?=\$PHP\_SELF?>">Añadir un nuevo alumno.</a></p>), por si se quiere dar de alta a un nuevo alumno.

### 5.6.3. buscador.php

Una herramienta de búsqueda donde cualquiera pueda buscar compañeros introduciendo su apellido.

```
<!-- buscador.php -->
<html>
<head>
<title> Buscador antiguos alumnos </title>
</head>
<body>
<?php
$dbcnx = @mysql_connect("localhost",
"ricardo", "maravillas");
if (!$dbcnx) {
echo( "<p>No es posible establecer conexión " .
"con el servidor. Inténtelo más tarde.</p>" );
exit();
}
if (! @mysql_select_db("alumnos") ) {
echo( "<p>No es posible conectar con la
base de datos.</p>" );
exit();
}
?>
<p> Introduce el apellido del antiguo alumno
que estas
buscando:</p>
<blockquote>
<?php
$result = @mysql_query ("SELECT ID, nombre,
apellido, nacimiento,
email FROM alumnos WHERE apellido LIKE
'$busca'");
if (! $result) {
```

```

echo("<p>Error performing query: " .
mysql_error() . "</p>") ;
exit() ;
}
while ( $row = mysql_fetch_array($result) ) {
echo("<p>" . $row["nombre"] . " " .
$row["apellido"] . " " . $row
["email"] . " " . $row["nacimiento"] . "</p>") ;
}
$busca=0
?>
<form action=<?=$PHP_SELF?>" method="post">
<p><br />
Apellido: <input type="text" name=
"busca" size="20"
maxlength="255" /><br />
<input type="submit" name="submit" value=
"BUSCA" />
</p>
</form>
</body>
</html>

```



**Figura 5.4. Ejemplo de buscador.**

Muy similar al ejercicio anterior. En este caso, utilizamos el formulario html para recoger el nombre del apellido que queremos buscar, y lo convertimos en la variable "\$busca". A continuación, conectamos con la base de datos y establecemos una búsqueda en la tabla "alumnos" con la siguiente restricción: "WHERE apellido LIKE '\$busca'", es decir, de todos los datos almacenados en la base de datos, solo cogeremos aquellos en que el campo apellido coincida con el dato suministrado por el usuario, y los mostramos en pantalla.

## 5.6.4. queridos.php

Una votación con los profesores más populares.

En primer lugar, deberemos crear una nueva tabla en nuestra base de datos **Alumnos**, introduciendo el siguiente código en la interfaz SQL de la aplicación *phpMyAdmin* (ver Cap. 2).

Veamos como llevarlo a cabo:

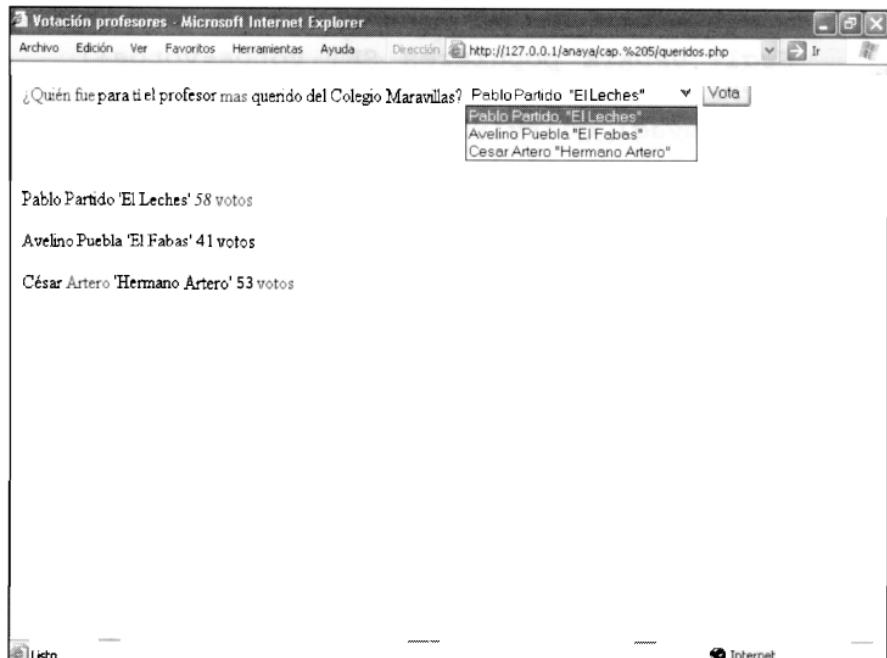
El primer paso será definir las bases de datos que vamos a usar. En el capítulo 7 estudiaremos más a fondo el tema de las relaciones entre bases de datos, y aprenderemos como cruzar datos entre diversas tablas. Hasta entonces, nos limitaremos a crear tres tablas separadas e independientes.

La primera, ya la conocemos. Se llama **alumnos** y contiene los campos **nombre**, **apellido**, **fecha de nacimiento** y **correo electrónico**.

El código SQL (el que deberemos introducir en la interfaz SQL del programa *phpMyAdmin*) sería el siguiente:

```
CREATE TABLE populares (
  ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  nombre TEXT,
  apellido TEXT,
  mote TEXT,
  votos INT
);
```

Este código creará una nueva tabla, llamada **populares**, con cinco campos: el identificador del profesor (**ID**), su **nombre**, su primer apellido, su **mote** (¡el nombre de un profesor puede que se olvide, pero su mote nunca!), y otra llamada **votos**, donde contabilizaremos todos los votos recibidos.



**Figura 5.5.** Ejemplo de votacion dinamica.

Para hacer mas sencilla la votacion, Ricardo ha decidido restringir los candidatos a cuatro.

Para insertar a dichos profesores en la tabla, debaremos introducir el siguiente codigo en la interfaz SQL de la aplicacion *phpMyAdmin*.

```
INSERT INTO populares values
(1, 'Pablo', 'Partido', 'El Leches', 0),
(2, 'Avelino', 'Puebla', 'El Fabas', 0),
(3, 'César', 'Artero', 'Hermano Artero', 0);
```

Veamos ahora el *script* en PHP:

```
<!-- queridos.php -->
<html>
<head>
<title>Votación profesores</title>
</head>
<form action="<?=$PHP_SELF?>" method=post>
<body>
    <?php
        if ($bueno == "a")
        {
            $con=mysql_connect("localhost", "ricardo",
            "maravillas");
```

```
mysql_select_db ("alumnos");
$sql= "UPDATE populares SET votos=votos+1
WHERE ID=1";
if (!mysql_query($sql)) {
echo ("error en la votación. Intentelo de
nuevo") ;
}
if ($bueno == "b")
{
$con=mysql_connect("localhost", "ricardo",
"maravillas");
mysql_select_db("alumnos");
$sql= "UPDATE populares SET votos=votos+1
WHERE ID=2";
if (!mysql_query($sql)) {
echo ("error en la votación. Intentelo de
nuevo") ;
}
if ($bueno == "c")
{
$con=mysql_connect("localhost", "ricardo",
"maravillas");
mysql_select_db("alumnos");
$sql="UPDATE populares SET votos=votos+1
WHERE ID=3";
if (!mysql_query($sql)) {
echo ("error en la votación. Intentelo de
nuevo") ;
}
?>
<table>
    <tr>
        <td>¿Quién fue para ti el
profesor más querido del
Colegio Maravillas?</td>
        <td><select name="bueno">
<option value = "a">Pablo
Partido, "El Leches"
<option value = "b">Avelino
Puebla "El Fabas"
<option value = "c">Cesar Artero
        "Hermano Artero"
</select></td>
        <td colspan=2 align=center>
```

```

        <input type=submit value="Vota ">
    </td>
</tr>
<br><br><br><br><br>
</table>
</form>
<?php
$bd=@mysql_connect("localhost", "ricardo",
"maravillas");
if (!$bd) {
echo ("Error, No se pudo conectar con la
base de datos en este
momento, Intentelo mas tarde");
exit();
}
$sel=@mysql_select_db("alumnos");
if (!$sel)
{
echo( "Error, No se puede acceder a la base
de datos en este
momento, Intentelo más tarde");
exit();
}
$busqueda= @mysql_query("SELECT nombre,
apellido, mote,
votos FROM populares");
if (!$busqueda)
{
echo ("Error al seleccionar los elementos
de la base de datos,
Intentelo más tarde");
exit();
}
while ($row = mysql_fetch_array($busqueda))
{
echo("<p>" . $row["nombre"] . " ".
$row ["apellido"]." " .
$row["mote"]." ".$row["votos"]." votos"
- "c/p> ");
}
?>
</body>
</html>

```

## Control y proceso de la información en nuestra Web

Los *scripts* realizados en el capítulo anterior, aunque a efectos didácticos funcionan perfectamente, presentan una serie de deficiencias que, en el caso de que esperemos que nuestra página tenga un tráfico de visitantes considerable, puede crearnos algunos problemas.

La principal deficiencia es que la información se almacena en la base de datos tal y como ha sido escrita por el usuario, sin ningún tipo de control previo.

Esto puede crear problemas como los siguientes:

- Si un usuario introduce su apellido como *martínez*, sin la mayúscula delante, cuando realicemos una búsqueda del apellido *Martinez*, con mayúscula, en nuestro buscador, puede dar lugar a errores.
- El usuario puede dejar campos en blanco en el formulario, que se almacenarán tal cual en la base de datos.
- Si, por error, dejamos un espacio en blanco al principio del formulario, este se almacenará de manera incorrecta.
- El usuario puede grabar palabras malsonantes que aparecerán directamente en la Web sin que nadie pueda evitarlo.
- Etc.

Por ello, toda página Web dinámica a cuyos contenidos pueda acceder un número grande de personas, necesita establecer unos parámetros de control en el proceso de la información que va a aparecer. Para ello, existen una serie de sentencias en PHP que nos ayudarán en nuestra tarea.

## 6.1. Funciones de control de texto en PHP

### 6.1 - ■ Funcion trim(), ltrim() y chop()

La función `trim()` elimina los espacios sobrantes al principio y al final de una cadena de texto. Ello lo convierte en una función esencial para controlar los textos a almacenar en nuestra base de datos MySQL.

Veamos un ejemplo:

```
$apellido=" Perez ";
```

Nuestro usuario acaba de escribir su apellido con un espacio al principio, y otro al final. Ello hará casi imposible su localización por parte de nuestro buscador, aparte de consumir un espacio innecesario.

Pero si hacemos:

```
$apellido=trim($apellido);
```

Antes de guardarlo en la base de datos, éste se almacenará de manera correcta, sin ningún tipo de espacio ni delante ni detrás.

**Soluciones alternativas son las funciones ltrim() y chop().**

`ltrim()` elimina solo los espacios al principio de la cadena de texto (los de la izquierda), y `chop()` elimina solamente los espacios del final (los de la derecha).

### 6.1.2. Funcion nl2br()

Si tiene alguna experiencia en el diseño de páginas Web, habrá comprobado que para que un texto aparezca en la página de manera idéntica a como lo hemos escrito en cualquier procesador de texto, deberemos incluir las sentencias HTML que sirven para señalizar los saltos de línea, los acentos, las palabras en negrita, etc.

La función `nl2br()` toma la cadena de texto y reemplaza todos los saltos de línea por la sentencia HTML `<br>`. Ello permite que el texto se vea en la Web de la misma manera en que se ha escrito.

Veamos un ejemplo:

```
<?
$prueba="Bienvenido al
    Colegio Maravillas";
echo ($prueba);
?>
```

Este *script* mostrara en pantalla la frase: "Bienvenido al Colegio Maravillas", eliminando los saltos de linea.

Sin embargo, si incluimos la función nl2br() de la siguiente manera:

```
<?
$prueba="Bienvenido al
    Colegio Maravillas ";
echo nl2br ($prueba);
?>
```

Este *script* mostrara en pantalla la frase: "Bienvenido al Colegio Maravillas" con sus correspondientes saltos de linea, tal y como fueron escritos originalmente. Ello nos servira para conservar la integridad de las cadenas de texto originales.

### 6.1.3. Funcion htmlspecialchars()

Como hemos visto con la funcion anterior, pueden darse confusiones entre los textos escritos por nosotros, o por los usuarios de nuestra pagina, y las sentencias empleadas por el lenguaje HTML, que es el que entiende nuestro navegador. La funcion htmlspecialchars() nos sirve para desactivar el codigo HTML en los textos que escribamos.

Veamos un ejemplo:

```
<?
$prueba="Bienvenido al colegio Maravillas <el
genuino>";
echo ($prueba);
?>
```

Este *script* mostrara en pantalla "Bienvenido al colegio Maravillas", omitiendo las palabras "el genuino", que, al estar rodeadas por el signo "<", utilizado tambien en HTML, a nuestro navegador le provoca una confusion que le lleva a eliminar dicha parte de la cadena de texto.

Pero si modificamos nuestro *script* añadiendo la función **htmlspecialchars()** de la siguiente manera:

```
<?
$prueba="Bienvenido al colegio Maravillas <el
genuino>"  
    echo htmlspecialchars ($prueba);  
?>
```

Nos mostrara el texto tal y como ha sido escrito, es decir “Bienvenido al colegio Maravillas <el genuino>”.

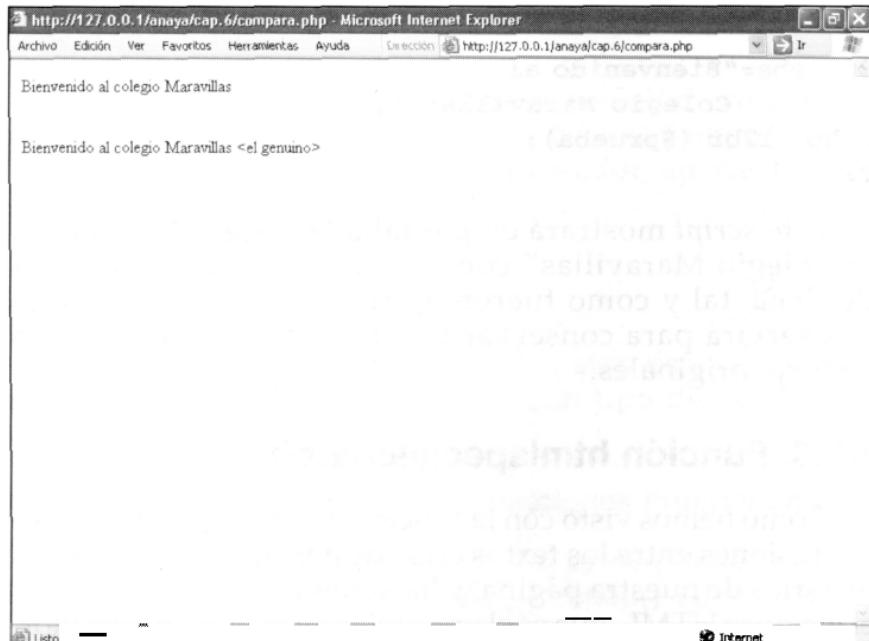


Figura 6.1. Comparativa de formato de texto.

**Nota:** Debemos advertir claramente si los usuarios pueden incluir sus textos en nuestra página Web con etiquetas(tags) de HTML. Ello nos ahorrara tener que revisarlos continuamente.

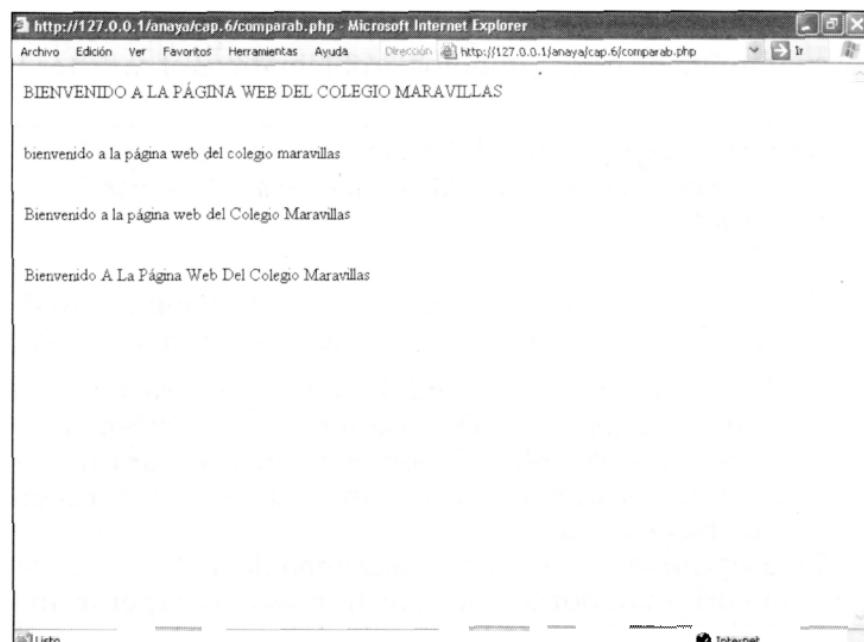
#### 6.1.4. Funcion **strtoupper()**, **strtolower()**, **ucfirst()** y **ucwords()**

Como ya hemos comentado, otro motivo de confusión en nuestra página Web dinámica es el tema de las mayúsculas

y las minusculas en las cadenas de texto. Dado que previsiblemente serán muchos los antiguos alumnos que introduzcan sus datos en nuestra página, será muy útil establecer unos parámetros de control automático del uso de mayúsculas que nos evite tener que revisar los datos manualmente.

Para ello, existen las siguientes funciones:

- **strtoupper()**: transforma toda la cadena de texto en letras mayúsculas.
- **strtolower()**: transforma toda la cadena de texto en letras minusculas.
- **ucfirst()**: transforma en mayúscula la primera letra de la cadena de texto.
- **ucwords()**: transforma en mayúscula la primera letra de cada palabra de la cadena de texto.



**Figura 6.2.** Diversas formas de mostrar el texto.

Veamos un ejemplo de cada uno de los casos:

```
$texto = "Bienvenido a la página web del Colegio  
Maravillas"  
strtoupper($texto) lo convertirá en "BIENVENIDO  
A LA PAGINA WEB DEL COLEGIO MARAVILLAS"
```

```
strtolower($texto) lo convertird en "bienvenido  
a la pdgina web del colegio maravillas"  
ucfirst($texto) lo convertird en "Bienvenido a la  
pdgina web del colegio maravillas"  
ucwords($texto) lo convertird en "Bienvenido A La  
Pdgina Web Del Colegio Maravillas"
```

## 6.1.5. Funcion AddSlashes() y StripSlashes()

A la hora de almacenar cadenas de texto en nuestra base de datos MySQL, hay algunos caracteres que causan problemas, pues son confundidos por caracteres de control del lenguaje SQL.; si no somos conscientes de ello, puede provocarnos mas de un quebradero de cabeza.

Estos caracteres problematicos son esencialmente: las comillas (simples y dobles) y el simbolo \.

Para evitar este problema, la solucin es agregar delante de cualquiera de estos simbolos problematicos el caracter \

---

**Nota:**Paradójicamente, el simbolo \ es un carácter problemático, pero a la vez es el que nos ayuda a solucionar los problemas.

---

Para evitarnos tener que realizar esta funcion manualmente, PHP incluye dos funciones que realizan este trabajo:

1. **AddSlashes()** se encarga de incluir el caracter \ en nuestra cadena de texto delante de cada simbolo problemático. Por ells, debemos utilizarla para filtrar cualquier cadena de texto antes de almacenarla en una base de datos.
2. **StripSlashes()** devuelve la cadena de texto a su estado original, por lo que la usaremos al recuperar una cadena de texto de la base de datos y antes de mostrarla en pantalla.

Veamos un ejemplo:

```
$texto = "Aquel profesor, conocido como "El  
Botijo", era un anciano por aquellos tiempos"  
$texto= AddSlashes($texto); , lo transformard en  
"Aquel profesor, conocido como \\"El Botijo\\", era  
un anciano por aquellos tiempos"
```

`$texto=StripSlashes($texto);` , lo devolvera a su estado original: "Aquel profesor, conocido como "El Botijo", era un anciano por aquellos tiempos".

## 6.2. Como buscar y reemplazar palabras y símbolos en las cadenas de texto

En una pagina Web en la que cualquiera puede incluir sus textos, es facil que entren personas malintencionadas que introduzcan datos falsos o palabras que pueden resultar ofensivas. Por ello, es necesario identificar todos aquellos datos que consideremos incorrectos, y establecer un mecanismo de control que los sustituya por otros.

### 6.2.1. Como identificar cadenas de texto

Para identificar unas cadenas de textos dentro de otras, existe una funcion dentro de PHP llamada `ereg()`, que hace esa funcion mucho mas sencilla.

El esquema de funcionamiento de `ereg()` es el siguiente:

```
ereg( "palabra-a-buscar", cadena de texto );
```

Veamoslo con un ejemplo:

```
<?
$texto = "Bienvenido al colegio Maravillas";
if (ereg("Maravillas", $texto)) {
echo( '$texto contiene la palabra "Maravillas"' );
}
else {
echo ( '$texto no contiene la palabra "Maravillas"' );
}
?>
```

Este *script* mostrara en pantalla la frase "\$texto contiene la palabra "Maravillas"" , puesto que, efectivamente, esta incluida dentro de dicha cadena de texto. Pruebe a sustituir dentro de la sentencia `ereg()` la palabra "Maravillas" por otra, y pruebe a ejecutar el *script*.

Una sentencia casi idéntica a la anterior vista es `ereg()`. La unica diferencia con `ereg()` es que esta ultima no distingue entre mayusculas y minusculas.

Veamoslo con un ejemplo. Partiendo de la cadena de texto siguiente:

```
$texto = "Bienvenido al colegio Maravillas";
```

La función `ereg` ("maravillas", \$texto) no encontraría ninguna concordancia, al estar maravillas en minúscula, mientras que `eregi` ("maravillas", \$texto) sí lo encontraría, al no importarle el tema de mayúsculas y minúsculas.

## 6.2.2. Sustituir una cadena de texto por otra

Una vez que hemos aprendido a encontrar palabras o cadenas de texto dentro de otras, el siguiente paso es aprender a sustituirlas de manera automática por otra. Para ello, se emplea la función `ereg_replace(palabra_a_reemplazar, palabra_nueva, cadena_de_texto);`

Veamoslo con un ejemplo:

```
<?
$texto = "Bienvenido al Colegio Maravillas";
$texto=eregi_replace("Colegio", "Instituto",
$texto);
echo $texto;
?>
```

Este *script* mostraría en pantalla la frase "Bienvenido al Instituto Maravillas". De manera alternativa, existe la función `eregi_replace()`, identica a `ereg_replace()`, excepto que no le influyen el tema de las mayúsculas y las minúsculas.

## 6.2.3. Clases de caracteres

La manera que hemos visto de buscar y reemplazar cadenas de texto puede resultar bastante tediosa, pues estamos identificando las palabras que queremos encontrar y sustituir una por una.

Para hacer más sencilla nuestra tarea, existen las clases de caracteres.

Veamoslo con un ejemplo:

- [a-z] significa "cualquier letra entre la a y la z, en minúsculas", y de esta manera nos evita tener que escribir todo el abecedario.
- [a-zA-Z] significa "cualquier letra entre la a y la z, ya sea en minúsculas o mayúsculas".

- [“a-z] significa ”cualquier caracter que NO esté entre la a y la z”.
- [aeiou] significa ”cualquier vocal”.
- ^[12345] significa ”cualquier numero del uno al cinco al principio de la cadena de texto”.
- [12345]\$ significa ”cualquier numero del uno al cinco al final de la cadena de texto”.

---

*Nota: El carácter ^ tiene una utilidad distinta si estd dentro o fuera de los paréntesis. Fuera del paréntesis significa que dicho conjunto de caracteres debe encontrarse al principio de la cadena de texto.*

---

## Aplicacion practica : validación del campo de correo electronico

En el capitulo anterior desarrollamos un *script* que permitia a los antiguos alumnos introducir sus datos en nuestra base de datos MySQL. Sin embargo, alumnos malintencionados pueden introducir cualquier texto que ellos quieran, aunque no tengan que ver con sus datos personales, o incluso dejarlo en blanco.

Veamos cómo solucionar estos temas con el campo de Correo Electronico. Sabemos que toda direccion de correo electronico sigue una estructura similar a esta: nombre@dominio.extension.

Tomando la cadena de texto \$email, que es donde el usuario ha incluido su dirección de correo electronico, aplicamos la siguiente linea de codigo:

```
if (!eregi("^[a-zA-Z0-9_]+@[a-zA-Z0-9\.-]+\.[a-zA-Z0-9\.-\.]+\.$", $email))  
{ echo "Su dirección de correo electrónico no es  
válida";  
exit;
```

Introduciendo esta sencilla linea de codigo, nos aseguramos que toda direccion de correo electronico incluida en nuestra base de datos tiene la estructura correcta.

# MySQL avanzado

En este capitulo estudiaremos los sistemas para mantener nuestra base de datos segura y a salvo de curiosos y visitantes malintencionados. Asimismo, avanzaremos en el estudio de las sentencias que nos permitiran crear bases de datos mas eficientes y utiles.

## 7.1. Por que necesitamos copias de seguridad

Desde el momento en que nos embarcamos en la realizacion de una pagina Web dinamica, tenemos que asimilar un nuevo concepto: "los contenidos lo son todo".

Por ello, debemos velar por la integridad de los contenidos de nuestra base de datos a toda costa, adoptando una actitud casi "paranoica" en su conservacion y cuidado. Si nuestros contenidos desaparecen, todo nuestro trabajo habra sido en vano. Los visitantes dejaran de acudir a nuestro sitio Web y pronto se correrá la voz, convirtiendo nuestro antes transitado *site* en un desierto cibernetico.

Por ello, debemos crearnos una disciplina de conservacion de los datos realizando copias de seguridad regularmente. Si hemos contratado una empresa de alojamiento profesional, estamos minimizando los riesgos de una perdida accidental de datos, pero aun asi el riesgo sigue existiendo. Debemos tener en cuenta que los servidores trabajan 24 horas al dia 7 dias a la semana, lo que incrementa las posibilidades de sufrir un fallo. Tambien cabe la posibilidad de que un usuario malintencionado (*cracker*) decida realizar sus "ejercicios" de sabotaje sobre nuestro servidor. Y no debemos olvidar que los edificios o las maquinas pueden asegurarse, pero ninguna compania asegurara los contenidos almacenados en nuestra base de datos.

Si nuestra pagina recibe un flujo de información considerable, o si es un *site* de comercio electronico, donde la perdida de un pedido puede causarnos grandes trastornos, debemos plantearnos realizar copias de seguridad diariamente, e incluso varias veces al dia. Sea como fuese, debemos obligarnos a realizar copias de seguridad como mínimo una vez a la semana.

**Nota:** *No hay que olvidar que, en caso de un fallo grave en el servidor, perderemos todos los datos que se introdujeron desde la última copia de seguridad. Hay que tener muy en cuenta este dato a la hora de planificar nuestro calendario de copias de seguridad.*

### 7.1.1. Como realizar copias de seguridad de nuestras bases de datos

Aunque el lenguaje SQL incorpora sentencias propias para la realización de nuestras copias de seguridad (`BACKUP`()), utilizaremos las posibilidades que nos ofrece la aplicación *phpMyAdmin* (ver capítulo 2) para realizar estas tareas.

Veamos como hacerlo paso por paso.

Entre en la aplicación *phpMyAdmin* y, en la parte superior derecha, seleccione la base de datos **Alumnos**.

Tabla	Acción	Campos	Tipo	Tam
alumnos	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	7	MyISAM	1
authors	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	2	MyISAM	2
categories	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
filestore	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	1,588	MyISAM	253
jokelookup	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
jokes	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
odiaodos	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	3	MyISAM	2
odiaodos2	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	0	MyISAM	1
populares	Examinar Seleccionar Insertar Propiedades Eliminar Vaciar	3	MyISAM	2
9 tabla(s)		1,603	--	266

Figura 7.1. Aparecerá la lista con todas las tablas almacenadas en dicha base de datos.

En el menu superior, pulse en Exportar, y se le mostrara una nueva pantalla donde estaran indicadas todas las tablas incluidas en la base de datos Alumnos. Haga clic en aquellas que desee hacer la copia de seguridad (para nuestro ejemplo, seleccione todas). Asimismo, de las diversas posibilidades que tiene, marque las llamadas Estructura y datos, Usar Backquotes, Tablas y Nombres de campo, y Enviar. El campo Enviar tiene a su vez dos opciones: Comprimido con zip, para el caso de que sean tablas muy extensas y su sistema operativo sea *Windows*, o Comprimido con gzip, en el caso de que trabaje con sistema operativo *Linux*. En nuestro caso, al ser tablas de datos pequeñas, no hay necesidad de compresion.

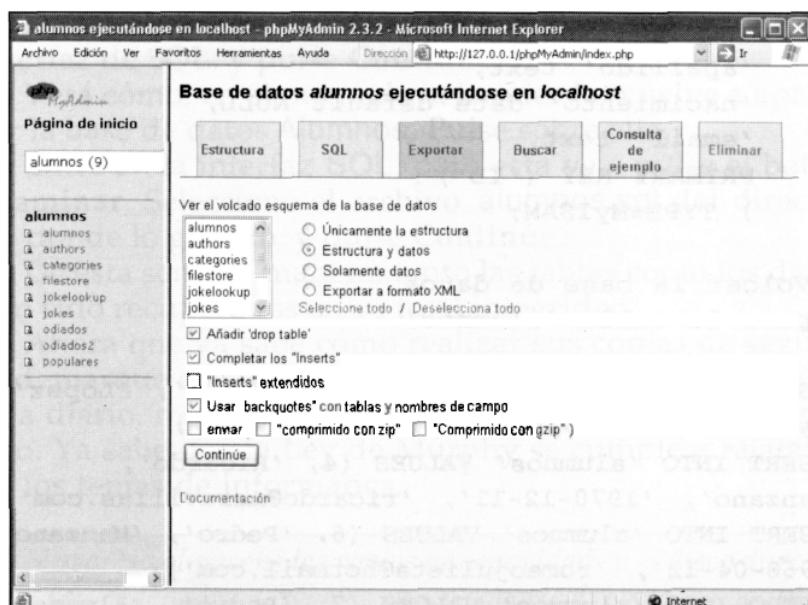


Figura 7.2. Pulse Continuar.

Le indicara que se dispone a descargar el archivo **alumnos.sql**, y le da la opción de **Abrir** o **Guardar**. Haga clic en **Guardar**. A continuación, seleccione el directorio donde almacenar este archivo, y recuerde bien de qué directorio se trata.

De esta manera tan sencilla, hemos realizado nuestra primera copia de seguridad.

Abra dicho archivo y examinelo:

```
# phpMyAdmin MySQL-Dump
# version 2.3.2
```

```
# http://www.ghpmyadmin.net/ (download page)
#
# servidor: localhost
# Tiempo de generación: 17-11-2002
# a las 20:42:24
# Versión del servidor: 3.23.52
# Versión de PHP: 4.4.0-dev
# Base de datos : 'alumnos'
#
# _____
#
# Estructura de tabla para la tabla 'alumnos'
#
CREATE TABLE 'alumnos' (
    'ID' int(11) NOT NULL auto-increment,
    'nombre' text,
    'apellido' text,
    'nacimiento' date default NULL,
    'email' text,
    PRIMARY KEY ('ID')
) TYPE=MyISAM;
#
# Volcar la base de datos
# para la tabla 'alumnos'
#
INSERT INTO 'alumnos' VALUES (5, 'Luis', 'Lopez',
'Ovejero', 'romeojulieta@hotmail.com');
INSERT INTO 'alumnos' VALUES (4, 'Ricardo',
'Manzano', '1970-12-11', 'ricardo@maravillas.com');
INSERT INTO 'alumnos' VALUES (6, 'Pedro', 'Manzano',
'1968-04-12', 'romeojulieta@hotmail.com');
INSERT INTO 'alumnos' VALUES (7, 'Pedro', 'Almansa',
'1968-04-12', 'romeojulieta@hotmail.com');
INSERT INTO 'alumnos' VALUES (8, 'Jose', 'Gallego',
'1974-01-01', 'jose.gallego@idoweb.net');
```

Todo en este documento debe resultarle familiar, pues se trata de sentencias en lenguaje SQL que ya hemos visto a lo largo de esta guía. Lo que hay en el documento es la estructura para reconstruir las diversas tablas, así como los datos que había en ellas. Así, en caso de cualquier pérdida o accidente, podremos reconstruirlas de manera idéntica a como eran.

Ahora simularemos una pérdida accidental de nuestros datos y los pasos necesarios para recuperarlos.

Volvamos a la pagina inicial de *phpMyAdmin*.

Seleccione la base de datos **Alumnos**, y pulse el boton **Eliminar**.

¡Todo nuestro trabajo echado a perder! Por mas que busque, no encontrara rastro alguno de la base de datos **Alumnos**, ni ninguna de las tablas que contenia, y que tanto nos costó crear. Conserve la sangre fria, pues esta es una situación que desgraciadamente se le puede presentar en algun momento, pero que, si se han tornado las medidas oportunas, no habrá demasiado problema.

Seleccione cualquiera de las bases de datos que quedan, y pulse la opción **SQL** del menu superior.

Escriba la frase **CREATE DATABASE alumnos;** en la interfaz de **SQL**, y pulse **Continue**.

Vera como, en el menu de la izquierda, vuelve a aparecer la base de datos **Alumnos**. Pulse sobre ella.

Entre en la interfaz **SQL**, pero esta vez, pulse el boton **Examinar**. Seleccione el archivo **alumnos.sql** del directorio donde lo guardo, y pulse **Continue**.

De esta sencilla manera, tanto las tablas como los datos han sido recuperados en su total integridad.

Ahora que ya sabe como realizar sus copias de seguridad, marque en su calendario los dias que las realizara (si es a diario, mejor), y bajo ningun concepto se lo pase por alto. Ya sabe que la Ley de Murphy se cumple a rajatabla en los temas de informática.

---

***Nota:** No almacene las copias de seguridad en el disco duro de su ordenador. Grabelos en un soporte externo, como un CD o un ~~zip~~, indicando siempre la fecha en que las realizó.*

*De esa manera, le sera mucho más fácil arreglar cualquier desastre. Y si en dichas copias hay almacenados números de tarjetas de crédito, contraseñas, o algun otro tipo de información confidencial, no olvide guardarlas en lugar seguro.*

---

## 7.2. Privilegios para acceder a nuestra base de datos

En un momento dado, podemos plantearnos la necesidad de que alguien nos ayude a administrar nuestra página y, por consiguiente, las bases de datos MySQL. Sin

embargo, no queremos que estas personas tengan el mismo control que nosotros (que tenemos control absoluto). Quizá solo queremos que puedan acceder a determinadas tablas, o realizar determinadas funciones. Esto se conoce como "Sistema de Privilegios".

Por privilegio se entiende el derecho a realizar una determinada acción en un determinado lugar, y está asociado al concepto de usuario. Cada usuario diferente debe tener su propio privilegio.

**Nota:** A la hora de conceder los permisos de acceso a nuestros futuros ayudantes, debemos regirnos por el "principio del menor privilegio", es decir, darles el mínimo de autorizaciones necesarias para que puedan realizar las tareas encomendadas.

### 7.2.1. Tipos de privilegios en MySQL

Basicamente, son tres los tipos de privilegios que podemos conceder: privilegios para usuarios, privilegios para administradores y privilegios especiales.

Antes de conceder cualquier tipo de privilegio a nadie, defina a qué bases de datos y tablas debe tener acceso esa persona. Si usted almacena en una tabla números de tarjetas de crédito, o algún otro tipo de información confidencial, no debe permitir bajo ningún concepto que alguien ajeno a usted tenga acceso a ella.

#### Privilegios para usuarios

- **SELECT:** permite al usuario seleccionar los registros de las tablas.
- **INSERT:** permite al usuario insertar nuevos registros en las tablas.
- **UPDATE:** permite al usuario actualizar los valores ya existentes en las tablas.
- **DELETE:** permite al usuario borrar registros de las tablas.
- **ALTER:** permite al usuario alterar la estructura de las tablas, como por ejemplo, añadiendo columnas, cambiando su nombre, etc.
- **CREATE:** permite al usuario crear nuevas bases de datos o tablas.

- **DROP**: permite al usuario eliminar bases de datos o tablas.

## Privilegios para administradores

- **FILE**: permite al administrador leer y grabar archivos en el servidor MySQL.
- **PROCESS**: permite al administrador controlar los procesos que sucedan en el servidor, e incluso detenerlos.
- **RELOAD**: permite al administrador reformar los accesos, privilegios, tablas, logs, etc.
- **SHUTDOWN**: permite al administrador apagar el servidor MySQL.

## Privilegios especiales

- **ALL**: concede todos los privilegios descritos anteriormente.
- **USAGE**: permite al acceso, pero nada mas.

### 7.2.2. Conceder y quitar privilegios: funciones GRANT y REVOKE

La función **GRANT** es la que utilizaremos para conceder todos los privilegios que acabamos de ver.

Su funcionamiento es el siguiente:

```
GRANT privilegios
ON base/tabla
TO usuario [IDENTIFIED BY 'contraseña']
[WITH GRANT OPTION];
```

Veamoslo mas en detalle, analizando línea por línea de esta estructura:

- **GRANT privilegios**: A continuación de **GRANT**, incluiremos todos los privilegios que queramos conceder a este usuario de entre los vistos anteriormente, separados por comas.
- **ON base/tabla**: A continuación de **ON** escribimos la base de datos o tabla sobre la que tendrá vigor el privilegio. Si queremos extender este privilegio a todas las bases de datos presentes en el servidor, lo indicaremos con **\*.\***.

- **TO usuario [IDENTIFIED BY 'contraseiiia']:** Usuario indica el nombre que se deberá introducir para poder acceder al servidor MySQL, y contraseña, evidentemente, la contraseña que necesitará para acceder al servidor MySQL.
- **WITH GRANT OPTION:** Se usa para permitir al usuario la facultad de conceder a otros usuarios sus mismos privilegios. Como regla general, es aconsejable no utilizarla nunca, pues nos podemos encontrar con una serie de usuarios con privilegios de los que nunca hemos oido hablar.

La función REVOKE se utiliza para retirar privilegios a los usuarios. Su sintaxis es muy similar a la de GRANT:

```
REVOKE privilegios
ON base/tabla
FROM usuario
```

El funcionamiento de cada una de las líneas es idéntico al de GRANT.

### 7.2.3. Conceder y quitar privilegios. Ejemplo práctico

Supongamos que Ricardo, abrumado por la gran cantidad de trabajo que le supone su nueva página Web dinámica dedicada al colegio Maravillas, decide conceder privilegios de administrador a su amigo Alfredo, antiguo compañero de clase, que ha mostrado gran interés por el proyecto.

El código a escribir en la interfaz SQL de la aplicación *phpMyAdmin* sería el siguiente:

```
GRANT ALL
on *
to Alfredo IDENTIFIED by 'romeo'
WITH GRANT OPTION;
```

Pero al cabo de unos días descubrimos que Alfredo está creando desmanes entre nuestras tablas y, para colmo, está utilizando los privilegios concedidos por WITH GRANT OPTION para dar privilegios a nuevos usuarios, así que Ricardo, ni corto ni perezoso, opta por quitarle todos los permisos.

```
REVOKE ALL
on *
from Alfredo;
```

## 7.3. Trabajar con varias tablas de datos

Hasta ahora, en nuestro ejemplo de pagina Web dinámica del colegio Maravillas solo hemos trabajado con una base de datos. Sin embargo, a medida que vayamos introduciendo aplicaciones mas complejas, sera imprescindible trabajar con diversas bases de datos. A continuación, aprenderemos los conceptos basicos del manejo de varias bases de datos y las sentencias empleadas para ello.

Retomemos el ejemplo de la biblioteca de la que hablamos en el capítulo 4. Hemos aprendido que una base de datos es la manera ideal de guardar todos nuestros libros para poder tenerlos siempre ordenados y recuperarlos en base al criterio deseado, ya sea por autor, tematica, editorial, etc.

### 7.3.1. Por que utilizar diferentes bases de datos

Uno podria preguntarse: ¿por qué molestarse en crear varias bases de datos cuando toda la informacion puede almacenarse en una? Se podria crear una gran base de datos donde, junto con el título, autor, editorial, incluyésemos la temática del libro, el idioma, etc. Esto podria parecer más sencillo a simple vista, pero estariamos creando una redundancia de datos. Con una tabla unica, tendriamos que repetir los datos del autor en cada uno de los libros que de ese escritor tuviksemos en nuestra biblioteca.

Veamos el ejemplo:

ID	Título	Autor	Editorial	Idioma
1	<i>Cañas y Barro</i>	Blasco Ibáñez, Vicente	Biblioteca Nueva	Castellano
2	<i>La Bodega</i>	Blasco Ibáñez, Vicente	Catedra	Castellano
3	<i>El Papa del Mar</i>	Blasco Ibáñez, Vicente	Anaya	Castellano
4	<i>La Catedral</i>	Blasco Ibáñez, Vicente	Biblioteca Nueva	Castellano
...				

Tabla 7.1. Algunos libros de nuestra biblioteca.

A simple vista, estamos viendo los inconvenientes de guardar todos los datos en la misma tabla. ¿Acaso hay cuatro escritores diferentes con el mismo nombre? ¿O son cuatro libros distintos escritos por la misma persona? ¿Por qué introducir el mismo autor varias veces? ¿Por que escribir la palabra Castellano tantas veces?

Debemos tener en cuenta que cuantos más datos introduzcamos en la tabla, tanto mas lento sera el proceso de busqueda. Asimismo, no es nada dificil cometer algun error a la hora de introducir los datos, y sustituir, por ejemplo, *Blasco* por *Basco*. Ello hara imposible identificar ese libro, y nos habrá creado un escritor fantasma. Por eso debemos evitar la redundancia de datos. Cuanto mas simples sean nuestras bases de datos, y menos veces tengamos que repetir datos, tanto mas seguras, rápidas y efectivas seran.

Veamos de qué manera podriamos dividir la base de datos de nuestro ejemplo en multiples tablas, y cómo se relacionan entre si.

Tabla Autores:

<i>ID</i>	<i>Autor</i>
1	Blasco Ibáñez, Vicente
2	Miller, Henry
3	Saramago, José
4	Gallego Vázquez, Jose Antonio
...	

Tabla Idioma:

<i>LID</i>	<i>Idioma</i>
1	Castellano
2	Catalán
3	Gallego
4	Euskera
5	Inglés

Tabla Editorial:

<i>EID</i>	<i>Nombre</i>
1	Anaya Multimedia
2	Biblioteca Nueva
3	Catedra
4	Anaya
...	

Tabla Libros:

<i>TID</i>	<i>Título</i>	<i>ID</i>	<i>LID</i>	<i>EID</i>
1	<i>Todos los nombres</i>	3	1	4
2	<i>Tropic of Cancer</i>	2	5	3
3	<i>Cañas y Barro</i>	1	1	2
4	<i>La Bodega</i>	1	1	2
...				

¿Ve que eficiente manera de relacionar nuestras distintas bases de datos? Para ello, hemos empleado el número identificador del que hablábamos en el capítulo 4: un numero único que identifica cada uno de los campos. De esta manera evitamos la redundancia de los datos, permite una búsqueda mucho más rápida, y las modificaciones son más sencillas.

## 7.4. Consultar diversas bases de datos

Veamos mediante ejemplos cómo realizar una búsqueda en nuestra nueva base de datos de biblioteca. Imaginemos que nos interesa saber cuantos libros de nuestra biblioteca están escritos en euskera.

La sentencia SQL que deberíamos emplear sería la siguiente:

```
SELECT autor FROM autores
WHERE LID= 3;
```

O, si acaso necesitamos saber todos los libros de Henry Miller que hay en nuestra biblioteca editados por Cátedra, escribiríamos la siguiente sentencia:

```
SELECT autor FROM autores  
WHERE ID=2 and EID=4;
```

Con todo lo que hemos aprendido hasta ahora, probemos a realizar una aplicación Web que nos permita:

1. Dar de alta todos los libros en nuestra biblioteca, introduciéndolos siguientes datos: título, autor, editorial, temática e idioma.
2. Consultar nuestros libros en base a cualquiera de esos criterios.
3. Ser accesibles vía Internet para cualquiera que desee consultarlos.

Crearemos la base de datos **biblioteca**, y en ella las siguientes tablas: autores, idioma, editorial, temática e idioma.

Creamos una nueva base de datos en nuestro servidor, llamada biblioteca. Introduzcamos la siguiente sentencia en la interfaz SQL de la aplicación *phpMyAdmin*.

```
create database biblioteca
```

Dentro de nuestra nueva base de datos, creamos cuatro nuevas tablas: autores, idioma, editorial y libros.

He aquí las sentencias que debemos introducir en la interfaz SQL:

#### **Tabla Autores:**

```
create table autores (  
ID int not null auto-increment primary key,  
AUTOR text);
```

#### **Tabla Idioma:**

```
create table idioma(  
LID int not null auto-increment primary key,  
IDIOMA text);
```

#### **Tabla Editorial:**

```
create table editorial(  
EID int not null auto-increment primary key,  
NOMBRE text);
```

## Tabla Libros:

```
create table libros(
TID int not null auto-increment primary key,
ID int not null,
LID int not null,
EID int not null,
TITULO text);
```

Creamos una pagina de entrada a nuestra biblioteca, en HTML, con un menu desde donde acceder a las diversas opciones que incluiremos en nuestra biblioteca.

**Nota:** Al ser una página en HTML, deberemos grabarla con la extensión .html, a diferencia del resto de scripts de nuestro ejercicio, que grabaremos con la extensión .php.

```
<!-- admin.html -->
<html>
<head>
<title>Gestión de una biblioteca</title>
</head>
<body>
    <h1>Gestión de biblioteca</h1>
    <ul>
        <li><a href="autores.php">Mantenimiento autores</a></li>
        <li><a href="idioma.php">Mantenimiento idiomas</a></li>
        <li><a href="editorial.php">Mantenimiento editoriales</a></li>
        <li><a href="libros.php">Lista de libros</a></li>
    </ul>
</body>
</html>
```

El primero de nuestros *scripts*, autores.php, establece conexión con la base de datos MySQL biblioteca, y realiza una búsqueda en la tabla autores, ordenando los resultados por orden alfabético.

```
($result = @mysql_query ("SELECT ID, AUTOR FROM autores ORDER BY autor"));
```



**Figura 7.3.** Pantalla de administración de la biblioteca.

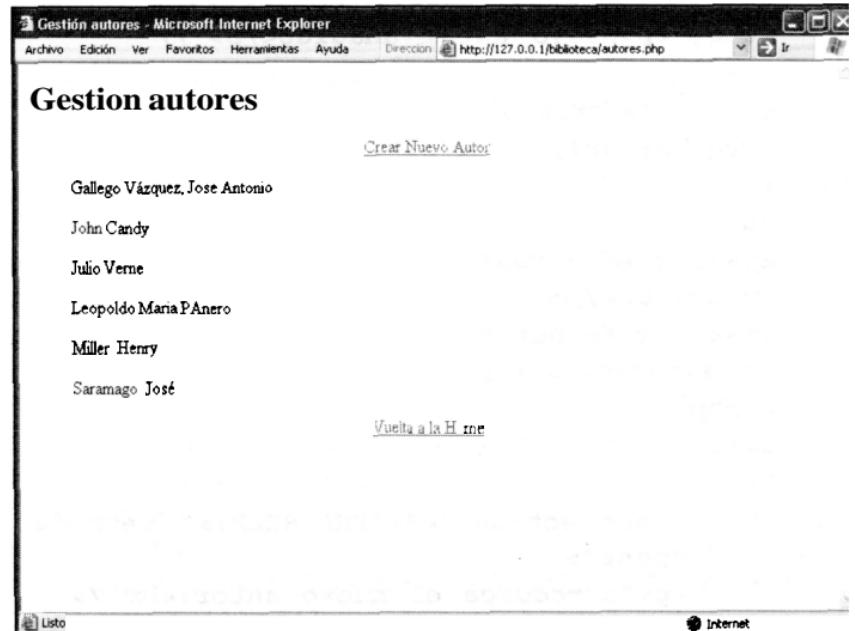
Y mostrandolos en pantalla. Asimismo, incluimos dos vínculos. Uno a la pagina de inicio, y otro al *script* nuevoautor.php, que es el que nos permitirá dar de alta a los escritores en nuestra base de datos.

```
<!-- autores.php -->
<html>
<head>
<title> Gestión autores </title>
</head>
<body>
  <h1>Gestión autores</h1>
  <p align="center">
    <a href="nuevoautor.php">Crear Nuevo Autor</a>
  </p>
  <ul>
    <?php
      $cnx = mysql_connect('localhost','ricardo',
      'maravillas');
      mysql_select_db('biblioteca');
      $result = @mysql_query ("SELECT ID, AUTOR FROM
      autores ORDER BY
      autor");
      if (! $result) {
```

```

echo("<p>Error al encontrar autores en la base
de dato e. !
<br />".
"Error: " . mysql_error() . "</p>"); .
exit();
}
while ($row = mysql_fetch_array($result)) {
echo("<p>" . $row["AUTOR"] . " " . "</p>"); .
}
?>
</ul>
<p align="center"><a href="admin.html">Vuelta
a la Página de
Inicio</a></p>
</body>
</html>

```



**Figura 7.4. Pantalla de muestra de autores.**

He aqui el *script* que utilizaremos para introducir los nombres de los escritores en la tabla autores. Consta de un sencillo formulario HTML que recoge los datos y los manda via POST a la zona PHP del *script*, que conecta con la tabla e inserta los datos en el campo **Autor**.

```
( $sql = "INSERT INTO autores SET AUTOR='$name'"; )
```

Incluye un vínculo para volver a la lista de autores y otro para dar de alta un nuevo autor.

```
c !- nuevoautor.php ->
<html>
<head>
<title> Añadir un nuevo autor </title>
</head>
<body>
<?php
if ($submit):
$dbcnx = mysql_connect('localhost',
'ricardo', 'maravillas');
mysql_select_db('biblioteca');
$sql = "INSERT INTO autores SET
AUTOR='".$name."'";
if (@mysql_query($sql)) {
echo("<p>Nuevo autor añadido</p>") ;
} else {
echo("<p>Error al añadir el nuevo autor: " .
mysql_error() . "</p>") ;
}
?>
<p><a href="<?=$PHP_SELF?>">Añadir un nuevo
autor</a></p>
<p><a href="autores.php">Volver a la lista
de autores</a></p>
<?php
else:
?>
      cform action="c?=$PHP_SELF?" method=
      "post">
      <p>Introduzca el nuevo autor:<br />
AUTOR: <input type="text" name=
"name" size="20"
maxlength="255" /><br />
<input type="submit" name=
"submit" value="SUBMIT" /></p>
      c/form>
<?php endif; ?>
</body>
c/html>
```



Figura 7.5. Pantalla de alta de autores.

```
<!-- editorial.php -->
<html>
<head>
<title> Gestión Editoriales </title>
</head>
<body>
    <h1>Gestión Editorial</h1>
    <p align="center"><a href=
    "nuevaeditorial.php">Crear nueva
    editorial</a></p>
    <ul>
        <?php
            $cnx = mysql_connect('localhost', 'ricardo',
            'maravillas');
            mysql_select_db('biblioteca');
            $result = @mysql_query("SELECT CID, NOMBRE
            FROM editorial
            ORDER BY NOMBRE");
            if (!$result) {
                echo("<p>Error al encontrar editoriales en
                la base de datos. !
                <br />".
                "Error: " . mysql_error() . "</p>");
```

```

    exit();
}
while ($row = mysql_fetch_array($result)) {
echo("<p>" . $row["NOMBRE"] . " " . "</p>");
}
?>
</ul>
<p align="center"><a href=
"admin.html">Volver a la Página de
Inicio</a></p>
</body>
</html>

```



Figura 7.6. Pantalla de búsqueda de editoriales.

```

<!-- nuevaeditorial.php -->
<html>
<head>
<title>Añadir nueva editorial</title>
</head>
<body>
    <?php
    if ($submit) :
    $dbcnx = @mysql_connect("localhost",
    "ricardo", "maravillas");

```

```

mysql-select-db("biblioteca");
$sql = "INSERT INTO editorial SET " .
"OMBRE=' $name' ";
if (@mysql_query($sql)) {
echo("<p>Nueva editorial añadida</p>") ;
} else {
echo("<p>error al añadir la editorial: " .
mysql_error() . "</p>") ;
}
?>
<p><a href=<?= $PHP_SELF?>>Añadir nueva
editorial</a></p>
<p><a href="editorial.php">Volver a la lista
de editoriales</a></p>
<?php
else:
?>
<form action=<?= $PHP_SELF?> method="post">
<p>Introduzca la nueva editorial:<br />
Name: <input type="text" name="name" size="20"
maxlength="255" /><br />
<input type="submit" name="submit" value=
"SUBMIT" /></p>
</form>
<?php endif; ?>
</body>
</html>

```

He aquí el *script* que ejecuta la búsqueda de libros en la base de datos.

```

<!-- libros.php -->
<html>
<head>
<title>Gestión de Libros</title>
</head>
<body>
<h1>Gestión de Libros</h1>
<p><a href="nuevolibro.php">Introducir un
nuevo libro</a></p>
<?php
$dbcnx = mysql_connect("localhost",
"ricardo", "maravillas");
mysql-select-db("biblioteca");

```

```
$authors = mysql_query("SELECT ID, AUTOR
FROM autores");
$cats = mysql_query ("SELECT CID, NOMBRE
FROM editorial");
$idioms = mysql_query("SELECT LID, IDIOMA
FROM idioma");
?>
<form action="listalibros.php" method="post">
<p>Ordenar los libros según el siguiente
criterio:<br />
Por autor:
<select name="aid" size="1">
<option selected value="">Todos los
autores</option>
<?php
while ($author = mysql_fetch_array($authors)) {
$aid = $author["ID"];
$aname = htmlspecialchars($author["AUTOR"]);
echo("<option value='".$aid."'>$aname</
option>\n");
}
?>
</select><br />
Por editorial:
<select name="cid" size="1">
<option selected value="">Todas las
editoriales</option>
<?php
while ($cat = mysql_fetch_array($cats)) {
$cid = $cat["CID"];
$cname = htmlspecialchars ($cat["NOMBRE"]);
echo("<option value='".$cid."'>$cname</
option>\n");
}
?>
</select><br />
Por idioma:
<select name="lid" size="1">
<option selected value="">Todos los idiomas</
option>
<?php
while ($idiom = mysql_fetch_array($idioms)) {
$lid = $idiom["LID"];
$lname = htmlspecialchars($idiom["IDIOMA"]);
```

```

echo("<option value='$lid'>$lname</option>\n");
}
?>
</select><br />
Que incluya el texto: <input type="text" name=
"searchtext" />
<br />
<input type="submit" name="submit" value=
"Search" />
</form>
<p align="center"><a href=
"admin.html">Volver a la
Página de Inicio</a></p>
</body>
</html>
<!-- listalibros.php -->
<html>
<head>
<title>Lista de libros</title>
</head>
<body>
    <h1>Listado de libros</h1>
    <p><a href="libros.php">Nueva búsqueda</
a></p>
    <?php
    $dbcnx = mysql_connect("localhost", "ricardo",
    "maravillas");
    mysql_select_db("biblioteca");
    $select = "SELECT DISTINCT libros.ID,
    libros.TID, libros.LID,
    libros.EID, TITULO, autores.ID, AUTOR,
    editorial.CID, NOMBRE,
    idioma.LID, IDIOMA";
    $from = " FROM libros, autores, editorial,
    idioma";
    $where = " WHERE libros.TID > 0 AND
    autores.ID=libros.ID AND
    editorial.CID=libros.EID AND idioma.LID=
    libros.LID";
    if ($aid != "") {
        $where .= " AND libros.ID=$aid";
    }
    if ($cid != "") { // A category is selected
        $where .= " AND CID=$cid";
    }
    $result = mysql_query($select . $from . $where);
    if ($result) {
        $row = mysql_fetch_array($result);
        $libros = $row[0];
        $TID = $row[1];
        $LID = $row[2];
        $EID = $row[3];
        $TITULO = $row[4];
        $ID = $row[5];
        $AUTOR = $row[6];
        $CID = $row[7];
        $NOMBRE = $row[8];
        $IDIOMA = $row[9];
    }
    mysql_free_result($result);
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Título</th>
            <th>Editorial</th>
            <th>Autor</th>
            <th>Categoría</th>
            <th>Idioma</th>
        </tr>
        <tr>
            <td>$libros</td>
            <td>$TITULO</td>
            <td>$AUTOR</td>
            <td>$NOMBRE</td>
            <td>$CID</td>
            <td>$IDIOMA</td>
        </tr>
    </table>
</body>
</html>

```

```

}

if ($lid != "") {
    $where .= " AND idioma.LID=$lid";
}

if ($searchtext != "") {
    $where .=
    " AND TITULO LIKE '%$searchtext%'";
}

?>


| Título | Autor    | Idioma | Editorial |
|--------|----------|--------|-----------|
| \$tit  | \$nombre | \$idio | \$edi     |


```

A continuación, el script para añadir un nuevo idioma.

```

<!-- nuevoidioma.php -->
<html>
<head>

```

```

<title>Añadir nuevo idioma</title>
</head>
<body>
<?php
if ($submit) :
$dbcnx = @mysql_connect("localhost",
"ricardo", "maravillas");
mysql_select_db("biblioteca");
$sql = "INSERT INTO idiom SET " .
"IDIOMA='$name'";
if (@mysql_query($sql)) {
echo("<p>Nuevo idioma añadido.</p>");
} else {
echo("<p>error " .
mysql_error() . "</p>");}
?>
<p><a href=<?=$PHP_SELF?>>Añadir nuevo
idioma</a></p>
<p><a href="idioma.php">Volver a la lista
de idiomas</a></p>
<?php
else :
?>
<form action=<?=$PHP_SELF?>" method="post">
<p>Introduzca el nuevo idioma:<br />
Name: <input type="text" name=
"name" size="20"
maxlength="255" /><br />
<input type="submit" name="submit" value=
"SUBMIT" /></p>
</form>
<?php endif; ?>
</body>
</html>

```

Para añadir un nuevo libro.

```

<!-- nuevolibro.php -->
<html>
<head>
<title>Añadir nuevo libro</title>
</head>
<body>

```

```
<?php
if ($submit): // Si se introduce un nuevo
libro
if ($aid == "") {
echo("<p>debes elegir un autor " .
"para este libro. " .
"Inténtalo otra vez.</p>") ;
exit();
}
$dbcnx = mysql_connect("localhost",
"ricardo", "maravillas");
mysql_select_db("biblioteca");
$sql = "INSERT INTO libros SET
LID='$lid',
TITULO='$nombre',
ID='$aid',
EID='$cid'";
if (@mysql_query($sql)) {
echo("<p>Ha añadido un nuevo libro</p>") ;
} else {
echo("<p>Error : " .
mysql_error() . "</p>") ;
}
?>
<p><a href=<?=$PHP_SELF?>">Añadir otro
libro</a></p>
<p><a href="admin.html">Volver a la Página
de Inicio</a></p>
<?php
else:
$dbcnx = @mysql_connect("localhost",
"ricardo", "maravillas");
mysql_select_db("biblioteca");
$authors = mysql_query ("SELECT ID, AUTOR
FROM autores");
$cats = mysql_query ("SELECT CID, NOMBRE
FROM editorial");
$idioms = mysql_query ("SELECT LID, IDIOMA
FROM idioma");
?>
cform action=<?=$PHP_SELF?>" method=
"post">
<p>Introduzca el nuevo libro:<br />
<p>nombre :cbr />
```

```

<textarea name="nombre" rows="15" cols="45" wrap>
</textarea></p><br />
<p>Autor:
<select name="aid" size="1">
<option selected value="">Selecciona Uno</option>
<option value="">-----</option>
<?php
while ($author =
mysql_fetch_array($authors)) {
$aid = $author ["ID"];
$aname =
htmlspecialchars($author ["AUTOR"]);
echo ("<option value=
'$aid'>$aname</option>\n");
}
?>
</select>
<br><br>
<p>Idioma:
<select name="lid" size="1">
<option selected value="">Selecciona
Uno</option>
<option value="">-----</option>
<?php
while ($idiom = mysql_fetch_array($idioms)) {
$lid = $idiom ["LID"];
$bname =
htmlspecialchars($idiom ["IDIOMA"]);
echo ("<option value=
'$lid'>$bname</option>\n");
}
?>
</select>
<p>Editoriales:
<select name="cid" size="1">
<option selected value="">Selecciona Uno</option>
<option value="">-----</option>
<?php
while ($cat = mysql_fetch_array($cats)) {
$cid = $cat ["CID"];
$cname = htmlspecialchars($cat ["NOMBRE"]);

```

```
echo("<option value='$cid'>$cname</
option>\n");
}
?>
</select>
<p><input type="submit" name="submit" value=
"SUBMIT"></p>
</form>
<?php endif; ?>
</body>
</html>
```

Sentencias avanzadas para el desarrollo de páginas Web dinamicas.

## 8.1. Por que reutilizar el codigo

Escribir codigo dista mucho de ser una actividad artística, donde la originalidad de una creación es un factor a tener en cuenta. A la hora de programar, es una magnifica idea reutilizar siempre que nos sea posible codigo ya escrito, pues de esta manera nos estamos asegurando su funcionamiento.

Todo elemento nuevo de *software* que escribamos requerira un periodo de prueba en el que indefectiblemente encontraremos fallos y elementos a mejorar. Ello nos lleva un tiempo, en muchos casos, mayor incluso del de escritura del codigo. Y ya conoce el refrán que dice “tiempo es dinero”.

Por ello, es una buena idea incluir en lo posible codigo ya utilizado y que sabemos que funciona perfectamente. El auto-plagio es, a la hora de escribir *software*, una manera de ahorrarnos problemas.

### 8.1.1. Sentencias require() e include()

El lenguaje PHP incluye dos potentes funciones que nos facilitaran la tarea de reutilizar codigo en cualquiera de nuestros *scripts*. Utilizando cualquiera de ambas funciones, podemos abrir un archivo dentro de un *script* PHP en cualquier momento. Dicho archivo sera, generalmente, otro *script* en PHP o algun tipo de diseño en HTML.

Creemos un sencillo *script* en PHP. Veamos un ejemplo:

```
<!-- reusar.php -->
<?
echo "Este es un ejemplo de código reutilizable";
?>
```

Y ahora creamos un *script* nuevo llamado *prueba.php*, que haga una llamada a *reusar.php*.

```
<?
echo "A continuación aparecerá un ejemplo de código
reutilizable";
require ("reusar.php");
?>
```

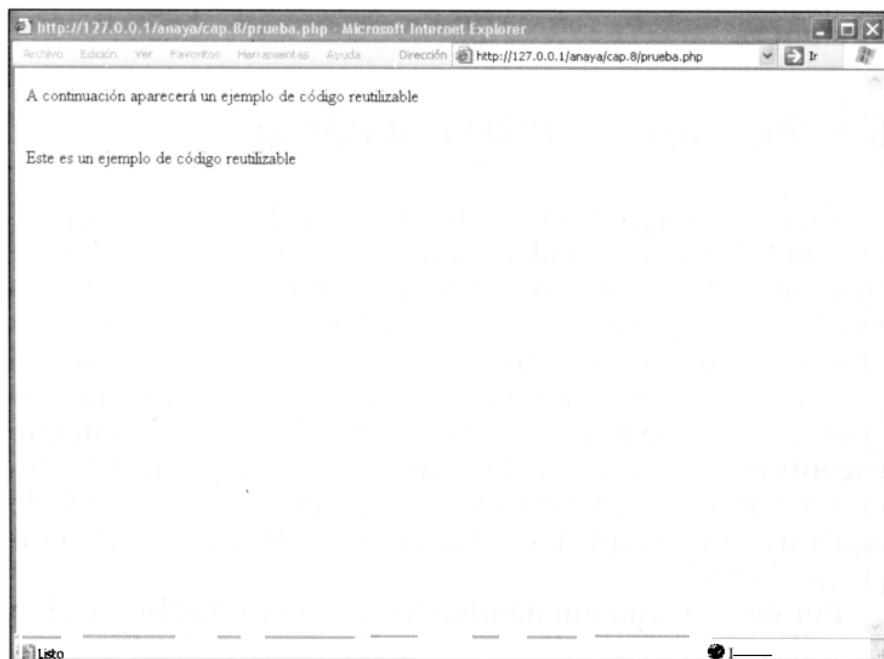


Figura 8.1. Ejemplo de código reutilizable.

A la vista del ejemplo anterior, la utilización de la sentencia *require()* no nos esta aportando demasiado en términos de ahorro de tiempo. Se trata de un ejemplo meramente didáctico para comprender su funcionamiento. A continuación, veremos algunas aplicaciones de *require()* e *include()* mucho mas prácticas.

Retomemos por unos instantes el ejemplo realizado en el capítulo 7 sobre la ordenación de los libros de nuestra biblioteca. Esta formado por numerosos *scripts* diferentes

que realizaban conexiones distintas con la base de datos. Podriamos haber simplificado esa tarea, en terminos de lineas de codigo y de utilización del servidor, creando un *script* con el siguiente texto.

```
<!-- conexion.php -->
<?
$cnx =
mysql_connect('localhost','tigreton','anais');
mysql_select_db('biblioteca');
?>
```

Y en el resto de *scripts* sustituir estas líneas de codigo por la sentencia:

```
require ("conexion.php");
```

De esta sencilla manera nos ahorramos codigo, y evitamos sobrecargar el servidor con demasiadas conexiones al realizar todas las consultas a traves de \$cnx en todos los *scripts*.

### 8.1.2. Diferencias entre require() e include()

El funcionamiento de ambas sentencias es muy similar, tanto, que en la mayor parte de los casos podemos utilizar cualquiera de ellas sin notar la menor diferencia.

Cuando en un *script* incluimos la sentencia include(), esta no se ejecutara si el resto del *script* no se ejecuta.

### 8.1.3. Utilización de plantillas gracias a include()

El conocimiento de la sentencia include() puede sernos sumamente útil a la hora de ahorrarnos tiempo cuando diseñemos nuestra pagina. Veamoslo mas claramente con un ejemplo. Supongamos que para nuestra pagina del colegio Maravillas hemos diseñado una bonita plantilla en HTML como la siguiente:

```
<!-- plantilla.php -->
<html>
<head>
<title>Colegio Maravillas</title>
```

```

</head>
<body bgcolor="#006666">
  <table width="90%" border="0" cellspacing="0"
cellpadding="0"
    height="90%" align="center">
    <tr valign="top" bgcolor="#339999">
      <td>
        <table width="100%" border="0"
cellspacing="0"
          cellpadding="0">
          <tr bgcolor="#99CCCC" valign="middle">
            <td height="130">
              <p align="center"><font face=
"Trebuchet MS , Arial"
size="6" color=
"#FFFFFF">Bienvenido al <u>Colegio
Maravillas</u></font></p>
              <p align="center"><font face=
"Arial, Helvetica,
sans-serif" size="4" color=
"#006666">Esta es la
página web dinámica
de antiguos alumnos del
colegio</font></p>
            </td>
          </tr>
        </table>
      </td>
    </tr>
  </table>
</body>
</html>

```

Una vez que tenemos la plantilla, en vez de reescribirla en todos nuestros *scripts*, solo tenemos que llamarla mediante la sentencia **include()**, y nos ahorraremos el tedioso proceso de copiarla.

```

<!-- ejemplo.php -->
<?
echo "Plantilla de ejemplo";
require ("plantilla.php");
?>

```

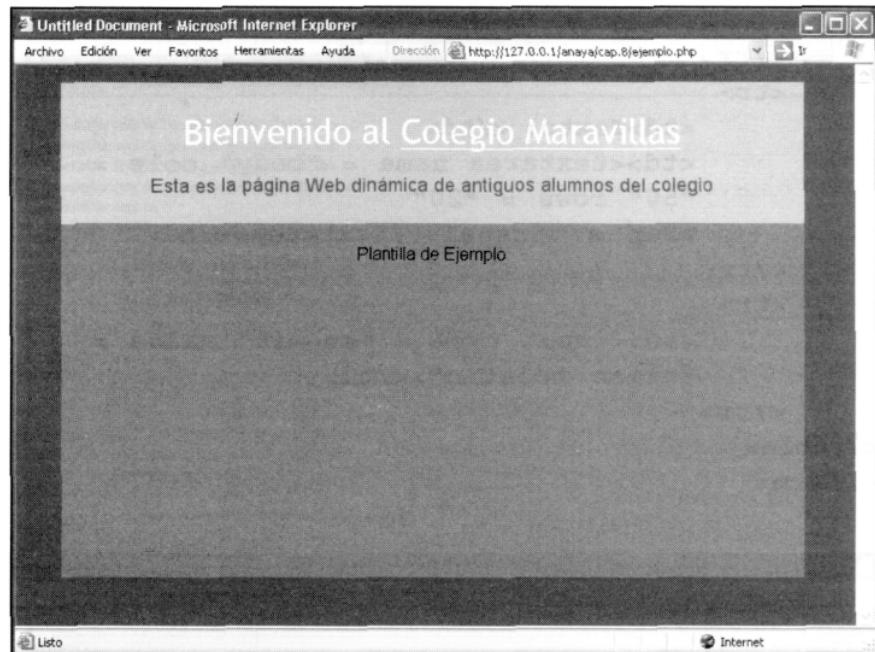


Figura 8.2. Ejemplo de diseño de plantilla.

## 8.2. Creación de una lista de correo

Ricardo se ha planteado incrementar las posibilidades de su pagina creando una lista de correo con la que mantener informados a todos los antiguos alumnos del colegio Maravillas. Con lo que ha aprendido hasta el momento en esta guía, se siente preparado para realizarlo el mismo.

Simplificando al maximo las funcionalidades de una lista de correo, vemos que se trata de un formulario donde escribir el boletín, y una herramienta que conecte con nuestra base de datos, recupere las direcciones de correo de todos los inscritos en nuestra pagina, les mande automaticamente los correos, y nos de una confirmación del envío.

El elemento esencial de este *script* es, lógicamente, la función **sendmail()** que vimos en el capítulo 6.

```
<!-- boletin.html -->
cform action = "mandarboletin.php" method = "post">
<table>
  <tr>
    <td>Título: </td>
    <td><input type = "text" name =
```

```

    "subject" size="70">></td>
</tr>
<tr>
    <td>Texto: </td>
    <td><textarea name = "body" cols =
    "50" rows = "20"
    wrap = virtual></textarea></td>
</tr>
<tr>
    <td><input type = "submit" value =
    "crear boletín"></td>
</tr>
</table>
</form>

```



**Figura 8.3.** Pagina de creación del boletín.

Este primer *script* no debe presentarnos ninguna dificultad, pues todos sus elementos los hemos visto numerosas veces a lo largo de la presente guía.

Se trata de un sencillo formulario en el cual escribimos el título y el texto del boletín que queremos mandar en la tabla creada para ello, y se envía el contenido al *script* **mandarboletin.php** por el método **POST**.



Figura 8.4. Envio del boletin.

```
<!-- mandarboletin.php -->
<?php
if (($subject == "") || ($body == ""))
{
header("Location : sendaletter");
exit;
}
else
{
$connection = mysql_connect('localhost' , 'Epi' ,
'Blas')
or die("Imposible establecer la conexión");
$db = mysql_select_db("alumnos" , $connection)
or die ("No se puede crear la conexión");
$sql_query = "SELECT email FROM alumnos";
$result = mysql_query($sql_query) or die("cannot
execute query");
$header = "From : Colegio Maravillas";
while ($row= mysql_fetch_array($result))
{
$address = $row[0];
mail($address , $subject , nl2br($body) , $header);
echo ("boletín enviado a: $address<br>");
```

```
}

echo ( "Completed sending emails" );
}

?>
```

En este *script* es donde empleamos las posibilidades del lenguaje PHP en combinación con bases de datos MySQL para poder enviar el boletín escrito en el *script* anterior automáticamente a toda la gente inscrita en nuestra base de datos.

Lo primero es analizar si tanto el encabezado del boletín como su texto están completos, cosa que haremos mediante la sentencia condicional:

```
if (($subject == "") || ($body == ""))
```

En caso de que ambos campos hayan sido correctamente rellenados, procedemos a conectar con la base de datos Alumnos.

```
$db = mysql_select_db("alumnos" , $connection)
```

Y a continuación, de nuestra tabla Alumnos, que es donde tenemos recogidos los datos de todos los antiguos alumnos que han querido apuntarse, seleccionamos el campo donde se almacena el correo electrónico.

```
$sql_query = "SELECT email FROM alumnos";
```

Una vez que ya tenemos seleccionadas las direcciones de correo electrónico de todos aquellos a los que vamos a mandar el boletín, solo nos quedará recurrir a la función `mail()` (tal y como vimos en el capítulo 6) para que nos haga todo el trabajo sucio.

```
while ($row= mysql_fetch_array($result))

$address = $row[0];
mail ($address , $subject , nl2br($body) , $header);
echo ("boletin enviado a: $address<br>");
}
```

Mediante el bucle `while`, vamos seleccionando uno por uno todos los campos recogidos en nuestra búsqueda de la base de datos, y se procederá al envío del boletín.

Las cuatro variables que vemos en la sentencia `mail` adquieren los siguientes valores:

1. `$address`. La dirección de correo electrónico a la cual mandamos el boletín. Mediante un bucle, vamos alternando entre todas las que hay almacenadas en nuestra base de datos.
2. `$subject`. El encabezado del mensaje. Definido en el *script* `boletin.html` y enviado via `POST`.
3. `$body`. El texto del mensaje. Definido en el *script* `boletin.html` y enviado via `POST`.
4. `$header`. El asunto del mensaje; lo hemos definido al principio de este *script*.

## PostNuke

### 9.1. Que es *PostNuke*

Ya hemos hablado anteriormente de la comunidad de programadores de *software libre*. Son un grupo de personas de todas las edades y nacionalidades que colaboran entre sí con el fin de crear nuevas aplicaciones informáticas que ceden de manera gratuita a todo el que quiera utilizarlas.

A ellos se debe el enorme éxito que ha adquirido el lenguaje PHP y la base de datos MySQL, y son centenares los programas y aplicaciones que se desarrollan anualmente utilizando estos lenguajes, y que podemos utilizar libremente en nuestra página Web con la simple condición de respetar su licencia GPL.

Ya hemos visto un ejemplo en el capítulo 2 al utilizar el programa *PHP Home Edition 2*, que nos simplificaba enormemente la tarea de instalar y configurar nuestro "trío mágico" (PHP, MySQL y Apache).

Nuestro amigo Ricardo, a la hora de preparar su página Web del colegio Maravillas, también podría haber recurrido a la comunidad de *software libre* y, al explicarles lo que necesitaba para su página (un registro de los antiguos alumnos, un foro, votaciones, poder publicar automáticamente los mensajes de los antiguos alumnos, lista de correo, etc.), sin duda le hubieran recomendado la aplicación *PostNuke*.

*PostNuke* es un sistema de administración de contenidos (*Content Management System*). Sus siglas, CMS, son de frecuente utilización en el mundo de Internet, y nos permite crear y gestionar una página Web dinámica.

Completamente escrito en PHP y MySQL, *PostNuke* nos permite crear un sitio Web donde los usuarios pueden

acceder libremente, registrarse en su base de datos, incluir sus opiniones y noticias, votar en encuestas, descargar todo tipo de archivos, participar en chats y listas de correo e, incluso, personalizar ellos mismos el aspecto que tendrá la página cada vez que la visiten.

*PostNuke* es la solución ideal para todos aquellos que desean crear una comunidad de usuarios sumamente activa donde todos puedan participar e intercambiar sus opiniones. Sin duda, eso es exactamente lo que necesitamos para nuestra página Web del colegio Maravillas, y aunque a lo largo del libro hemos desarrollado una página que nos sirve a la perfección a nuestros propósitos, el utilizar una aplicación como *PostNuke* presenta algunas ventajas:

- Es una aplicación probada por miles de usuarios en todo el mundo, por lo que los *bugs* o errores son simplemente inexistentes.
- Existe abundante documentación sobre su instalación y funcionamiento.
- Está en continuo desarrollo, por lo que cada poco tiempo aparecen nuevos módulos que mejoran las funcionalidades de *PostNuke*, y actualizaciones que lo transforman en una herramienta más completa y segura cada vez.
- Y todo esto... absolutamente gratis. ¿Alguien da más?

Alguien podría pensar: ¿qué necesidad tengo de aprender PHP y MySQL y diseñar mi propia página Web si puedo acceder libremente a aplicaciones tipo *PostNuke*? Sin embargo, y el caso de *PostNuke* es uno de ellos, su instalación y manejo no siempre son intuitivos, y es necesario poseer unos sólidos conocimientos de ambos lenguajes para sacarle todo el partido posible. Además, al saber programar en PHP y MySQL, podremos introducir nuestras propias mejoras en el código fuente y compartirlas con el resto de usuarios de *PostNuke*.

Pero, no todo son ventajas. *PostNuke* también presenta algunos inconvenientes que pueden hacernos considerar el no utilizarlo y realizar nuestra propia página.

Al ser una aplicación estándar, todas las páginas realizadas con *PostNuke* se parecen (y en muchos casos son idénticas). Aunque insuperable en cuanto a prestaciones y seguridad, si deseamos una página realmente exclusiva y única, tendremos que realizarla nosotros mismos.

## 9.2. Características de *PostNuke*

PostNuke no es un programa original, sino un derivado de PHP-Nuke, el pionero en el mundo de los CMS (Sistemas de administración de contenidos), creado por Francisco Burzi, y que rápidamente alcanzó una gran popularidad.

Pronto salieron diversos competidores (o imitadores, según algunos) de este PHP-Nuke que mejoraban las funcionalidades del mismo. De todos ellos, el más popular, seguro y completo es PostNuke, pero no por ello queremos dejar de mencionar que PHP-Nuke es un programa plenamente vigente, como demuestran los miles de usuarios que continúan empleandolo. Si quiere juzgarlo usted mismo, en su página Web ([www.phpnuke.org](http://www.phpnuke.org)) encontrara completa documentación, así como el código fuente.

La página Web oficial de *PHP-Nuke* en España es: <http://phpnuke-es.net/>.

Características fundamentales de PostNuke:

- Administración vía Web. Nada de complejos programas instalados en nuestro disco duro. Podemos administrar y controlar nuestro sitio Web directamente desde su página Web, gracias a su sistema de contraseñas.
- Interfaz personalizable por parte del usuario: cada usuario puede elegir la apariencia que tendrá *PostNuke* al entrar desde su ordenador, eso sí, de entre una serie de diseños predeterminados.
- Opción para editar todos los artículos publicados por los usuarios, incluyendo la posibilidad de publicación automática, o con aprobación previa.
- Traducido a más de 30 idiomas, incluyendo castellano, catalán, gallego y euskera.
- Posibilidad de realizar encuestas e incluir comentarios por parte de los usuarios en ellas.
- Completo gestor de *banners*, que permite incluir publicidad en la Web y controlar el número de visitas que produce.
- Completa base de datos que recoge los datos de todos los usuarios registrados.
- Indica cuantos usuarios están visitando la página en ese momento.
- Completo sistema de estadísticas.

- Fácil integración de todo tipo de módulos adicionales, entre ellos *chat* en tiempo real.

## 9.3. Documentación de *PostNuke*

Son numerosas las Web donde encontrar información sobre esta aplicación y descargar su código fuente.

Las más importantes son:

[www.postnuke.com](http://www.postnuke.com)

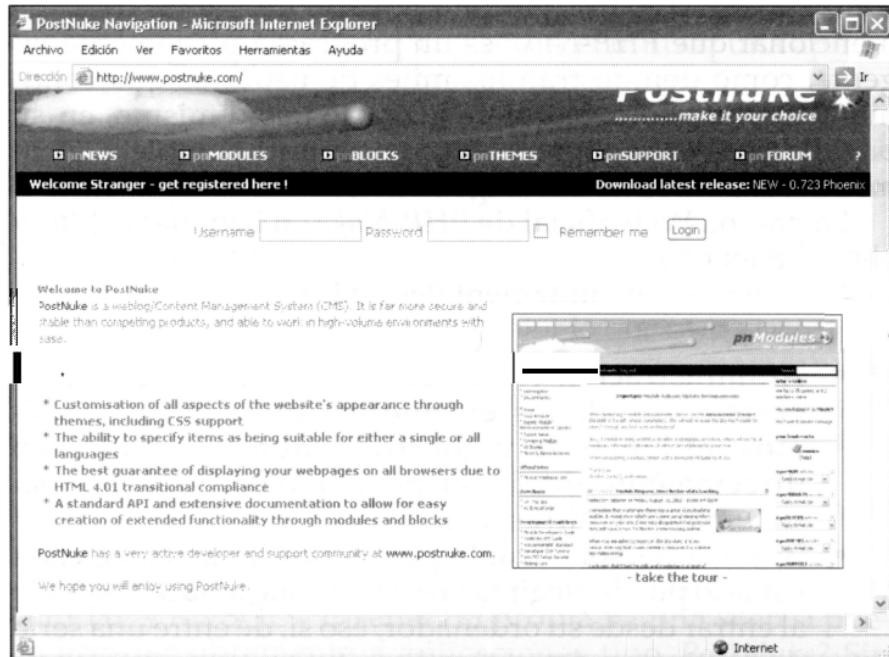


Figura 9.1. Página inicial de [www.postnuke.com](http://www.postnuke.com).

Se trata de su página oficial, y es el mejor sitio para estar al tanto de todo tipo de noticias, descargas, nuevos manuales, etc., sobre *PostNuke* (si dominas el inglés, claro está). Es el lugar donde (lógicamente) primero se publican las actualizaciones. Incluye vínculos a todos los *sites* oficiales de *PostNuke* alrededor del mundo.

<http://www.postnuke-espanol.org/>

Si prefieres el castellano, no dudes en visitar esta página. La comunidad española de *PostNuke* es sumamente activa, sobre todo a la hora de las traducciones. En su foro estarán encantados de ayudarte con cualquier tipo de duda que pueda quedarte, eso sí, después de leer sus guías de instalación, configuración y uso de *PostNuke*.

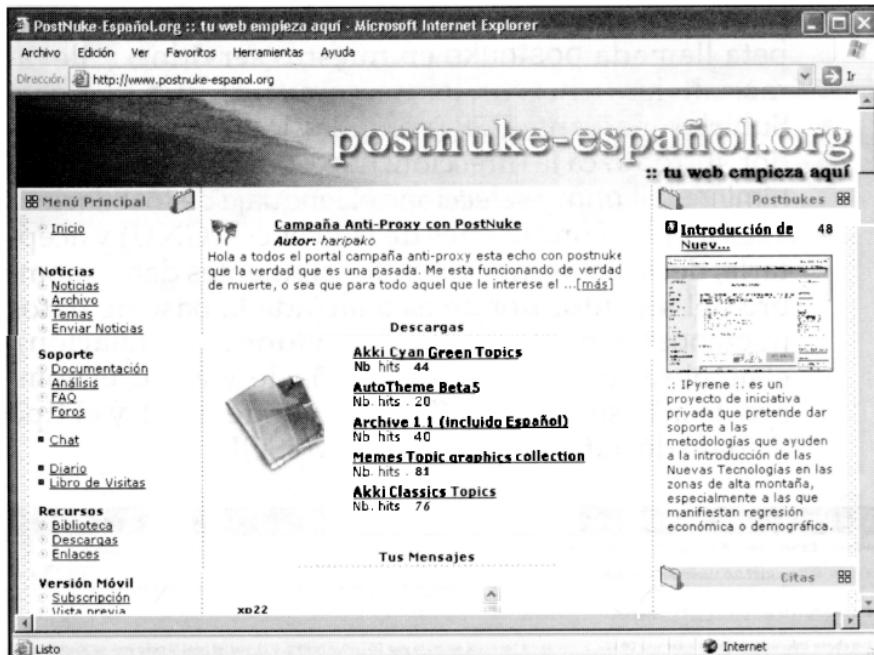


Figura 9.2. Web oficial de PostNuke en castellano.

## 9.4. Instalacion de *PostNuke*

¡Llegó el momento de pasar a la acción! Advertimos que la instalacion de PostNuke no es tan sencilla como la mayoría de los programas comerciales a los que estamos acostumbrados. Sin embargo, con los conocimientos que hemos adquirido a lo largo de esta guía, no deberemos tener demasiadas dificultades si seguimos los pasos indicados a continuación.

- Obtención del código fuente. El primer paso será conseguir la última versión de PostNuke. Como hemos indicado, el lugar más indicado es su página oficial, [www.postnuke.com](http://www.postnuke.com), y visitar su zona de descargas (*downloads*, en inglés), o la página de inicio de *PostNuke* español. En el momento de escribir estas líneas, es la .714. Dependiendo de si nuestro sistema operativo es *Windows* o *Linux*, elegiremos el formato de compresión (zip, si es *Windows*; tar, si es *Linux*), y procedemos a descargarlo en nuestro disco duro. Asimismo, deberemos descargar el paquete de idioma español en el área **descargas** de esa misma página.

- Descomprimir en servidor Apache. Abrimos una carpeta llamada **postnuke** en nuestro servidor, y llevamos allí los ficheros y los descomprimimos.
- Subirlos mediante FTP (pantalla). A través del buscador, introduzca la dirección: **http://127.0.0.1/postnuke/html/install.php**, y seleccione el lenguaje de instalacion.
- Tras leer las condiciones de la licencia (GNU) y aceptarla, deberemos introducir los siguientes datos: nombre del servidor donde esta alojada la base de datos (**localhost**, si coincide con el servidor de instalación), el nombre de usuario y contraseda de MySQL, el nombre de la base de datos (**Rogue**, por defecto), y el tipo debase de datos (logicamente, MySQL)

PostNuke Installation - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección: <http://127.0.0.1/postnuke/html/install.php>

PostNuke Installation

**Database Information** Please enter your DB info. If you do not have root access to your DB (virtual hosting, etc), you will need to make your database before you proceed. A good rule of thumb, if you cannot create databases through phpMyAdmin because of virtual hosting, or security on MySQL, then this script will not be able to create the db for you. This script will still be able to fill the database, and will still need to be run.

DatabaseHost: localhost

DatabaseUsername: tigreton

DatabasePassword: enais

Database Name: Rogue

Table Prefix (for Table Sharing): nuke

Database Type: MySQL

Site is for intranet or other local (non-internet) use

You must set the "intranet" option if your site does not use a fully-qualified host name for access. Examples of fully qualified hostnames are www.postnuke.com and foo.bar.com. Examples of hostnames that are not fully qualified are foo.com, localhost, and mysite.org. If you do not set this parameter properly you might not be able to log in to your site.

Submit

Thank you for trying PostNuke and welcome to our community.

**Figura 9.3. Pantalla de configuración de PostNuke.**

**Nota:** En caso de que estemos instalando PostNuke en un servidor de pago, será el proveedor quien nos debe proporcionar el nombre de usuario, contraseña, y nombre de la base de datos en que haremos la instalación.

Tras verificar que estos datos son correctos, y ver la lista de agradecimientos de los creadores de *PostNuke*, el programa creara las bases de datos necesarias para su funcionamiento, y procedera a finalizar la instalacion.

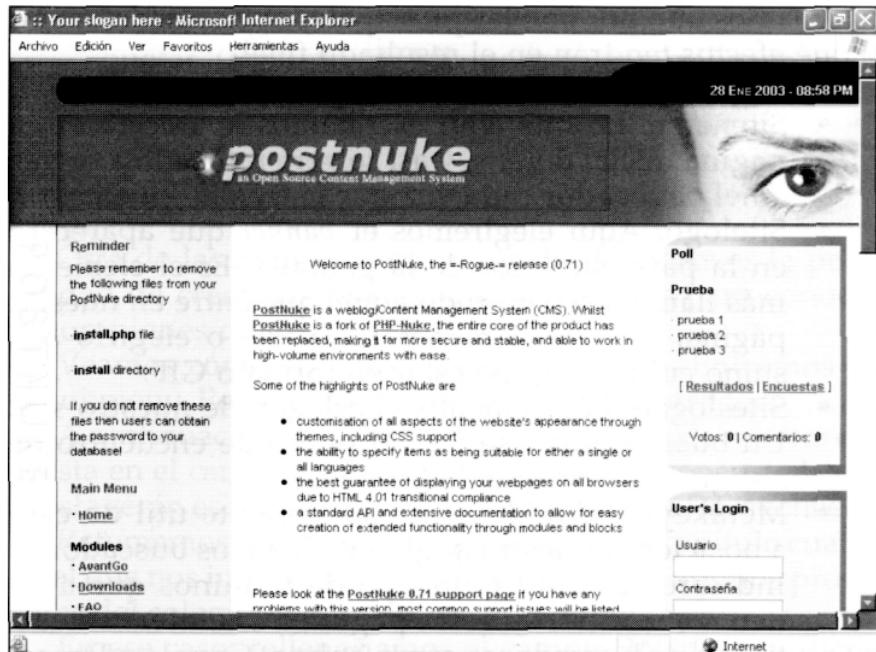


Figura 9.4. Imagen Inicial del PostNuke.

## 9.5. Configuración de *PostNuke*

¡Ahora empieza lo divertido! *PostNuke* esta perfectamente instalado, y ahora aprenderemos a configurarlo segun nuestras necesidades. De momento, tenemos una configuracion por defecto, que iremos modificando poco a poco para adaptarlo a nuestras necesidades.

El primer paso sera borrar de nuestro directorio el archive `install.php`, y el subdirectorio `html/install`. De esta manera, evitamos que ningun usuario malintencionado nos modifique la configuración o acceda a los datos de los usuarios almacenados en nuestra base de datos.

El siguiente paso es entrar en el menu Administración, al que solo podremos acceder mediante contraseña. Este sera el menu esencial para hacer todo tipo de modificaciones y adaptaciones en la pagina.

### 9.5.1. Opciones del menu Administración

#### Submenu Settings

Entremos en el submenu Settings. Alli realizaremos las modificaciones esenciales en la apariencia de *PostNuke*.

Veamos uno por uno las principales opciones del menu y que efectos tendrán en el resultado final.

- Sitename. Lo que aqui escribamos aparecera en la pagina inicial del *site*, y asi se verá reflejado a su vez en el navegador cada vez que entremos en ella.
- Sitelogo. Aqui elegiremos el *banner* que aparecera en la parte superior de la pantalla. Es el elemento mas llamativo para todo aquel que entre en nuestra pagina, asi que debemos diseñarlo o elegirlo con sumo cuidado. Debe estar en formato GIF.
- Siteslogan. Logicamente, el eslogan de nuestro site. Un buen ejemplo seria: "El punto de encuentro de antiguos alumnos del Maravillas".
- Metakeywords. Elemento sumamente util de cara a posicionar nuestra pagina Web en los buscadores. Indicaremos aqui todo tipo de terminos que ayuden a encontrar nuestra pagina a los internautas a traves de buscadores como Yahoo, Google o Dmoz. Un ejemplo seria: "Colegio Maravillas, Alumnos, Amigos, Reunion, etc."
- Adminemail. Nuestra dirección de correo, por si alguien quiere ponerse en contacto con nosotros.
- Defaultheme. Los *themes* son la apariencia visual que deseamos que tenga nuestra pagina. Incluye varios por defecto, pero pueden descargarse mas en la pagina oficial. Pruebelos uno por uno y elija el que mas le guste. Con el tiempo, usted puede intentar crear nuevos *themes* y compartirlos con la comunidad de usuarios de *PostNuke*.
- Timezoneoffset. Aqui seleccionaremos la franja horaria en que nos encontramos.
- Startpage. Seleccionaremos cual de todas las páginas de *PostNuke* verá el visitante cuando entre en la pagina de inicio.
- Sellanguage. Seleccionamos el idioma de los muchos que incluye esta version de *PostNuke*.
- Footerline. Configuración de los elementos que aparecen en la zona inferior de la pagina de inicio, incluyendo los vinculos. Podemos modificarlo, pero teniendo en cuenta que es una referencia a *PostNuke*, por lo que aconsejo que se deje sin tocar, para informar a todos nuestros visitantes que tenemos un *site* tan fabuloso gracias a *PostNuke*.

- HTMLallowed. Aquí definiremos cuantos de las etiquetas (*tags*) de HTML podrán ser utilizadas por nuestros visitantes, y cuales no.

## Submenu Polls

Otra de las atractivas opciones de *PostNuke* es la posibilidad de crear nuestras propias encuestas para conocer las opiniones de nuestros visitantes.

Veamos como crear una nueva encuesta. Entraremos en el submenu Polls dentro del menu Administrador.

A continuación, estableceremos la pregunta de nuestra encuesta en el campo Polltitle. Las distintas opciones de la encuesta serán escritas en los distintos campos de Polleachfield.

Retomemos el ejercicio desarrollado en el capítulo cuatro en el que nos interesaba conocer cual de los antiguos profesores del colegio Maravillas era mas popular.

En ese caso, rellenaremos el campo Polltitle con dicha pregunta, y los campos de Polleachfield con los nombres y mote de los diversos profesores.

Si ahora volvemos a la pagina de inicio, veremos nuestra nueva encuesta en la zona derecha del menu, con la posibilidad de que, además del voto, los antiguos alumnos intercambien comentarios.

## Submenu Admin

- **Messages.** Esta opción es la que nos permitirá configurar los mensajes de bienvenida con los que se encontrara todo visitante cuando entre en la pagina de inicio. Consta de titulo y mensaje propiamente dicho, y como característica destacada, señalaremos el que podemos configurar el mensaje para que lo vea todo el mundo, solo usuarios registrados, o incluso todos aquellos que tengan la condición de administradores.
- **Banners.** En el momento en que nuestra pagina obtenga un trafico considerable de visitas, podríamos plantearnos introducir publicidad con el fin de sufragar los gastos. Y en el campo de la publicidad via Internet, el *banner* es el elemento mas frecuente. Con esta herramienta, podremos configurar los *banners* que queremos que aparezcan en

nuestro sitio Web. Incluye completas estadísticas sobre nuestros clientes y el numero de visitas que obtiene cada *banner*, lo cual lo hace ideal a la hora de comprobar su eficacia.

- **Blocks.** Una herramienta muy necesaria para, entre otras cosas, personalizar un poco mas nuestro *site*. Con el, podremos cambiar los nombres de todos los elementos de nuestra pagina, como las encuestas, el menu de categorias, las noticias, etc., asi como su situación en el *site*. Es otro de los elementos fundamentales de personalización de *PostNuke*, aunque, como hemos comentado, por mas que nos esforzemos, todos los sitios Web desarrollados con *PostNuke* siempre van a tener una apariencia similar.
- **FAQ.** Otro de los elementos fundamentales en toda pagina Web bien diseñada. FAQ (abreviatura de *Frequently Asked Questions*, Preguntas mas frecuentes). Conviene identificar todas las dudas sobre navegabilidad que se puedan encontrar en nuestra pagina Web y responderlas en esta sección. Esto ayudara mucho a nuestros visitantes a sentirse comodos, y nos ahorrara tener que contestar una y otra vez los mismos correos electronicos con preguntas del tipo “¿Cómo puedo registrarme? o ¿Qué es el colegio Maravillas?”.
- **Modules.** Muy similar a la herramienta *blocks*, nos permitira traducir, configurar y colocar a nuestro gusto todos los elementos del menu en nuestro *site*.

Veamoslo mas sencillamente con un ejemplo. En el menu existe una opcion llamada **Submit news**. Debemos traducirla al castellano, pues consideramos que el intercambio de mensajes es esencial entre los antiguos alumnos del colegio Maravillas.

En el menu **Administración**, pulsemos en la opcion **Modules**, y a continuación en **Listar**.

Nos apareceran todas las opciones del menu por orden alfabetico. Vamos hasta **News** y pulsamos en **Editar**.

Nos apareceran dos campos de texto: uno para cambiar el nombre y otro para la descripción. Elegimos el **Nombre** y la **Descripción** nuevos, y lo validamos. Si ahora actualizamos la pagina, veremos como ha cambiado al nuevo nombre, y ha ocupado su nuevo lugar en el menu, siguiendo el orden alfabetico.

## 9.6. Utilización de *PostNuke*

Una vez que nuestro sitio Web este perfectamente instalado, configurado y correctamente posicionado en buscadores, es de prever que cientos de antiguos alumnos comiencen a utilizar nuestra pagina para encontrar a sus antiguos alumnos. Para ello, deberemos estar familiarizados con las tareas mas frecuentes de *PostNuke*.

### 9.6.1. Validación de mensajes

Nuestros visitantes podran dejar sus mensajes en el *site* a traves del submenu **Submit news**, o como hayamos decidido llamarlo nosotros durante el proceso de configuracion. Asimismo, durante ese proceso habremos elegido si los mensajes se publican directamente en la pagina o, la mas recomendable, si dichos mensajes debemos validarlos nosotros antes.

Cada vez que entremos en el *site* bajo la contraseña de administrador, veremos un mensaje en la parte inferior del menu izquierdo con la frase **Contenidos en espera**. No tenemos mas que pulsar sobre el para acceder a un menu donde apareceran todos los mensajes enviados por nuestros visitantes y pendientes de ser publicados. Aqui podremos rechazarlos directamente, modificarlos o aceptarlos para su publicación, ya sea en la pagina de inicio o en cualquiera de las otras secciones. Tambien podemos decidir si a este mensaje se le podran incluir comentarios o no.

### 9.6.2. Control de estadisticas

Hay ciertos tipos de preguntas que todo aquel que administra una pagina Web se hace con bastante frecuencia: ¿Estara gustando mi pagina a los visitantes? ¿Cuál sera su sección favorita? ¿A que horas hay mas trafico de visitas?, o incluso ¿Qué sistema operativo es el mas usado por mis visitantes?

Todas esas preguntas y mas pueden ser contestadas en la sección **STATS** de nuestro menu principal, en el que mediante graficos podremos analizar el comportamiento de la gente que nos visita.

### 9.6.3. Lista de miembros

Si los visitantes estan a gusto con nuestra pagina, se registraran con el fin de gozar de todos los privilegios. En la sección **Member List** podremos visionar, por orden alfabetico, todos aquellos que efectivamente lo han hecho. Allí nos aparecera su nombre, el apodo con el que desean ser vistos en la pagina, su correo electronico, su URL, los privilegios especiales que les hayamos concedido, y cuántos de ellos estan ahora mismo en linea. Cuanto mas crezca, sera señal de que nuestra pagina esta teniendo éxito.

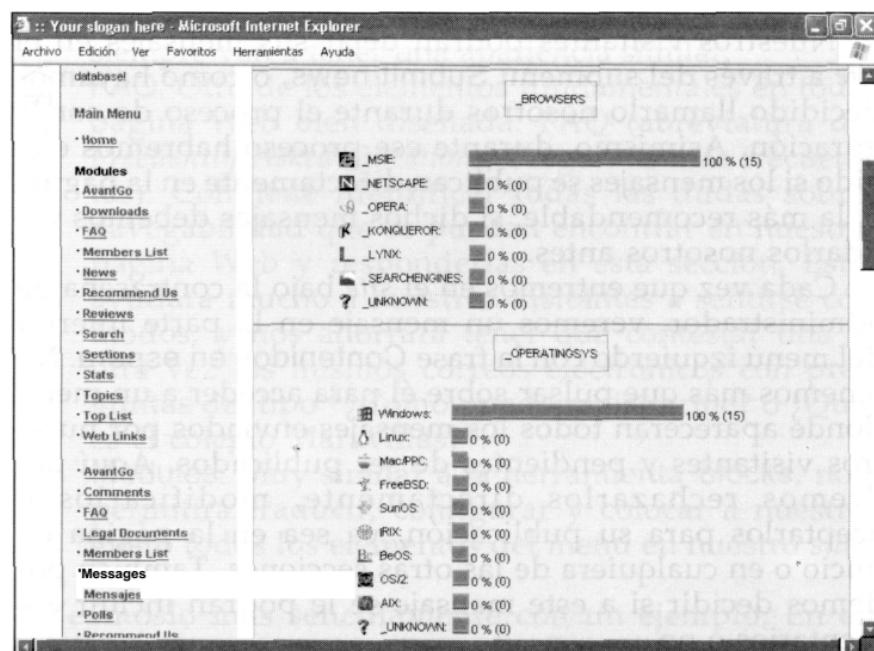


Figura 9.5. Ejemplo de pantalla de estadísticas.

## 9.7. Conclusion

Como hemos visto a lo largo del presente capítulo, *PostNuke* nos ofrece una solucion gratuita, plena de funcionalidades, y que representa un magnifico exponente de las posibilidades que una pagina Web dinamica en PHP y MySQL pueden ofrecernos.

En su contra, se puede alegar que nuestra pagina carecera de originalidad y personalidad propias. De nosotros depende optar por una solucion u otra.

## *osCommerce:* solución *Open Source* de Comercio Electronico

### 10.1. Introducción

Quizá en alguna ocasión se haya planteado la opción de comercializar sus productos vía Internet. Es una idea sumamente atractiva, pues nos permite ahorrarnos la mayor parte de los costes fijos, como puede ser el alquiler de la tienda o el pago de la nómina de nuestros empleados. Asimismo, una tienda en Internet puede recibir pedidos las 24 horas del día, 7 días a la semana.

Sin embargo, no debemos desdeñar las dificultades que presenta abrir una tienda en Internet. Esencialmente, dificultades tecnológicas. Nuestra primera opción suele ser pedir presupuesto a una empresa informática para que nos monten la tienda de nuestros sueños. Y, probablemente, sean capaces de hacerlo. De lo que no estén tan seguros es de si serán capaces de pagar la factura. Los profesionales competentes en el mundo de Internet son tan escasos como bien pagados, y el contratarlos puede estar por encima de nuestras posibilidades; y más, si a lo único que aspiramos es a una pequeña tiendecita.

La otra opción es adquirir algún programa comercial de *software* que nos prometen configurarnos nuestra propia tienda en cuestión de minutos por un precio casi siempre inferior a los 100 EUR.

Y realmente estos programas no mienten. Sin embargo, su instalación no suele durar minutos, sino más bien días enteros, y los resultados pueden dejarnos un tanto fríos. La tienda tendrá una apariencia absolutamente idéntica a la de todas las tiendas realizadas con el mismo programa; visualmente, no será muy buena; y sus opciones, en cuanto

a numero de productos, opciones comerciales, envio, etc., dejan mucho que desear.

Es por ello que la mayoría de pequeños proyectos comerciales en Internet rara vez llegan a buen puerto. Obien la inversion realizada no puede ser amortizada por el escaso volumen de ventas, o bien la apariencia de la tienda causa desconfianza entre los clientes potenciales.

Sin embargo, no debemos desanimarnos. Lo que nos diferencia del resto es la guia que tenemos entre las manos, gracias a la cual dominamos los fundamentos del lenguaje PHP/MySQL y, a su vez, conocemos algo de la comunidad de *software* libre y los increibles proyectos que ponen a disposición de todo aquel que quiera utilizarlos.

Y el campo del comercio electronico no es una excepcion. Son muchas las soluciones de comercio electronico de libre distribución a las que podemos acceder, pero, de todas ellas, optaremos por una cuya calidad rivaliza (y, en la mayor parte de los casos, supera) con cualquier desarollo a medida que la mejor consultora informática pueda desarrollar y cuyo precio, sencillamente no existe. Es gratuito. ¿Intrigado? Siga leyendo.

## 10.2. *osCommerce*

*osCommerce* es probablemente el proyecto de comercio electronico mas ambicioso y potente que se haya desarrollado jamás en el campo del *software* libre.

Esta desarrollado integralmente en PHP y MySQL, y diseiido especialmente para trabajar con servidores Apache, por lo que su codigo fuente nos sera conocido. Sin embargo, la enorme dimension del proyecto puede llegar a confundirnos al principio. Pero, si seguimos atentamente las instrucciones de este capitulo, podremos llegar a dominarlo.

El proyecto *osCommerce* empezo en mayo del aiio 2000 de la mano de Harald Ponce de Leon, y desde entonces, se ha convertido en una herramienta plenamente segura que utilizan alrededor de 800 tiendas *on line* alrededor del mundo.

Actualmente, un equipo internacional de programadores (dirigido por el propio Harald) se encarga de mejorar el proyecto e implementar nuevas funcionalidades. La ultima version estable es la 2.2, pero de seguro que este incansable equipo seguira mejorando el resultado final por muchos aiios.

Al ser cada día mas popular, es facil encontrar gente amigable y dispuesta a echarnos una mano cada vez que tengamos una duda con *osCommerce*.

La pagina oficial de *osCommerce* (en inglés, como era de esperar), con numerosos foros de ayuda es: <http://www.oscommerce.com/>.



Figura 10.1. Pagina oficial de *osCommerce*.

Existe asimismo una pagina de *osCommerce* en español, donde podrán brindarnos ayuda en nuestro idioma: <http://oscommerce.qadram.com/>.

### 10.3. Características fundamentales de *osCommerce*

Antes de entrar en materia, veamos algunas de las características fundamentales de *osCommerce*.

De cara al cliente, las posibilidades que nos ofrece son las siguientes:

- Presentación absolutamente profesional.
- Completo registro de los clientes, que incluye su histórico de pedidos, dirección, etc.

- Posibilidad de busqueda tanto por producto como por fabricante, autor, etc.
- Los clientes pueden escribir sus propias reseñas de los productos.
- Notificaciones por *e-mail* al cliente, incluyendo factura.
- Transacciones seguras con SSL.
- Lista de *Best.sellers*.
- Diversas posibilidades de pago: contra reembolso, tarjeta de credito, transferencia, etc.

osCommerce: soluciones OpenSource para comercio electrónico - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección  Ir

Enviar noticias Estadísticas Top 10

**Soporte**  
Foros Canal IRC Lista de correo

**En línea**  
Actualmente hay 3 invitados, 0 miembro(s) conectado(s).

Eres un usuario anónimo. Puedes registrarte aquí.

**User's Login**  
Nombre   
Contraseña   
Login

¿Todavía no tienes una cuenta? Puedes crear una. Como usuario registrado tendrás ventajas como seleccionar la

**osCommerce ??? FÁCIL No gracias**  
FJPL escribió: "Llevo varios días intentando instalar "osCommerce" y la verdad es que no encuentro por ningún lado las bondades del sistema en la versión 2.2 ni la anterior 2.1 que también he intentado instalar y me ha dado errores a pesar de supuestamente ser estable."

esta web :)

Enviado por ttm el Miércoles, 29 enero a las 03:27 (29 Lecturas) | Leer más... | 1674 bytes más | 0 Comentarios | Puntuación 0

Faltan cosas, las pongo abajo en un comentario.

Horrible, quitalo para no gastar ancho de banda

voto

**Resultados Encuestas**  
votos: 250 Comentarios: 9

**Past Articles**  
Martes, 29 octubre • Módulo de pago de la Caixa (3)  
Lunes, 28 octubre • Diseños Templates o themes para oscommerce (5)  
Martes, 22 octubre • Integración de OSCommerce con PostNuke (7)  
• Pasarelas y formas de pago por Sudamérica (0)

Internet

**Inventario para opciones**  
Anonymous escribió: "Para todos aquellos que como yo estaban buscando un módulo que controlara el inventario"

Figura 10.2. Pagina en castellano sobre osCommerce.

Para el administrador de la tienda (o sea, nosotros), presenta una serie de utilísimas características, entre las que destacaremos:

- Diseño plenamente sencillo.
- Posibilidad de añadir, quitar o modificar categorías, productos, fabricantes, clientes y reseñas.
- Estructura por categorías.
- Estadísticas de productos y clientes.
- Atributos de producto dinámicos.
- IVA plenamente configurable.
- Herramienta de Copia de seguridad (*Backup*).

Destacaremos los siguientes vínculos:  
<http://www.oscommerce.com/osCommerce/>

osCommerce, Open Source E-Commerce Solutions - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección <http://www.oscommerce.com/shops/goto,876> Ir

This is a live shop - please do not make any test orders here!  
Report bad link or non osCommerce related site | remove this frame-set

osCommerce

Wear with All

Displaying 1 to 10 (of 11 products) Result Pages: 1 [\[Next >\]](#)

[Listo](#) [Internet](#)

**Figura 10.3.** Ejemplo de tienda desarrollada en osCommerce.  
<http://www.oscommerce.com/shops/goto,876>

osCommerce, Open Source E-Commerce Solutions - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección <http://www.oscommerce.com/shops/goto,955> Ir

This is a live shop - please do not make any test orders here!  
Report bad link or non osCommerce related site | remove this frame-set

**Habanomania**

**Welcome to Habanomania.com**

**This month's special offers**

**NEW! - Important information. Please read here!**  
The lowest prices in the net for the best Cuban Cigars!

**Top > Catalog**

**Cigar Brands**

- Bolívar (4)
- Cohiba (11)
- Cuba (4)
- Fonseca (4)
- H. Upman (5)
- La Flor de Cano (2)
- La Gloria Cubana (3)
- Montecristo (10)
- Partagás (6)
- Punch (5)
- Rafael González (2)
- Ramon Allones (2)
- Rey del Mundo (1)
- Romeo y Julieta (3)
- Sancho Panza (5)
- Trinidad (1)
- Vegar Pobaina (5)

**Information**

- About us
- Shopping & Ordering
- Shipping Information
- Contact Us

**My Account** | **Cart Contents** | **Checkout**

**Shopping Cart**

0 items

**Currencies**

Euro

**What's New?**

**Bestsellers**

01. Romeo y Julieta Churchill (Tube 10)  
02. Montecristo N°4  
03. Montecristo N°2  
04. Montecristo N°1

[Listo](#) [Internet](#)

**Figura 10.4.** Otro ejemplo de tienda desarrollada en osCommerce.

<http://www.oscommerce.com/shops/goto,955>

## 10.4. Instalacion de *osCommerce*

La instalacion de *osCommerce*, si bien supone una verdadera pesadilla para muchisimos usuarios que abarrotan los foros con sus consultas y que, desesperados, abandonan el proyecto, puede ser un proceso sumamente sencillo si seguimos las siguientes instrucciones con cuidado.

### 10.4.1. Requisitos de instalacion

*osCommerce* no presenta ningun requisito especial de instalación, aparte de los que ya hemos visto, pero conviene tenerlos presentes para evitarnos quebraderos de cabeza.

- **Sistema operativo.** Funciona con sistemas operativos *Windows* (98,2000 o *XP*) de Microsoft o con *Linux*.
- **Servidor.** Compatible con el lenguaje PHP y con MySQL. Es decir, el servidor Apache es ideal para nuestros propósitos.

### 10.4.2. Como subir los archivos via FTP al servidor

Tras descomprimir el codigo del programa (.zip, si trabajamos con sistema operativo *Windows*, y .tar, si trabajamos con *Linux*), veremos dos grandes carpetas que almacenan toda la informacion: *admin* y *catalog*. Cada una de ellas contiene respectivamente la informacion referida a las dos grandes areas de trabajo de *osCommerce*: la del administrador, que nos permitira gestionar todos los aspectos de la tienda (alta de productos, precios, *stock*, etc.), y la del cliente, es decir, el aspecto de nuestra tienda tal y como la veran los visitantes.

Deberemos conservar este arbol de directorio cuando lo subamos via FTP a nuestro servidor.

---

*Nota:* Conviene asegurarse de que los archivos PHP son enviados en formato ASCII via FTP.

---

El siguiente paso sera establecer los niveles de permisos de las diferentes carpetas, utilizando el comando CHMOD.

Los permisos a establecer son los siguientes:

Directorio	Niveles de permiso
/admin/includes	chmod 755
/catalog/includes	chmod 755
/admin/includes/configure.php	chmod 777
/catalog/includes/configure.php	chmod 777
/catalog/images	chmod 777

Tabla 10.1. Niveles de permisos en los directorios.

### 10.4.3. Ejecutar el *script* de instalacion

Una vez que haya concluido el proceso de subir los archives al servidor, deberemos ejecutar el *script* de instalacion. Dicho archivo se encuentra en catalog/install/install.php. Por tanto, el URL a introducir en nuestro navegador sera:

<http://www.nuestroservidor.com/catalog/install/install.php>

Donde nuestroservidor.com sera el dominio sobre el que estemos trabajando.

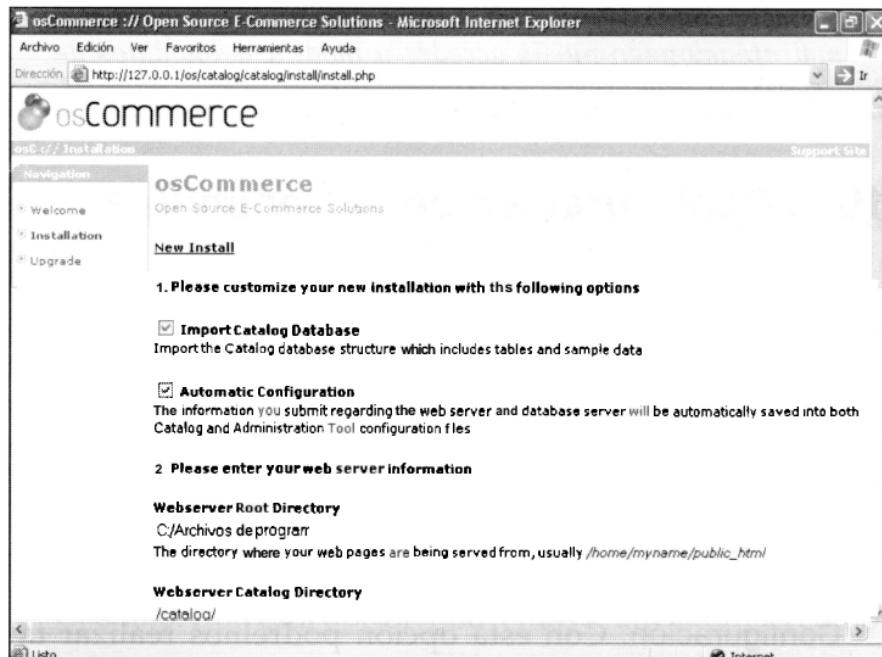


Figura 10.5. Pantalla de instalacion de *osCommerce*.

En esta pantalla, en los puntos uno y dos encontraremos una serie de casillas rellenas por defecto. Por lo

general, esos valores por defecto servirán perfectamente para nuestra Configuracion, pero aun asi deberemos repasarlos para ver que son correctos, ya que se refieren a las rutas de instalacion en el servidor.

En el punto tres deberemos introducir los datos del servidor, usuario, contraseña y base de datos.

- Database server (Servidor de base de datos): por lo general, localhost sera la Configuracion correcta.
- Username (Nombre de usuario): aqui introduciremos el nombre de usuario necesario para acceder al servidor de la base de datos.
- Password (Contraseña): aqui introduciremos la contraseña necesaria para acceder al servidor de la base de datos.

Tras pulsar **Continue**, habremos configurado completamente nuestra version de *osCommerce*.

---

*Nota: El proceso de instalacion **no estará** terminado **si no tomamos unas medidas** de seguridad, **como** es eliminar las carpetas catalog/install y catalog/incluye/configure.php.*

*De esta manera nos evitaremos que cualquier usuario malintencionado **pueda** acceder y **modificar** los datos de nuestra tienda.*

---

## 10.5. Configuracion de *osCommerce*

Una vez terminada la instalación de la tienda, aprenderemos a realizar las acciones fundamentales para convertir nuestra tienda en un lugar atractivo para los clientes.

Como hemos visto, nuestra tienda esta formada por dos interfaces: la del Administrador, al que solo deberemos acceder nosotros, y la del Cliente.

### 10.5.1. Opciones Administrador

Configuracion. Con esta opción podremos realizar tareas como:

- Establecer nombre de la tienda.
- Establecer nombre del propietario de la tienda.
- Establecer país, idioma y moneda de la tienda.

- Establecer el orden en que apareceran los productos en la tienda.
- Establecer la dirección, teléfono y *mail* de nuestra tienda fisica.



Figura 10.6. Modulo del administrador.

**Catalogo.** Con esta opción podremos realizar tareas como:

- Establecer el arbol de categorias y subcategorias de productos.
- Dar de alta los nuevos productos de nuestra tienda.
- Establecer todos los fabricantes de nuestros productos.
- Introducir los comentarios y descripciones que acompañan a nuestros productos.
- Establecer nuestras ofertas comerciales.

**Clientes.** Con esta opción podremos realizar tareas como:

- Comprobar los datos de todos nuestros clientes.
- Comprobar todos los pedidos realizados por nuestros clientes, con la posibilidad de comprobar la factura, el albaran, y el modo en que el pedido fue abonado.

**Informes.** Con esta opcion podremos realizar tareas como:

- Comprobar que productos son los mas vendidos, o simplemente los mas vistos por nuestros clientes, asi como el total de pedidos por cliente.

**Módulos de pago.** Con esta opcion podremos realizar tareas como:

- Establecer los métodos de pago que existiran en nuestra tienda. Es una elección sumamente importante a efectos de seguridad, teniendo en cuenta que cada método tiene sus pros, sus contras y, sobre todo, sus costes. Entre los más habituales en cualquier tienda estan: mediante tarjeta de crédito, transferencia, contra reembolso, o utilizando los servicios de alguna pasarela de pago externa tipo Paypal.

**Zonas/Impuestos.** Con esta opcion podremos realizar tareas como:

- Establecer los impuestos que gravaran nuestras ventas, dependiendo del país o de la provincia adonde se realice el envio. Por poner un ejemplo, la comunidad canaria o los envios al extranjero estan exentos del **IVA** que, sin embargo, rige en el resto del estado español.

**Localización.** Con esta opcion podremos realizar tareas como:

- Establecer la moneda y el idioma en que se harán las transacciones en nuestra tienda.

**Herramientas.** Con esta opcion podremos realizar tareas como:

- Realizar copias de seguridad de nuestra base de datos.
- Controlar el numero de visitas que obtienen los *banners* incluidos en nuestra tienda.
- Administrar los boletines informativos que podemos mandar a nuestros clientes.
- Conocer cuantos clientes se encuentran en tiempo real en ese momento.
- Información del servidor en el que esta alojada nuestra tienda.

## 10.5.2. Opciones Cliente (estructura de la tienda)

He aqui lo que realmente verá el cliente cuando entre a nuestra tienda. Aunque trae un diseño estandar, con un poco de esfuerzo, es facil adaptarlo a nuestras necesidades.



Figura 10.7. Modulo del cliente.

Lo primero que nos interesara cambiar sera el mensaje de bienvenida.

Por defecto, aparecera el texto:

**"Este es un catálogo de demostracion, cualquier producto comprado aquí NO será enviado ni cobrado. Cualquier información de estos productos debe ser tratada como ficticia.**

**Si desea descargar este catálogo de demostracion, o desea contribuir al proyecto, por favor visite la página de soporte. Esta tienda esta basada en osCommerce 2.2-CVS."**

Obviamente, no queremos que nuestros clientes vean semejante mensaje, asi que deberemos entrar en el

archivo `default.php`, que se encuentra en `catalog\includes\languages\espanol`, hacer los cambios pertinentes y volver a grabarlo, tal y como hemos aprendido a lo largo de este libro.

A izquierda y derecha de la pantalla, podremos encontrar todas las opciones que hemos configurado en la interfaz de Administrador: categorías, fabricante, novedades ... Prueba a hacer algún cambio en la interfaz de Administrador y comprobará como esta modificación aparecerá al instante en la tienda. De esta manera, podremos tener nuestra tienda siempre actualizada en tiempo real. Porque, de la misma manera que en cualquier gran almacén los escaparates y las implantaciones se cambian prácticamente a diario, nuestra tienda virtual debe mostrar actividad a los visitantes.

### 10.5.3. Compra simulada

Para conocer mejor el proceso de compra que deben realizar nuestros clientes, simularemos realizar una compra en nuestra tienda.

Elección del producto:

- Suponiendo que hemos montado una tienda de películas en formato DVD, buscaremos entre las diversas secciones (acción, drama, aventura, etc.), hasta encontrar el título que andamos buscando, *Matrix*. Tras leer las características, pulsamos el icono de Añadir a la cesta y, como esta es la única compra a realizar hoy, a continuación pulsamos en **Realizar Pedido**.

Registro en la tienda:

- A continuación, nos preguntará si ya estamos registrados o es nuestra primera compra. En este caso, diremos que somos nuevos clientes, para seguir el proceso de registro. Los datos necesarios para registrarse, así como los campos a llenar obligatorios podemos configurarlos desde la interfaz de Administrador.

Forma de pago:

- Una vez hecho el registro, deberemos elegir la **Forma de pago**, entre todas aquellas que hayamos configurado en el panel del administrador (cuenta

corriente, transferencia bancaria, contra reembolso). No olvidemos que cada forma de pago tiene sus pros y sus contras en temas como la seguridad y el coste adicional, y que en algunos casos podemos trasladar al precio final (como en el caso del contra reembolso), y en otros no (como el pago por tarjeta).

Finalización:

- Tras un breve mensaje de confirmación, nos indicara que nuestro pedido esta siendo procesado, y que en breve tiempo nos sera entregado.

Si ahora abrimos nuestra cuenta de correo, veremos dos correos nuevos enviados desde la tienda. En el primero se nos da la bienvenida como clientes, y en el otro se nos indica nuestro ultimo pedido y su precio. ¡Y sin que nosotros, como administradores de la tienda, hayamos tenido que tocar una sola tecla! Este es uno de esos pequeños detalles que hacen de *osCommerce* un programa tan genial.

#### 10.5.4. Gestión de los pedidos

Ahora cambiaremos nuestro rol en esta simulacion, y de clientes pasaremos a dueños de la tienda.

Supongamos que ha finalizado el dia y queremos conocer el numero de pedidos que ha tenido nuestra tienda.

El primer paso sera volver a la interfaz de Administrador y entrar en la opcion de **Pedidos**.

En la zona **Cièntes** veremos nuestro nombre tal y como lo introducimos durante el proceso de registro.

Y en la zona **Pedidos** veremos todos aquellos que tenemos pendientes de enviar, y entre ellos, lógicamente, nuestro encargo de la pelicula en DVD, *Matrix*.

Tenemos la opcion de **Editar** e **Imprimir** tanto la factura como el albaran, que son documentos que debemos adjuntar al cliente en su pedido.

El siguiente paso consistiria en ir al almacén, coger la pelicula y enviarsela al cliente, procediendo previamente al cobro, o una vez entregado si es contra reembolso.

Asimismo, entrariamos en la opcion de **Editar pedido** y cambiariamos el estado de este pedido de **Pendiente** a **Entregado**. Veremos que dicho proceso tambien es comunicado por correo electronico al cliente.

## 10.6. Conclusion

El proyecto *osCommerce* es, sin duda, uno de los mejores exponentes de hasta donde es capaz de llegar la comunidad del *software* libre.

Su calidad es comparable, cuando no superior, a la de la mayor parte del *software* comercial, y el apoyo y entusiasmo que nos brindaron el resto de usuarios de *osCommerce* cuando nos enfrentemos a alguna dificultad es sencillamente impagable.

# ***phpBB: solución Open Source para la creacion de foros personalizados***

## **11. ■ Introducción**

En los dos capítulos anteriores hemos aprendido a instalar y configurar programas completos de *software libre* que, como hemos indicado, pese a sus innegables virtudes, presentan el defecto de que con ellos nuestra página adolecerá de falta de originalidad.

Ahora aprenderemos como instalar y configurar una aplicación de creación de foros que tiene la inmensa ventaja de que puede ser instalada en cualquier página Web ya diseñada, a la que dotara de unas prestaciones verdaderamente profesionales a la vez que nos permitirá conservar la originalidad del diseño de nuestra página Web.

En nuestro caso, sería una buena opción diseñar nosotros mismos la página Web del Colegio Maravillas, para dotarla de personalidad y estilo propios, e instalar después una aplicación de foros como *phpBB* que adaptaríamos al diseño de nuestra página.

### **11.2. *phpBB***

Como ya hemos indicado, *phpBB* es una extraordinaria aplicación que nos permitirá crear nuestros propios foros personalizados e incluirlos en nuestra página Web.

El equipo de creación y desarrollo de *phpBB* está formado por diseñadores, programadores y testeadores de todo el mundo, encabezados por el fundador del proyecto, James Atkinson.

El programa, como podemos imaginar, ha sido creado con PHP y MySQL, aunque con la posibilidad de trabajar con otro tipo de bases de datos, como *PostgreSQL 7, Microsoft SQL Server o Microsoft Access*.

Actualmente, se encuentra en su versión número 2, y hace hincapié en la seguridad de la información y en la sencillez de instalación y manejo.

### **11.3. Características de *phpBB***

Las características esenciales de *phpBB* se pueden resumir en:

- Instalación sumamente sencilla.
- Capacidad de trabajar con diversas bases de datos (entre ellas, MySQL).
- Sistema de autorizaciones que evita accesos indeseados.
- Las contraseñas se guardan encriptadas en la base de datos.
- Foros ilimitados que se pueden organizar en tantas categorías como deseemos.
- Posibilidad de crear foros privados para usuarios escogidos.
- Gran diversidad de estilos y fuentes.
- Posibilidad de crear encuestas dentro de los foros.
- Notificación por correo electrónico cuando alguien responde a alguno de nuestros mensajes.
- Posibilidad de incluir emoticones.
- Incluye censor de palabras.
- Almacena la IP de todo el que deja un mensaje.
- El administrador puede enviar correo electrónico masivo a todos los usuarios del foro.
- Amplias posibilidades de configuración.

### **11.4. Instalación de *phpBB***

Este es uno de los puntos fuertes de esta aplicación, debido a su gran sencillez.

El primer paso, lógicamente, será obtener el código fuente de su página oficial (en formato .zip o .tar, según trabajemos con *Windows* o *Linux*): <http://www.phpbb.com/downloads.php>.



Figura 11.1. Página oficial de phpBB.

Tras descomprimirlo, procederemos a subirlo vía FTP a nuestro servidor, tal y como ya hemos explicado en los capítulos anteriores.

El siguiente paso consistirá en ejecutar el *script* *install.php*, y deberá aparecernos la siguiente pantalla.

En ella, deberemos introducir los datos habituales: nombre del servidor, nombre de la base de datos, contraseña, y dirección de correo electrónico del administrador.

**Nota:** Nos solicitará nombre y contraseña de administrador. Guardémosla en lugar seguro, pues será nuestra clave de entrada para configurar todas las opciones del foro.

Este es todo el proceso de instalación que requiere *phpBB*. Si lo comparamos con *PostNuke* u *osCommerce*, la diferencia es notable, jverdad?

## 11.5. Configuración de *phpBB*

A continuación, se nos mostrará la pantalla del administrador (jimagineban una aplicación sin su correspondiente interfaz de administración?) donde haremos los ajustes iniciales.

phpBB Administration - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

http://127.0.0.1/fi  
rto/index.php

**Bienvenido a phpBB**

Gracias por elegir phpBB como su solución para su foro. Esta pantalla le dará una síntesis de las principales estadísticas de su foro. Puede regresar a esta página clickando en el link de [Indice del Administrador](#) en el panel de la izquierda. Para regresar al índice de su foro, clickee el logo de phpBB también ubicado en el panel izquierdo. Los otros links ubicados a la izquierda de esta pantalla le permitirán controlar todos los aspectos de este foro, cada pantalla tendrá instrucciones de como utilizar las herramientas.

**Estadísticas del Foro**

Estadística	Valor	Estadística	Valor
Cantidad de envíos:	1	Envíos por día:	0.00
Cantidad de tópicos:	1	Tópicos por día:	0.00
Cantidad de usuarios:	3	Usuarios por día:	0.00
Fecha de inicio del Foro:	02 Ene 1997 06:25 pm	Tamaño del directorio de Imágenes:	No está disponible
Tamaño de la Base de Datos:	71.41 KB	Tipo de Compresión Gzip:	OFF

**Quien está Online**

	Login	Última Actualización	Ubicación del Foro	Dirección IP
tigreton	31 Ene 2003 06:08 pm	31 Ene 2003 06:11 pm	Indice del Foro	127.0.0.1

**Figura 11.2.** Panel de administración del foro.

Examinemos una por una todas las opciones que nos brinda nuestro menu de administrador:

- **Admin Index.** Rapida vision de las estadísticas básicas de nuestros foros, tales como numero de temas, de envíos, de visitantes, etc., así como el nombre e IP de todos aquellos que actualmente estan en linea.
- **Forum Index.** Nos lleva a la pantalla principal del foro, aquella que ve cualquier visitante no registrado la primera vez que entra.
- **Preview Forum.** Lo mismo que **Forum Index**, pero conservando el menu de Administrador en la zona izquierda de la pantalla.
- **Management.** Probablemente la opción fundamental de este programa. Nos permite crear nuevos foros y nuevas categorias, así como colocarlas en el orden que necesitemos.
- **Permissions.** Aqui estableceremos los niveles de acceso de cada uno de los foros. Por ejemplo, podemos crear un foro en el que cualquier visitante pueda leer los mensajes del foro, pero solo los usuarios registrados puedan escribir nuevos mensajes y solo nosotros, como administradores, tengamos opción de crear encuestas.

- **Pruning.** Con esta opcion podemos hacer que se eliminan los mensajes antiguos a los que nadie ha respondido. Se establecera el numero de dias que puede permanecer dicho mensaje antes de ser automaticamente eliminado.
- **Backup Database.** Utilisima opcion que deberemos emplear con frecuencia, y que nos permite realizar copias de seguridad de nuestro foro. Incluso, nos da opcion de comprimirlo para ahorrar espacio.
- **Configuration.** Nos permite configurar numerosas opciones del foro, entre las que podriamos destacar el numero de mensajes por pagina, el nombre de nuestro sitio, el numero de segundos que un usuario debe esperar entre mensaje y mensaje, la descripcion de nuestro sitio, las opciones de las *cookies* y las del correo electronico.
- **Mass Mailing.** Es de suma utilidad enviar de vez en cuando un correo electronico a todas las personas registradas en nuestro foro para mantenerlas informadas de las novedades en nuestra pagina. Esta opcion hara todo el trabajo por nosotros.
- **Restore Database.** En caso de una perdida accidental de datos, con esta opcion podremos recuperar la information que previsoramente teniamos guardada gracias a la opcion **Backup Database**.
- **Smilies.** Si queremos darle un toque de originalidad a nuestra pagina, podemos incluir una serie de "emoticones" que, de seguro, haran las delicias de nuestros visitantes.
- **Word Censor.** Gracias a esta utilidad podremos introducir una lista de todas aquellas palabras malsanas que no queremos que aparezcan en el foro, ademas de las palabra por las que se sustituiran automaticamente.
- **Group Admin Management.** Desde aqui podemos crear diversos grupos en nuestro foro. Los pertenecientes a un determinado grupo podran tener sus foros propios, a los que solo podran acceder ellos.
- **Group Admin Permissions.** Para gestionar los permisos que otorgamos a cada miembro de un grupo especifico, asi como a su administrador.
- **Styles Admin.** Solo para usuarios avanzados que tengan experiencia en HTML y CSS. Nos permite gestionar los estilos de nuestro foro, tales como fuentes, colores, encabezamientos, etc.

- **Ban Control.** Es típico de los foros encontrarse con usuarios molestos. Con esta opción podremos impedir a esos usuarios molestos acceder a nuestro foro, e incluso bloquear el acceso al foro desde una IP determinada.

## 11.6. Creacion de un foro

Vamos a estudiar el funcionamiento práctico de nuestro foro desde el punto de vista del administrador y desde el punto de vista del usuario que entra por primera vez y quiere dejar un mensaje en él.

El primer paso del administrador es entrar en **Forum Admin Management** y crear una nueva categoría a la que llamaremos **Colegio Maravillas** y, dentro de esta categoría, un foro llamado **Antiguos Alumnos promoción 1977**.

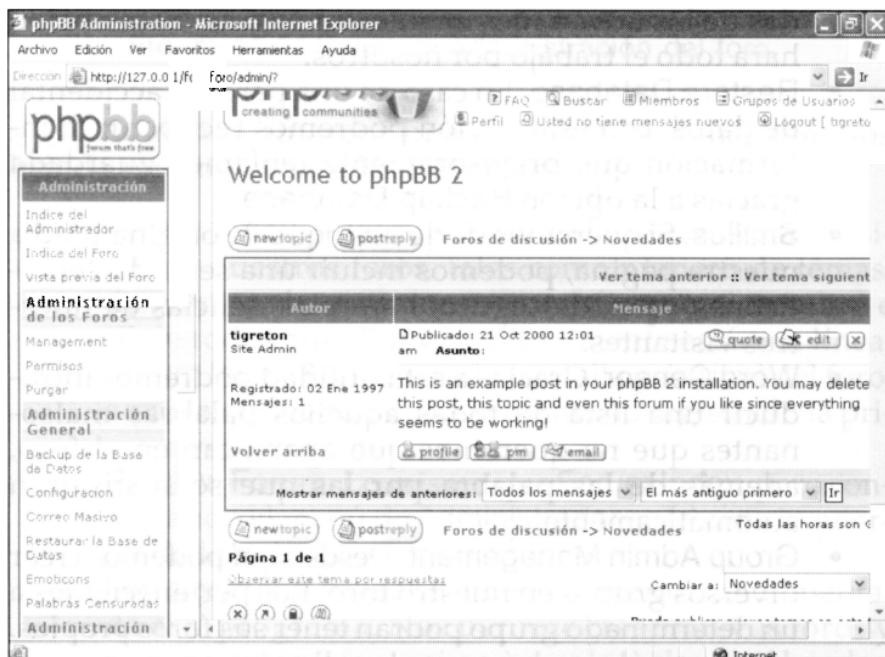


Figura 11.3. Creacion de un foro.

Nos pedirá algunos datos, como la descripción del foro, y cuánto tiempo dura cada mensaje no respondido.

Asimismo, dado que Ricardo tenía en el colegio el mote de "orejotas", quiere evitar a toda costa que dicha palabra aparezca en el foro. Para ello, nos dirigimos a la sección

**Word** Censors, e introducimos la palabra "orejotas", y aquella que la sustituirá, algo así como "#####".

A continuación, pulsamos en **New Topic**, y creamos nuestro primer mensaje: "Bienvenidos al foro de antiguos alumnos del Colegio Maravillas".

Ahora a Ricardo solo le queda sentarse y esperar a que los antiguos alumnos empiecen a dejar sus mensajes.

Mientras tanto, Pedro Vazquez, antiguo alumno del Colegio Maravillas ha oido hablar de la nueva pagina, y decide entrar en el foro y dejar un mensaje de saludo.

Su primer paso será registrarse en el foro, para lo que entrara en la opcion de **Register**. Tras aceptar las condiciones, deberá introducir datos tales como su nombre, su dirección de correo electrónico, contraseña, información sobre sus gustos y preferencias, así como todo tipo de opciones visuales.

En breves instantes Pedro Vazquez recibira un mensaje en su dirección de correo dandole la bienvenida al foro y recordandole su nombre de usuario y su contraseña.

Entonces, a Pedro no le quedara mas que introducir estos datos en la opcion **Login**, con lo que tendra libertad para dejar mensajes en el foro.

## 11.7. Conclusion

Aunque aprender a programar un foro con PHP y MySQL es una experiencia sumamente didáctica que, con los conocimientos adquiridos con esta guia, no debería serle demasiado complicado, a la hora de incorporar un foro a nuestra pagina Web, es aconsejable recurrir a una solución mas profesional como es *phpBB* que, además, es gratuita.

Aunque en este capítulo hemos visto su funcionamiento básico, hay miles de usuarios deseosos de brindarnos su apoyo y colaboración cuando deseemos profundizar más en su funcionamiento.

---

*Nota: Difícil será encontrar alguna duda que no pueda ser resuelta en <http://www.phpbb.com/phpBB>.*

---

## 12.1. Que es una *cookie*

Desde un aspecto teórico, podríamos decir que una *cookie* es una pequeña pieza de información, enviada desde el servidor, que aloja la página y que se aloja en el ordenador del cliente, aunque la conexión con dicha página haya finalizado. Dicha información es enviada vía HTTP, y es lo primero que busca la página cuando volvemos a entrar en ella.

Veamoslo más claramente con un ejemplo. Si alguna vez hemos hecho una compra, o al menos nos hemos registrado en una tienda tipo Amazon, y volvemos a entrar al cabo de unos días, encontramos un saludo personalizado en el que se nos llama por nuestro nombre y se nos recuerdan nuestras últimas compras, sin necesidad de que introduzcamos ningún dato.

Esto es debido a que en nuestra pasada visita se envió una *cookie* a nuestro ordenador con dichos datos, datos que han sido recuperados por Amazon en el momento en que entramos en la página.

A primera vista, una *cookie* es un elemento realmente útil que nos permite ahorrarnos trabajos repetitivos, y permite una interacción más rápida entre empresas y clientes.

Sin embargo, como usted bien sabrá, las *cookies* no tienen buena fama, y enseguida podremos comprobar por qué.

Hagamos un pequeño experimento. Abra su navegador habitual, entre en la opción **Herramientas** y, a continuación, en **Privacidad**. En esta pantalla pulse en **Editar**, y nos aparecerá un listado de todas las *cookies* que tenemos almacenadas en nuestro ordenador.

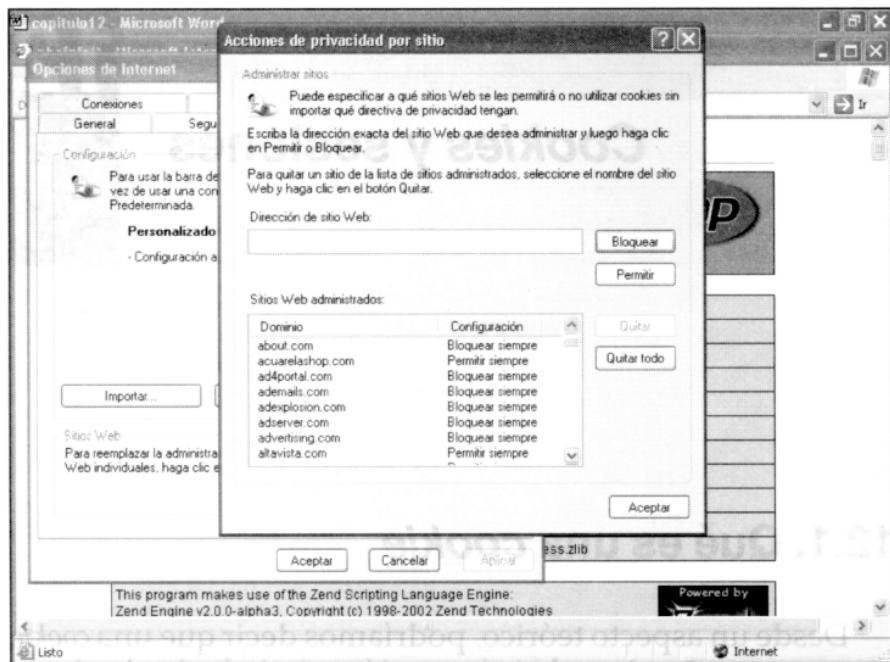


Figura 12.1. Cookies almacenadas en el navegador.

Asusta, ¿verdad? En esa lista encontrara *cookies* de los sitios Web que usted suele frecuentar, pero de la inmensa mayoría no sabrá ni de quien son, ni como se han colado en su ordenador. Y estas *cookies* están enviando información sobre sus usos de Internet a destinatarios no autorizados.

Este es el principal problema de las *cookies*. Se introducen en nuestro ordenador sin nuestro permiso y se hacen con datos privados. ¿Cómo puede esto suceder?

En general, empresas deshonestas aprovechan nuestra entrada a páginas de información general para instalar *cookies* específicas de su negocio (con la connivencia y el beneficio de la empresa que aloja la página). Por ello, muchos internautas han optado por deshabilitar la opción de recepción de *cookies* en sus servidores.

Sin embargo, a modo didáctico nos será útil conocer como funcionan las *cookies*, y como crearlas nosotros mismos e instalarlas en los ordenadores que visiten nuestra página.

## 12.2. Funcionamiento de las *cookies*

Veamos con mas detenimiento el funcionamiento de una *cookie* generada mediante PHP.

En primer lugar, el buscador utiliza el URL que nosotros le proporcionamos y que, en este caso, corresponde a una pagina escrita en PHP (como, por ejemplo, la pagina Web del colegio Maravillas). Dentro de esa pagina hemos introducido la funcion `setCookie()`, incluida entre las sentencias PHP para estos menesteres.

Dicha pagina es enviada por el servidor que la aloja, en cuyo encabezamiento se encuentra una llamada a la funcion `setCookie()` con el nombre de la *cookie* (por ejemplo, maravillas) y los valores de dicha *cookie*.

Al recibir esta informacion, nuestro buscador creara una variable con el nombre de la *cookie* (en este caso, `$maravillas`), y la almacenara junto con los valores que esta incorpora.

En lo sucesivo, cada vez que entremos en dicha pagina, nuestro navegador enviara dicha variable, cuyos valores seran interpretados por la pagina de origen, que actuara en consecuencia.

## 12.3. Como crear nuestras propias *cookies*

Hasta ahora hemos estudiado las *cookies* desde el punto de vista del visitante. Ahora aprenderemos como crearlas nosotros mismos e incorporarlas a nuestras propias paginas.

La estructura en PHP para crearla es la siguiente:

```
setCookie(nombre, [valor [, fecha_expiración,
[, ruta [,dominio [,seguridad 11111]]]]])
```

Veamos uno a uno los diferentes parametros que conforman una *cookie*:

- **Nombre.** Obviamente, el nombre que se le asigna a esta *cookie*. Es el unico valor realmente obligatorio, aunque hay que tener en cuenta que crear la funcion `setCookie()` con solo el valor del nombre lo unico que hara sera borrar la *cookie* que lleve ese nombre.
- **Valor.** Son los datos que almacenamos en el navegador del visitante.
- **Fecha de expiracion.** El numero de segundos que han de pasar antes de que la *cookie* sea automaticamente borrada. Si, por ejemplo, queremos que la *cookie* expire en una hora (3600 segundos), tendriamos que establecer la fecha de expiracion como `time() + 3600`.

- **Ruta.** Para el caso de que queramos restringir el acceso de la cookie a una determinada ruta de nuestro servidor (por ejemplo, /colegio). Funcion util solo en el caso de que compartamos el servidor con otros usuarios y cada uno tenga sus carpetas propias.
- **Dominio.** Similar al anterior, sirve para restringir el acceso de la cookie a un dominio determinado. Por defecto, la cookie solo es enviada al dominio original que la creó. En el caso de que trabajemos con varios dominios y queramos que la cookie sea enviada a cualquiera de ellos, deberemos indicarlo en este campo.
- **Seguridad.** Como bien indica su nombre, en este parametro indicamos el grado de seguridad de envío de la cookie. Tiene dos valores, 1 y 0. El valor 1 indica que la cookie solo sera mandada via servidor seguro (SSL) , y el 0, que podra ser mandada con cualquier conexión.

*Nota: Como norma general, debemos recordar que la función **setCookie()** debe estar siempre al principio de cada script, o de otro modo no funcionará.*

Veamos ahora con un sencillo ejemplo cómo funciona una cookie:

```
<!-- cookiecontadora.php -->
<?php
$visitas = $visitas + 1;
setCookie("visitas",$visitas,time() + 3600*24*365);
?>
<html>
<head>
<title> Título </title>
</head>
<body>
    <?php
        if ($visitas > 1) {
            echo("Esta es tu visita número $visitas.");
        } else {
            echo("¡Bienvenido por primera vez a la
            página Web del colegio Maravillas!");
        }
    ?>
</body>
</html>
```



**Figura 12.2.** Pagina que crea cookies.

La primera vez que entremos en esta pagina, nos aparecerá el saludo “¡Bienvenido por primera vez a la pagina Web del colegio Maravillas!”. Y en lo sucesivo, lleva la cuenta de todas las veces que entramos en la pagina, y nos lo indicara en pantalla.

Analicemos línea por línea el *script*:

```
$visitas = $visitas + 1 :
```

Definimos la variable *visitas* e indicamos que, cada vez que abramos el *script*, le sumemos una unidad a la variable.

```
setCookie("visitas",$visitas,time()+3600*24*365);
```

Aquí utilizamos la función *setCookie()* que hemos visto en las páginas anteriores, y definimos tres parámetros. El nombre de la *cookie* será “visitas”; su valor, el de la variable *\$visitas*, que hemos definido anteriormente; y su duración, un *ado* (desde el momento actual, por los 3600 segundos que tiene una hora, por las 24 horas que tiene un día, por los 365 días del año). Pasado este *ado*, la *cookie* desaparecerá del navegador de los visitantes de nuestra página.

```
if ($visitas > 1) {  
echo("Esta es tu visita número $visitas.");  
} else {  
echo("¡Bienvenido por primera vez a la pagina  
Web del colegio Maravillas!");  
}
```

Creamos un bucle **if/else** para determinar que respuesta daremos a nuestros visitantes. Si el valor de la variable **\$visitas** es mayor que uno, suponemos que es porque ya ha visitado nuestra pagina con anterioridad, y nos limitamos a imprimir en pantalla el valor de **\$visitas**, es decir, el valor que nos es facilitado por la cookie llamada Visitas.

En caso contrario, debemos suponer que es la primera vez que entra en nuestro site, asi que le mostramos un mensaje de bienvenida.

## 12.4. Conclusion

Pese a su evidente utilidad, las cookies presentan numerosos problemas. Esencialmente, la mala fama que arrastran por su poca privacidad hace que muchos usuarios las hayan deshabilitado de sus buscadores, por lo que las hacen inutiles en muchos ordenadores.

Y aunque no hayan sido deshabilitadas, hay que tener en cuenta que los buscadores admiten un numero limitado de cookies, alrededor de 20, y a partir de ahí, cada vez que admiten una nueva, borran una antigua, por lo que nunca podremos estar seguros de si la *cookie* seguira vigente o no. Además, una cookie no puede aceptar una cantidad de informacion importante. Ello hace que cada vez sean menos usadas.

Afortunadamente, existen soluciones mas efectivas a las cookies, que veremos en el siguiente epigrafe.

## 12.5. Que son las sesiones

Las sesiones son la mejor forma de almacenar datos e informacion de los visitantes de nuestra pagina, y conservarlos a traves de toda la visita a nuestro sitio Web.

Volvamos a poner el ejemplo de una pagina tipo Amazon. Cuando incluimos un articulo en nuestro carrito

de compra (por ejemplo, **Guia practica: Desarrollo Web con PHP y MySQL**), esta informacion queda almacenada de tal manera que podemos seguir navegando por las diversas paginas del catálogo sin que esa informacion desaparezca. Todo ello se hace a traves de sesiones, que presentan una serie de ventajas decisivas sobre las cookies:

- Un usuario puede deshabilitar las *cookies* de su navegador, pero no las sesiones. Podemos tener la seguridad de que las sesiones siempre funcionaran.
- Mientras que las *cookies* se almacenan en el servidor de nuestro visitante, la informacion de las sesiones se almacena en nuestro propio servidor. Ello permite que la informacion almacenada sea mucho mayor, y nuestro acceso a ella mucho mas facil.

**Nota:** Para que las sesiones funcionen correctamente, todas las paginas de nuestro site deben tener la **extension .php**. De otro modo, la informacion almacenada en las sesiones se perderia al abandonar la pagina php.

El funcionamiento de una sesion en PHP es el siguiente: Cuando el visitante entra en nuestra pagina, se le asigna un numero unico de sesion, popularmente conocido como ID, aleatorio y protegido criptograficamente. Generado por PHP, es almacenado en nuestro servidor junto con la informacion asociada a dicha sesion, y mantenido mientras el visitante se mueve a traves de las paginas de nuestro site.

## 12.6. Como crear las sesiones

El primer paso es, lógicamente, iniciar la sesion. La sentencia que utilizaremos es:

### 12.6.1. **session\_start()**

Esta función, en primer lugar, comprobará si existe una sesión actualmente abierta, y si no, abrirá una. Esto es importante, pues nos evitara tener varias sesiones abiertas simultaneamente. En caso de que ya exista una sesion abierta, se limitara a abrir toda la informacion asociada a dicha sesion activa.

---

**Nota:** Dada esta característica de las sesiones, es una buena idea incluir la sentencia **session\_start()** al principio de todos los scripts de nuestro sitio que incluyan sesiones.

---

El siguiente paso para conseguir que la información sea transmitida sin problemas de una página a otra sería registrar una variable mediante la sentencia:

### 12.6.2. **session\_register()**

Veámoslo con un ejemplo. Supongamos que mediante un formulario hemos obtenido el nombre del visitante, y queremos transmitir esta información de página en página.

```
$nombre= "Pedro Martín";  
session_register("nombre");
```

---

**Nota:** Tenga en cuenta que no es necesario incluir el símbolo \$ a la hora de registrar la variable.

---

De esta manera, la variable será enviada a través de las páginas hasta que la sesión termine o nosotros manualmente "desregistremos" la variable.

Se puede registrar más de una variable simplemente separándolas por comas. Por ejemplo, `session_register("nombre", "apellidol", "apellido2");`

Otro aspecto a tener en cuenta es que una variable registrada no puede ser enviada mediante las sentencias GET o POST, lo cual, en determinadas ocasiones, puede llevarnos a confusión.

Para saber si una variable está registrada, podemos utilizar la sentencia:

### 12.6.3. **session\_is\_registered();**

Si, por ejemplo, quisieramos saber si la variable \$nombre es una variable de sesión ya registrada, haríamos:

```
$registro=session_is_registered("nombre");
```

Esta variable devolverá verdadero o falso (true o false), dependiendo de que la variable esté registrada o no.

## 12.7. Cerrar las sesiones

El primer paso para cerrar una sesión, es "desregistrar" todas las variables. Para ello, empleamos la sentencia `session_unregister()`. Su funcionamiento es idéntico a la función `session_register()`, con la salvedad de que solo podemos quitar el registro de una variable cada vez.

Si lo que deseamos es quitar el registro de todas las variables de una sola vez, la función a emplear será `session_unset()`.

Una vez realizada esta tarea, solo nos queda terminar la sesión, cosa que faremos con facilidad con la siguiente sentencia: `session_destroy()`.

## 12.8. sesión.php

Ahora repasaremos todos estos conceptos con un ejemplo práctico. Crearemos una sesión y registraremos una variable que pasaremos a través de tres páginas distintas.

Veamos los diferentes *scripts*:

```
<!-- sesión1.php -->
<?
session_start();
$nombre= "Pedro";
session_register("nombre");
echo "Bienvenido a mi pagina, $nombre <br>";
?>
<a href ="sesión2.php">Siguiente página</a>

<!-- sesión2.php -->
<?
session_start();
echo "hola de nuevo, amigo $nombre<br>";
session_unregister("nombre");
?>
<a href = "sesión3.php">Siguiente página</a>

<!-- sesión3.php -->
<?
session_start();
echo "Hola de nuevo, $nombre";
session_destroy();
?>
```



**Figura 12.3. Ejemplo de sesiones.**

Estudiemos los diferentes *scripts* uno a uno. En **session1.php** creamos la sesión; con la sentencia **session-start()**, creamos la variable **\$nombre**, le asignamos un valor (en este caso, *Pedro*), y la registramos con la sentencia **session-register()**, para que su información sea mandada a través de las diferentes páginas.

A través del vínculo que hemos incluido en el *script*, pasamos a la siguiente página, llamada **sesion2.php**, en la que comprobamos que el valor de la variable **\$nombre** se ha traspasado sin ningún problema a través de la sesión. El siguiente paso es "desregistrar" esa variable mediante la sentencia **session\_unregister()**.

Pasamos a la siguiente página, **sesion3.php**, y vemos que aunque la sesión sigue vigente, el valor de la variable no ha pasado a esta página, puesto que la "desregistramos" en el *script* anterior. El siguiente paso es terminar la sesión con la sentencia **session-destroy()**.

## 12.9. duracion.php

```
<!-- duracion.php -->
<?
session-start();
```

```

if ( !session-is-registered("cuenta"))
{
    session-register("cuenta");
    session-register("empezar");
    $cuenta=0;
    $empezar=time();
}
else
{
    $cuenta++;
}
$sessionID= session-id();
?>

<html>
<head>
<title>Contador de Sesiones</title>
</head>
<body>
    <p> Esta pagina incluye la sesión
    (<?=$sessionID?>)
    <br>cuenta = <?=$cuenta?>.
    <p>Esta sesión ha durado
    <?php
        $duracion = time() - $empezar;
        echo "$duracion ";
    ?>
    segundos .
</body>
</html>

```

## 12.10. Autentificacion de usuarios mediante sesiones

Ahora utilizaremos nuestras recien aprendidas nociones de sesiones para una aplicacion eminentemente práctica, que se puede aplicar a la pagina Web que nos esta sirviendo de ejemplo a lo largo del libro, el sitio Web de antiguos alumnos del colegio Maravillas.

En un momento dado, podríamos plantearnos crear determinadas secciones del *site* de acceso exclusivo para usuarios registrados. Es decir, solo aquellos que han introducido sus datos en nuestra base de datos podrán acceder.

Lo habitual en muchos sitios es pedir una contraseña, pero para facilitar la tarea a nuestros usuarios (el tema de incluir contraseñas en nuestras páginas tiene un gran peligro: ¿a cuantas páginas ha dejado usted de entrar por no recordar la contraseña?), simplemente, les pediremos su apellido y su dirección de correo electrónico. En caso de que esos datos coincidan con los que tenemos en la base de datos, se le permitirá entrar en la zona exclusiva. Si no coinciden, le será imposible acceder.

Nuestro sistema de autenticación estará formado por tres *scripts*:

```
<!-- home.php -->
<?
session_start();
if ($apellido && $email)
{
$db_conn = mysql_connect('localhost', 'ricardo',
'maravillas');
mysql_select_db("alumnos", $db_conn);
$query = "select * from alumnos "
."where apellido='".$apellido' "
." and email='".$email."'";
$result = mysql_query($query, $db_conn);
if (mysql_num_rows($result) >0 )
{
$valid-user = $apellido;
session_register("valid-user");
}
}
?>
<html>
<body>
    <h1>Página de libre acceso</h1>
    <?
    if (session_is_registered("valid_user"))
    {
        echo "Estás registrado, y tu apellido es:
$valid-user <br>";
        echo "<a href=\"desregistro.php\">Salir</
a><br>";
    }
    else
    {
```

```

if (isset($apellido))
{
echo "No te has registrado correctamente";
}
else
{
echo "No estas registrado.<br>";
}
echo "<form method=post action=
\"authmain.php\">";
echo "<table>";
echo "<tr><td>apellido:</td>";
echo "<td><input type=text name=apellido></
td></tr>";
echo "<tr><td>email:</td>";
echo "<td><input type=password name=
email></td></tr>";
echo "<tr><td colspan=2 align=center>";
echo "<input type=submit value=
\"Log in\"></td></tr>";
echo "</table></form>";
}
?>
<br>
ca href="solomiembros.php">Sección
exclusiva para miembros registrados</a>
c/body>
</html>

```

```

<!-- desregistro.php -->
<?
session-start();
$old-user = $valid-user;
$result = session_unregister("valid_user");
session-destroy();
?>
<html>
<body>
    <h1>Log out</h1>
    <?
    if (!empty($old_user))
    {
        if ($result)
    {

```

```

echo "Salir del registro.<br>";
}
else
{
echo "No ha podido salir del registro.<br>";
}
}
else
{
echo "Dado que no se había registrado,
no puede salir del
registro.
<br>";
}
?>
<a href="home.php">Vuelta a la pagina
principal</a>
</body>
</html>

<!-- solomiembros.php -->
<?
session_start();
echo "<h1>Sección exclusiva miembros
registrados</h1>";
if (session_is_registered("valid-user"))
{
echo "<p>Estás registrado y tu apellido es
$valid_user.</p>";
echo "<p>Contenidos exclusivos para vosotros</
p>";
}
else
{
echo "<p>No esta registrado.</p>";
echo "<p>Solo miembros registrados pueden
acceder a esta página</p>";
}
echo "<a href=\"home.php\">Vuelta a la pagina
principal</a>";
?>

```

El primero de ellos, `home.php`, es aquel que crea la sesión y establece la conexión con la base de datos, de donde recupera los valores del apellido y del correo electrónico.

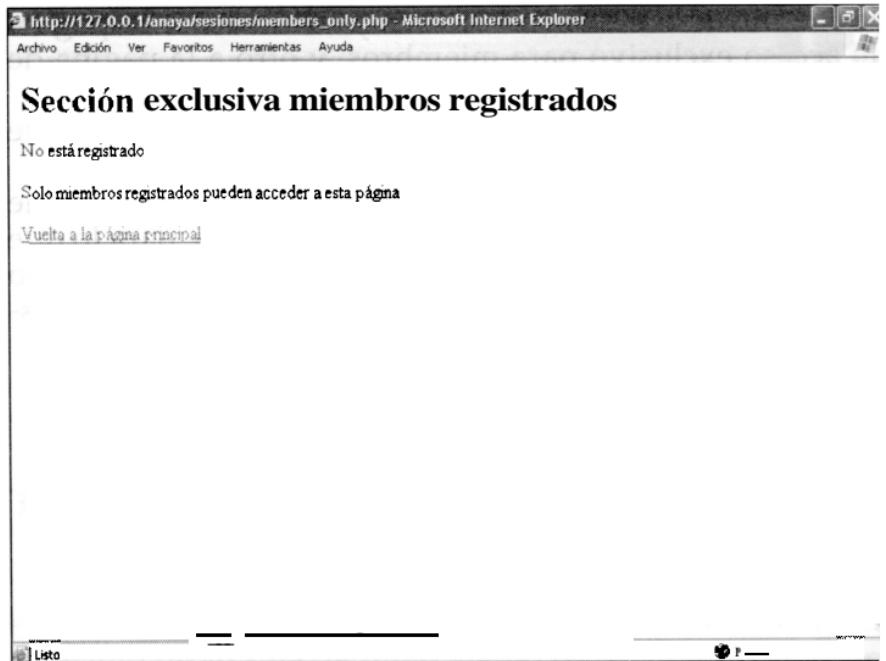


Figura 12.4. Autentificación con sesiones.



Figura 12.5. Pagina de registro mediante sesiones.

Asimismo, validamos la variable de sesión "**valid-user**", que sera la que pasaremos de una página a otra. Finalmente, creamos un pequeño formulario donde el visitante

puede introducir sus datos. Incluye vinculos a la pagina de acceso exclusivo para miembros, y otro a la pagina de quitar registro.

El siguiente *script*, **desregistro.php**, es aquel en el que cerramos la sesion mediante **session\_destroy()**.

El ultimo de ellos, **solomiembros.php**, es aquel al que solo pueden acceder los miembros registrados, para lo que analizamos la variable “**valid-user**”, para ver si el usuario esta correctamente registrado. En caso afirmativo, nos mostrara los contenidos exclusivos, y si no, se lo indicamos.

# **eMule: Solución Open Source de intercambio de archivos**

## **13.1. Introducción**

He aqui otra aplicacion realizada en PHP que esta causando furor entre los internautas. Se trata de *eMule*, un utilissimo programa de intercambio de archivos con el resto de usuarios conectados a Internet.

Dado que para su buen conocimiento es fundamental unas nociones básicas del concepto de sesiones, hemos incluido este capitulo a continuación del 12, que es el que desarrolla el tema de *cookies* y sesiones en PHP, por lo que recomendamos su atenta lectura antes de pasar a este capitulo.

Seguramente, en alguna ocasion habrá escuchado hablar, ya sea en prensa, radio o television, del fenomeno *Napster*. Antes que problemas legales y financieros lo llevasen al borde de su desaparicion, llego a contar con ¡millones! de usuarios fielmente conectados dia tras dia.

¿Qué ofrecía *Napster* para lograr tal record de fidelidad? Realmente nada. *Napster* no era mas que una herramienta de intercambio de archivos mediante la cual los internautas podian intercambiarse unos con otros diversos archivos almacenados en su disco duro. Aunque se dio a conocer fundamentalmente por los intercambios de música en MP3, *Napster* permitia el intercambio de todo tipo de archivos: *software*, *ebooks*, películas, fotos, textos, etc.

Un servidor central se encargaba de gestionar las peticiones de descargas mediante un buscador, que nos ponía en contacto con aquel usuario en cuyo disco duro existia el archivo que nosotros buscábamos, y se iniciaba la descarga.

Las compañías discograficas pusieron en marcha una durisima ofensiva legal y publicitaria contra *Napster*, que llevo a su intervención judicial y posterior cierre.

Sin embargo, diversas aplicaciones se propusieron recoger el testigo dejado por *Napster*: nombres como *winmx*, *Kazaa*, *Morpheus* o *eDonkey* pronto se hicieron conocidos entre todos aquellos antiguos "napsterianos" que habian quedado huérfanos.

Sin embargo, en el presente capitulo vamos a referirnos a *eMule*, programa de intercambio de archivos realizado en PHP bajo licencia GPL, y que parece haberse convertido en la opcion favorita de los internautas.

## 13.2. ¿Qué es *eMule*?

Como hemos visto, el punto fuerte de *eMule* no es su originalidad. No es mas que un programa que hace lo mismo que otros muchos, aunque lo hace mejor y mas rapido, al estar basado en el protocolo MFTP (*Multisource File Transfer Protocol*, Protocolo de Transferencia de Archives Multifuente).

Realmente, *eMule* nacio como una ampliacion del programa *eDonkey2000*, que no estaba programado en codigo abierto y presentaba numerosos fallos y limitaciones que impedian su desarrollo.

*eMule*, al ser un programa *Open Source*, permite que haya una comunidad de programadores trabajando desinteresadamente en su desarrollo, y el resto de usuarios podamos beneficiarnos de las continuas actualizaciones que van surgiendo.

A diferencia de *Napster*, *eMule* no cuenta con un unico servidor, sino que podemos elegir entre diversos servidores a la hora de conectarnos. Ello reduce espectacularmente los tiempos de espera, y nos da la opcion de entrar en los servidores menos saturados en cada momento.

Asimismo, incluye la opcion de descargar el mismo archivo desde diversas fuentes, lo cual multiplica las velocidades de descarga. Tambien permite recuperar descargas interrumpidas. Y en caso de que nuestra conexión a Internet se haga a traves de un "cortafuegos", podemos elegir el puerto de descargas.

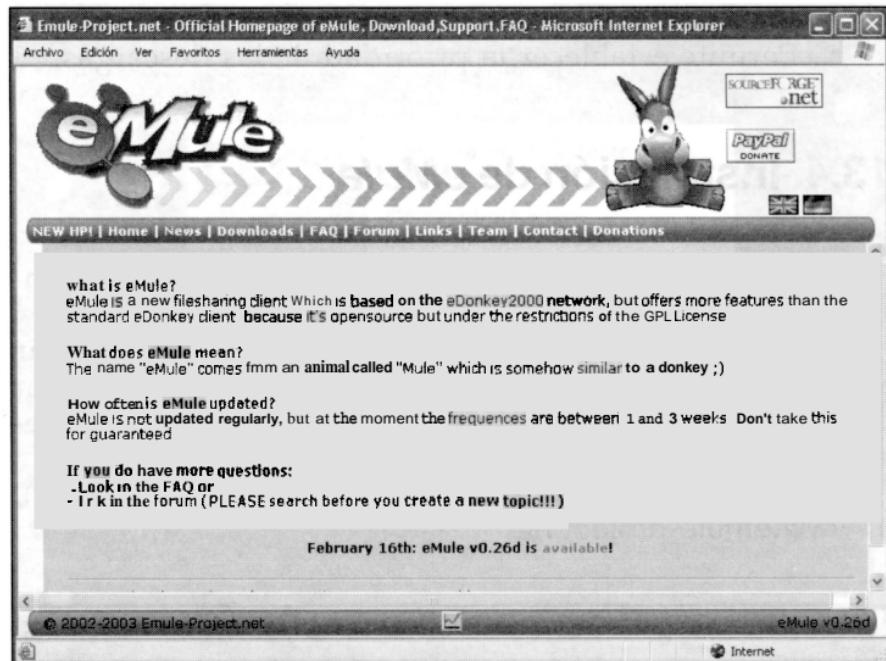


Figura 13.1. Logotipo de eMule.

### 13.3. Características fundamentales de eMule

Entre las características fundamentales de este programa podemos destacar:

- Permite compartir cualquier tipo de archivo (audio, archivos de programas, video, imagen de CD, etc.).
- Posee una red propia de servidores descentralizados completamente creciente.
- Busca entre todos los usuarios de la red, sin importar en qué servidor están conectados.
- Las descargas son retomadas automáticamente al comienzo de cada sesión.
- Permite una actualización dinámica de los servidores cada vez que nos conectamos.
- Incluye un sistema de créditos que prima a los que más archivos comparten y penaliza a los usuarios que se limitan a descargar, sin compartir ningún archivo.
- Efectúa compresión de los archivos antes de la descarga, lo que redundará en una mayor velocidad de la misma.

- Incluye conexión directa con el *Chat IRC*.
- Permite establecer la prioridad de las descargas.

## 13.4. Instalacion de eMule

La instalacion de *eMule* es un proceso sumamente sencillo, si seguimos con detenimiento las instrucciones.

El primer paso es obtener el código fuente. Pese a venir en el CD adjunto a la presente guía, es aconsejable buscar directamente en Internet, pues suele salir una versión nueva cada tres o cuatro semanas.

La página oficial de *eMule* (en inglés, naturalmente), es [www.emule-project.net](http://www.emule-project.net)

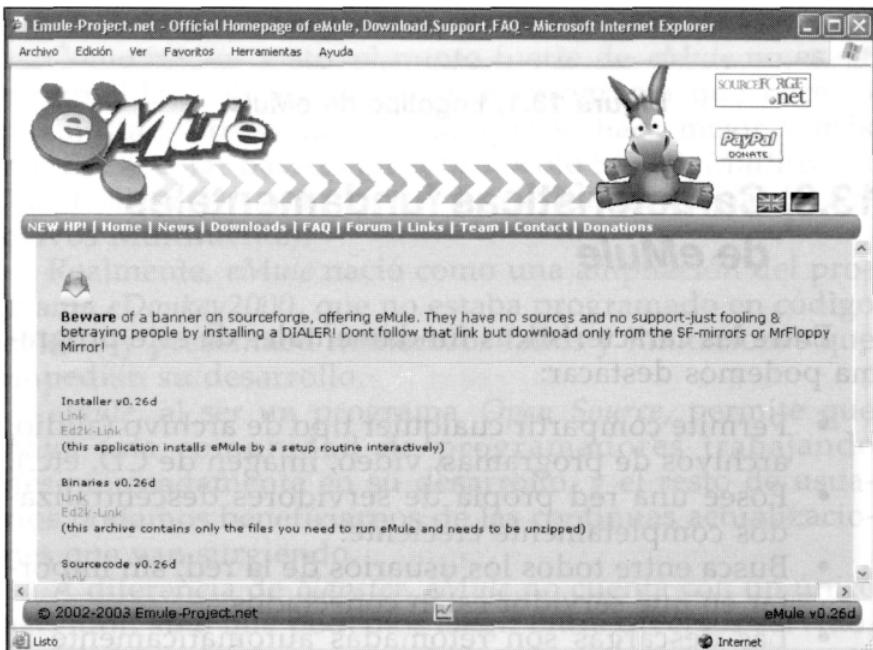


Figura 13.2. Página oficial de *eMule*.

Una vez en nuestro disco duro, simplemente pulsamos sobre el archivo, y elegimos el idioma de instalación.

Tras aceptar los términos de la licencia y el directorio de instalación, en apenas unos segundos, la instalación se habrá completado.

Más fácil, imposible, ¿verdad?

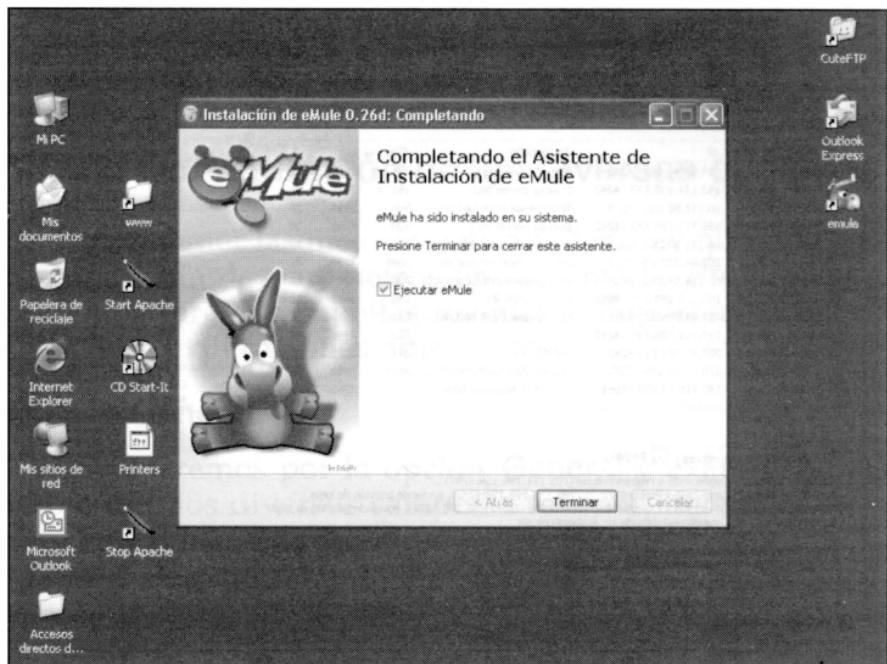


Figura 13.3. El programa eMule ha sido instalado

## 13.5. Configuración e instalacion de eMule

A continuación analizaremos una por una las opciones de *eMule* para una descarga optima de archivos:

### 13.5.1. Conexión a un servidor

Lógicamente, el primer paso sera conectar con alguno de los servidores. Existen varias páginas que nos informan, en tiempo real, de los mejores servidores para conectarnos. Entre ellas, yo, personalmente, suelo utilizar [www.servermet.has.it](http://www.servermet.has.it)

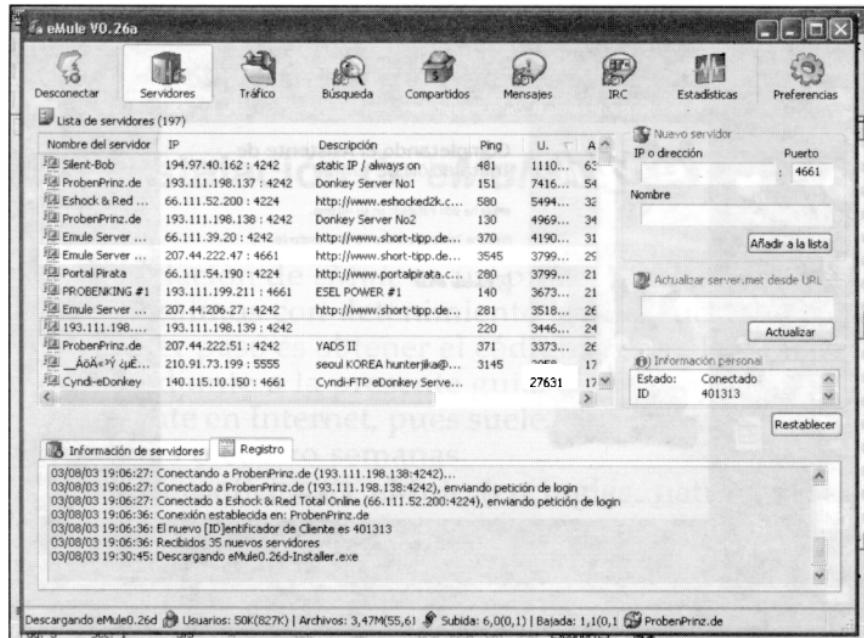
Otros vinculos que suelen dar buena información son:

<http://ocbmaurice.dyns.net/pl/slist.pl?download/server-max.met>

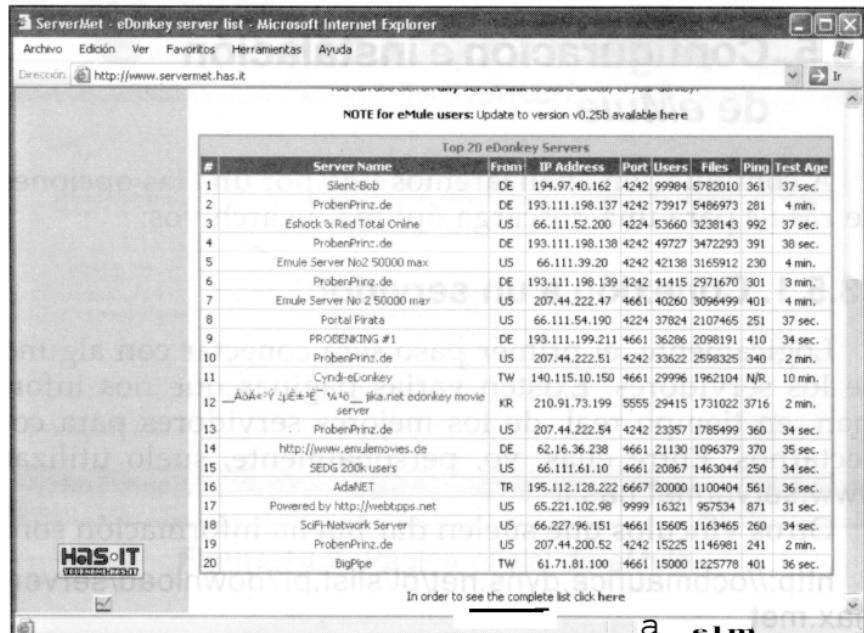
<http://2z4u.de/4eom6pql/min/server.met>

Una vez en dicha pagina, debemos situar el cursor sobre la palabra **Here**, pulsar en el botón derecho del ratón y elegir la opción **Propiedades**. Nos aparecerá el siguiente vínculo:

<http://usuarios.lycos.es/guillesspi/server.met>



**Figura 13.4.** Lista de servidores para eMule.



**Figura 13.5.** Una de las mejores páginas de información sobre eMule.

Debemos copiar dicho vínculo y pegarlo en la interfaz de *eMule*, en la pantalla de servidores, en el espacio titulado

Actualizar server.met desde URL. Tras pulsar el botón **Actualizar**, se descargara la lista de servidores, y solo tendremos que optar por el que mas nos interese.

### 13.5.2. Configuración de las diversas opciones

Para obtener las mejores velocidades, hay que configurar una serie de parametros en la pantalla Preferencias. Sólo mostraremos aquellas realmente importantes, o que presenten explicaciones adicionales.

#### Opción General

Empezaremos por la opcion **General** y examinaremos uno a uno los diversos campos a rellenar, deteniéndonos sólo en los mas importantes.

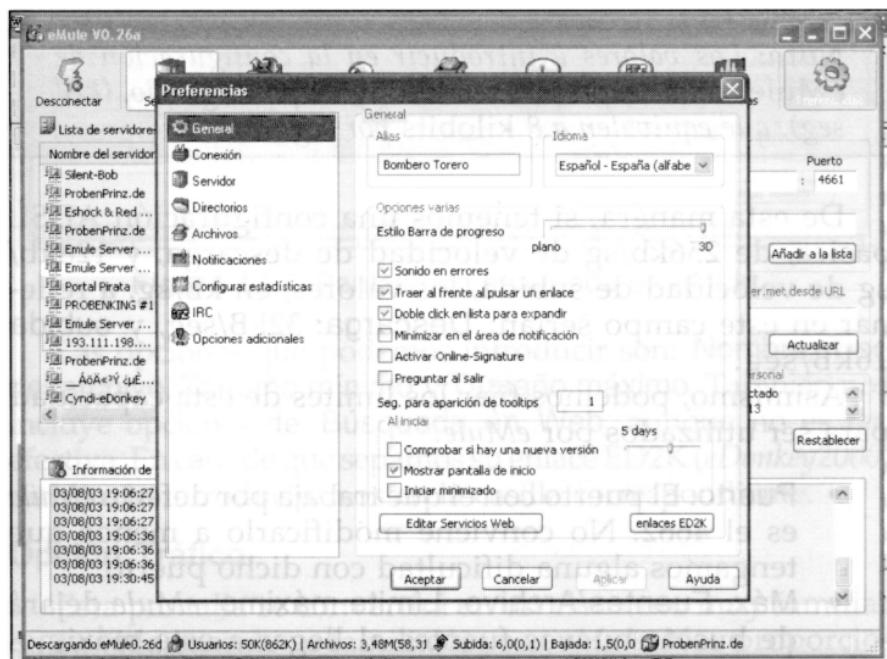


Figura 13.6. Configuración general de eMule.

- **Alias.** Aquí introduciremos el nombre por el que nos reconoceran el resto de usuarios.
- **Idioma.** Por defecto, el que hayamos seleccionado durante la instalacion.
- **Estilo Barra de Progreso.** Una mera opcion estética que no influye en la velocidad de descarga.

- **Sonido en errores.** Emitira un pitido cuando detecte un error o algun archivo corrupto.
- **Traer al frente al pulsar un enlace.** Si esta activada, al hacer clic sobre una pagina de enlaces, la ventana del *eMule* se activara y se abrira sobre el fondo del escritorio.
- **Segundos para aparicion de tooltips.** Marcando esta opción con un numero (Ej. 1), si en la pantalla de descargas pasamos el ratón sobre alguno de los archives, nos mostrara la información sobre ese archivo al cabo de un segundo.

## Opción Conexión

Aquí es donde especificaremos las características de nuestra conexión a Internet.

---

***Nota:*** Los valores a introducir en la configuración de *eMule* han de expresarse en kilobytes por segundo, (kB/seg), que equivalen a 8 kilobits por segundo.

---

De esta manera, si tenemos una configuración ADSL básica, de 256kb/sg de velocidad de descarga, y 128kb/sg de velocidad de subida, los valores, en kB/sg, a rellenar en este campo serían: Descarga: 32kB/seg, y subida 16kB/seg.

Asimismo, podemos fijar los límites de esta capacidad para ser utilizados por *eMule*.

- **Puerto.** El puerto con el que trabaja por defecto *eMule* es el 4662. No conviene modificarlo a menos que tengamos alguna dificultad con dicho puerto.
- **Max. Fuentes/Archivo. Límite maximo.** *eMule* dejará de buscar nuevas fuentes al llegar a este máximo. Conviene no dejarlo superior a 400, o podría resultar contraproducente.
- **Límites de conexión/Conexiones máximas.** El número maximo de conexiones TCP/IP simultaneas que puede tener abiertas el *eMule*. No deben superar las 50.
- **Mostrar ancho de banda excedente.** Nos indicara el grado de saturación de la conexión en relación a su capacidad y a las descargas empleadas.

## Opcion Busqueda

Una vez conectado con el mejor servidor y optimizada la conexión, ahora nos llega la parte divertida. La búsqueda de aquellos archivos que queramos descargar.

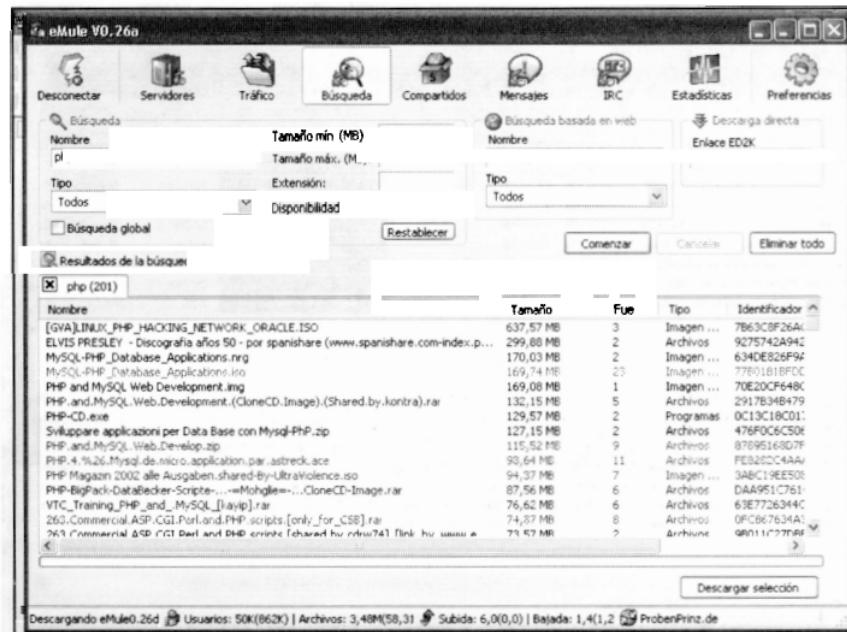


Figura 13.7. Pantalla de búsqueda en eMule.

Las opciones que podemos introducir son: **Nombre**, **Tipo** del archivo, **Tamaño mínimo** y **Tamaño maximo**. Tambien nos incluye opciones de **Busqueda en Web**, aunque no es tan efectiva. En caso de que sepamos su enlace ED2K (*eDonkey2000*) directo, lo introduciremos en la casilla correspondiente.

## Opcion Trafico

Esta pantalla se convertira en un elemento muy familiar en su vida cuando se familiarice con *eMule*. Nos proporcionara los siguientes datos de nuestras descargas: nombre del archivo, tamaño total, cantidad descargada, velocidad de la descarga, barras de progreso de la descarga, fuentes de descarga, prioridad, y cantidad que falta de la descarga.

## 13.6. Ayuda con eMule

Como hemos visto con anteriores aplicaciones en PHP, existen numerosas páginas que pueden darnos información

Foto (realizada con pappb) donde podemos plantear nubes rasas dudas, si es que no se encuentra entre la gran cantidad de preguntas resultados.

<http://www.emule-es.tk>

Esta excelente página, dedicada a diversos programas de descarga, cuenta con un estupendo tutorial de funciónamiento y configuración de Emule que no pasa nada por alto. Muy recomendable.

<http://www.zonap2p.com/>

Práctica oficial de EMule, por lo que debe ser nuestra primera opción cuando tengamos alguna duda o deseemos descargar la última actualización. Su único inconveniente es que esta en inglés.

Figura 13.8. Tráfico de descargas.



<http://www.emule-project.net/>

### Ways of importance:

Y responder a todos nuestros dudas. Veamos algunas de las más importantes:

www.zonap2p.com :: Ver tema - FAQ. Preguntas frecuentes sobre el cliente emule - Microsoft Internet Explorer

Archivo Edición Ver Favoritos Herramientas Ayuda

Dirección: http://www.zonap2p.com/viewtopic.php?t=554

## FAQ. Preguntas frecuentes sobre el cliente emule

[nuevo tema](#) [cerrado](#) Foros de discusión -> eMule

Ver tema anterior :: Ver tema siguiente

**Autor**

cariatide  
e-**VETERINARIO** de  
BURGOPOLIS



Estado: Offline

Registrado: 13 Ago 2002

Mensajes: 1741

**Mensaje**

Publicado: Mar Nov 05, 2002 9:02 pm Asunto: FAQ. Preguntas frecuentes sobre el cliente emule

### FAQ. Preguntas frecuentes sobre el cliente emule

---

- ¿Qué es el Emule?
- ¿Cómo funciona el eMule?
- Modificaciones en la última versión disponible: Emule0.26d
- Empezando con el eMule...
- Preferencias - Configuración del emule
- Iconos de conexión en el emule: ID alto o bajo
- Puertos predeterminados del eMule
- ¿Cómo añadir servidores en el eMule?
- ¿Cómo utilizar la búsqueda de archivos del eMule?
- Colores en la barra de descargas
- ¿Qué es el eMule?

Figura 13.9. Una de las mejores páginas de consulta.

# Programacion orientada a objetos

## 14.1. Introducción

La programacion orientada a objetos (termino hartamente conocido para programadores de otros lenguajes, como por ejemplo Java) puede sernos util a la hora de afrontar proyectos de gran tamaño con PHP y MySQL.

Entre las ventajas que presenta la programacion orientada a objetos esta la de permitirnos una reutilizacion mucho mas sencilla del codigo que escribamos. Por ello, si nos enfrentamos a un proyecto largo y complejo, en el que las mismas tareas se repiten numerosas veces, es aconsejable dedicar unas cuantas horas a planificar la estructura de nuestro codigo en base a la programacion orientada a objetos (POO).

## 14.2. Concepto

El concepto de la programacion orientada a objetos no es facil de comprender a simple vista. Por ello, no se desanime si en principio le parece poco menos que una complicacion o un galimatias ininteligible.

Poco a poco, a base de ejemplos practicos, iremos dando un poco de luz; y al final del capitulo tendra usted una vision bastante clara de la POO.

El objetivo de la programacion orientada a objetos es crear piezas de codigo modulares que incluyan en un mismo *script* datos, en forma de variables, y el codigo que los hace funcionar de una determinada manera. De esta manera, los

*scripts* realizan su tarea de manera independiente al resto del programa, por lo que pueden ser incluidos en cualquier parte del código con minimas modificaciones.

La diferencia entre un programa en PHP convencional, como los que hemos visto a lo largo de este libro, y un programa realizado mediante POO, estriba en que el código "convencional" forma un todo indivisible desde la primera linea hasta la ultima. Cualquier cambio o modificación afecta a la integridad del código. Un código escrito en POO estara formado por pequeños bloques independientes cuyos cambios son independientes del resto del código.

No nos llamemos a engaño. El resultado final, usando un tipo de programación u otro sera exactamente el mismo. Por otro lado, la programación orientada a objetos es, sin lugar a dudas, mas complicada, y requiere una planificación mucho mayor. ¿Dónde esta la ventaja entonces? Que su código es facilmente reutilizable. Utilizando la POO nos ahorraremos muchas tareas repetitivas, tales como escribir la conexión a la base de datos cada vez que creemos un nuevo código, introducir el usuario o la contraseña, paginar los resultados, etc., etc. Simplemente, tendremos que coger el "bloque" que realiza esa función en el anterior código y "encajarlo" en el nuevo con las minimas modificaciones.

## 14.3. Clases y objetos

A continuación introduciremos dos conceptos teóricos básicos a la hora de ver la programación orientada a objetos.

Por objeto podemos entender cualquier ente del mundo real (un coche, un gato, una piedra, un cliente...) o cualquier término conceptual que solo existe en el mundo del *software*, tal y como un área de texto o un botón. El objetivo de la POO es construir una serie de objetos que incluyan en sí mismos sus propias características o atributos, y una serie de operaciones que les permitan realizar nuestros deseos.

Desde un punto de vista físico, un objeto puede estar representado por un frigorífico (una caja de metal que incorpora un mecanismo específico que conserva los elementos a baja temperatura), o por un camarero (un ser humano con uniforme que realiza la función de servirnos el café cada mañana).

Los objetos, asimismo, pueden agruparse en clases. Las clases pueden contener diversos objetos. La clase *electrodomesticos*, por ejemplo, incluirá a nuestro frigorífico, y a otros muchos objetos que guardan similitud con él, como las batidoras o las lavadoras. La clase *profesiones* incluye diversos oficios, aparte del de camarero, como el de cartero, o el de escritor de guías de informática.

En *software*, pasa exactamente lo mismo. Una clase contiene diversos objetos que realizan funciones, todas diferentes pero con un componente en común.

## 14.4. Características de clases y objetos

Las clases y objetos, para poder ser considerados como tales, deben poseer una serie de características, tales como:

- **Polimorfismo.** Diferentes clases pueden tener diferentes comportamientos para la misma operación. En el mundo real esto es bastante obvio. En la clase *electrodomesticos*, existe el objeto lavavajillas, cuya función todos conocemos. Y en la clase *profesiones* existe el objeto friega-platos, que realiza la misma función. Sin embargo, aunque realicen la misma tarea, nadie confunde a un hombre con una máquina. Desgraciadamente, las cosas en el mundo de los lenguajes de programación no son tan sencillas. Es por ello que necesitamos del polimorfismo, para saber qué objeto debemos usar para cada función. Volviendo a los ejemplos del mundo real, aunque el lavavajillas (máquina) y el friega-platos (hombre) realicen la misma función, si tenemos una tarea consistente en fregar cinco platos en un quinto piso sin ascensor, es evidente que debemos encargar la tarea al hombre antes que a la máquina (a menos que nos llamemos Lou Ferrigno). Dado que el sentido común que hemos empleado para esta decisión no existe en el mundo del *software* (ni en el de aquellos que trabajan en él), necesitamos la cualidad del polimorfismo para poder encontrar el objeto óptimo para cada tarea.
- **Herencia.** Una opción que nos permite ahorrarnos líneas de código y dar más fluidez a nuestros *scripts*. Las clases presentan relaciones jerárquicas, de tal

manera que las funciones y características de digamos, la clase **A**, pueden ser heredadas por diversas subclases, llamémoslas **B** y **C**. Volviendo a los ejemplos del mundo real, características de la clase *electrodomésticos*, como la de pesar 100 Kg y la función de lavar ropa, pueden ser heredadas por un miembro de otra clase (el que les habla, por ejemplo).

Conviene tener en cuenta que en PHP la herencia solo funciona en una dirección. Es decir, "Platón es un hombre", pero no todos los hombres somos Platón.

## 14.5. Creacion de clases, atributos y funciones en PHP

Tras esta serie de consideraciones mas o menos abstractas, veamos de una vez como se crean las clases, los atributos y las funciones en PHP.

### 14.5.1. Creacion de una clase

```
<?
class nombre
{
}
?>
```

Así de sencillo. Sin embargo, el viaje no ha hecho mas que empezar. Hemos creado la clase **nombre**, utilizando la sentencia **class**. Sin embargo, esta clase, para que nos sirva de algo, debemos dotarla de una serie de atributos y funciones.

```
<?
class nombre
{
var $atributo1;
var $atributo2;
}
?>
```

Un pequeño paso adelante. Con el código anterior hemos creado la clase **nombre**, y la hemos otorgado dos atributos.

Ahora aprenderemos a definir funciones para nuestra clase:

```
<?
class nombre
{
function lavar($param1, $param2, $param3)
{
}
}
?>
```

Hemos otorgado a nuestra clase nombre la función **lavar**, para lo que empleara tres parametros.

## 14.5.2. Constructores

Por este nombre se define un tipo especial de funciones entre cuyas características esta la de llamarse igual que la clase a la que pertenece, y por realizar una serie de tareas imprescindibles para el funcionamiento de la clase, como por ejemplo, dar los valores de inicio para determinadas funciones.

Veamos como definir un constructor para nuestra clase nombre:

```
<?
class nombre
{
function clase($param)
{
echo "Se crea un constructor que emplea el
parámetro $param <br>";
}
}
?>
```

## 14.5.3. Creación de objetos dentro de una clase

Tras aprender a declarar las clases, ahora mostraremos como crear objetos dentro de ellas. Los objetos, como hemos visto, son individuos que pertenecen a esa clase (batidoras, hornos microondas, etc., son todos objetos de la clase electrodomésticos). A este proceso tambien se le conoce como "instanciar" una clase, o crear una instancia.

Los objetos se crean utilizando la sentencia **new**, y deberemos especificar a qué clase pertenecerá este objeto, y dotarle de los parámetros requeridos por nuestro constructor.

Veámoslo con un ejemplo.

```
<?
class ejemplo
{
function ejemplo($param)
{
echo "Este constructor está utilizando el
parámetro $param <br>";
}
}
$objetoa = new ejemplo("uno");
$objetob= new ejemplo("dos");
$objetoc= new ejemplo ("tres");
?>
```

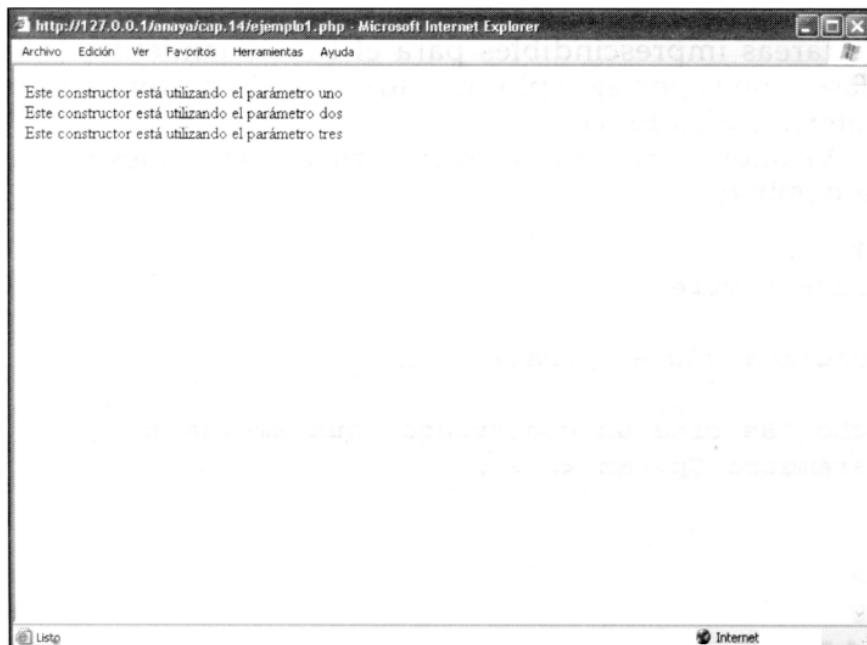


Figura 14.1. Ejemplo de funcionamiento de una clase.

## 14.6. Como utilizar los atributos de la clase

Otra de las sentencias que podemos utilizar dentro de una clase es la llamada **\$this**. Hace las funciones de un

puntero. Lo utilizamos para otorgar a determinada variable, dentro de la clase, las propiedades de un determinado atributo.

Veamoslo mas claro con un ejemplo. Supongamos que dentro de la clase `ejemplo` hemos creado un atributo llamado `$atributo`, y una variable llamada `$variable` (la originalidad tuerce el brazo frente a la sencillez!).

Para otorgar a la variable `$variable` las características especiales del atributo `$atributo`, utilizamos el puntero `$this` de la siguiente manera:

```
$this-> atributo = $variable.
```

El codigo completo seria asi:

```
<?
class ejemplo
{
var $atributo;
function ejemplo($variable)
{
$this->atributo = $variable
echo $this->atributo;
}
}
?>
```

### 14.6.1. Funciones de acceso

En el ejemplo anterior estamos accediendo a los atributos de la clase directamente, es decir, toda la operación se realiza siempre dentro de la clase. Una manera alternativa de realizar las cosas es mediante la creación de funciones de acceso, que nos permitirán un considerable ahorro de código, puesto que los accesos se realizaran con una sola linea.

Veamos estas funciones con un ejemplo:

```
<?
class ejemplo
{
var $atributo;
function acc_atributo()
{
return $this->atributo;
}
function ent_atributo($nuevo_valor)
```

```
{  
    $this->atributo = $nuevo_valor;  
}  
}  
?  
>
```

Mediante este código, aparentemente tan farragoso, hemos creado dos funciones (`acc_atributo()` y `ent_atributo()`), que nos permiten acceder al atributo. La primera de ellas, `acc_atributo()`, simplemente devuelve el valor de `$atributo` (mediante la sentencia `return`). La segunda de ellas, `ent_atributo()`, lo que hace es asignarle un nuevo valor (`$nuevo_valor`).

Aunque aparentemente estas funciones no nos aporten gran cosa ni permitan nada que no pudieramos hacer con anterioridad, de cara a construcciones futuras, el utilizar las funciones de acceso implica que solo una determinada sección del código es la que accede a algun atributo en particular. De esta manera, es mucho mas sencillo establecer mecanismos de control que aseguren que el valor solo se almacenara en el caso de que cumpla las condiciones que nosotros hemos fijado.

Si, por ejemplo, queremos controlar que el valor de `$nuevo_valor` sea menor a 1987 (por ejemplo, que solo puedan acceder las personas que sean mayores de 18 años), con una sola linea de código hemos solventado el problema.

```
function ent_atributo($nuevo_valor)  
{  
    if ($nuevo_valor < 1974)  
        $this->atributo = $nuevo_valor;  
}
```

Aunque a primera vista no parezca un gran avance, tengamos en cuenta de que de no tener esta función de acceso tendríamos que haber revisado nuestro código línea por línea y establecer sentencias de control siempre que encontrásemos la palabra `$atributo`.

#### **14.6.2. Acceder a las operaciones dentro de las clases**

De la misma manera que hemos accedido a los atributos dentro de una clase, podemos hacer algo similar con las operaciones. Veámoslo con un ejemplo:

```
class ejemplo
{
function operacion1()
{
}
function operacion2 ($parametro1, $parametro2)
{
}
$a= new ejemplo();
}
```

Gracias al nuevo objeto \$a que hemos creado podemos utilizar las operaciones de la misma manera que antes hemos utilizado otras funciones: simplemente, usando su nombre e introduciendo cualquier parametro que necesitemos entre parentesis. Pero no olvidemos que, dado que cada operación pertenece a un objeto, necesitamos especificar a su vez a que objeto pertenece.

```
$a->operacion1();
$a->operacion2(4, 16);
```

Si estas operaciones devuelven algun valor, podemos capturarla de la siguiente manera:

```
$b= $a->operacion1();
$c= $a->operacion2 (4,16) ;
```

## 14.7. Herencia en PHP

Otra de las posibilidades que nos ofrece la POO es que una clase herede las propiedades de otra. Esto se hace mediante la sentencia `extends`.

Veamoslo con un ejemplo en el que tenemos dos clases, `padre` e `hijo`, en el que la clase `hijo` hereda de la clase `padre`. Supongamos que la clase `padre` ya ha sido definida previamente.

```
class hijo extends padre
{
var $atributob;
function operacionb()
{
}
}
```

Si la clase padre hubiese sido definida de la siguiente manera:

```
class padre
{
var $atributoa;
function operaciona()
{
}
}
```

Dado que la clase hijo ha heredado de la clase padre, gracias a la sentencia `extends`, las siguientes sentencias serian válidas:

```
$b = new hijo();
$b ->operaciona();
$b->atributoa= 2;
```

Tenga en cuenta que nos estamos refiriendo a `operaciona()` y a `atributoa`, que fueron declaradas en la clase `padre`, pero que, debido a la herencia, podemos utilizarlas libremente con un objeto que ha sido declarado en la clase hijo.

No conviene olvidar que la herencia solo funciona en una dirección. La clase hijo puede utilizar funciones y atributos declarados en la clase padre, pero las que declaremos en la clase hijo no podrán ser empleadas en la clase padre.

## 14.8. Ejemplo. Programacion orientada a objetos vs. Programacion convencional

Tras haber analizado teoricamente los fundamentos de la programacion orientada a objetos, vamos a comparar este estilo de programar con el estilo convencional, que hemos estudiado a lo largo de este libro, con un sencillo ejemplo.

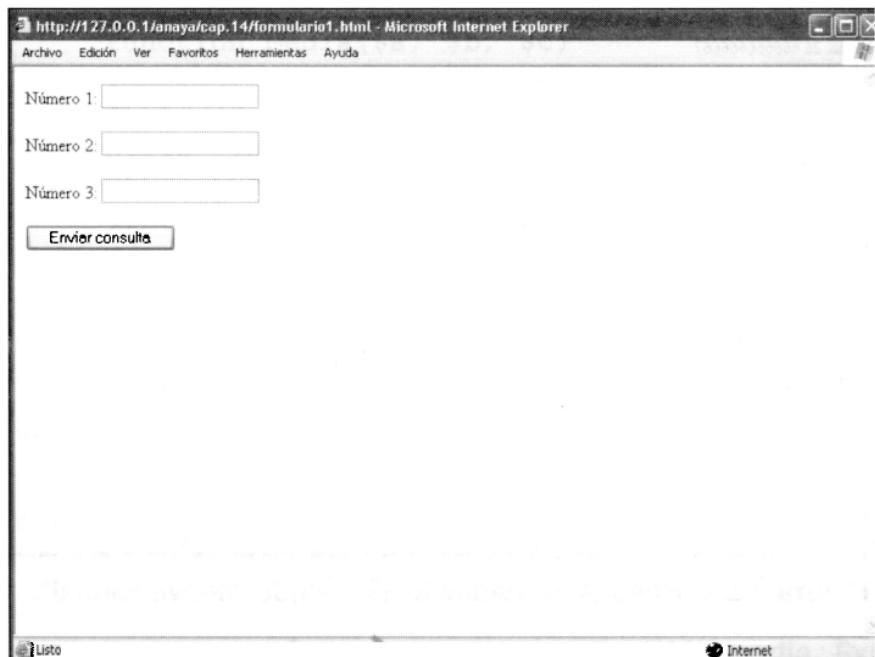
Crearemos un *script* que calcule la media aritmetica de tres cifras que nosotros introduzcamos, de las dos maneras. Examinemos el estilo "convencional".

Empezaremos creando un formulario en html, gracias al cual podremos introducir los tres numeros, que seran mandados vía `get` a otro *script* en php llamado `calculol.php`

```

<!-- formulariol.html -->
<form action="calculol.php" method="get">
    Número 1: <input type="text" name=
    "a" /><br /> <br>
    Número 2: <input type="text" name=
    "b" /><br /> <br>
    Número 3: <input type="text" name=
    "c" /><br /> <br>
    <input type="submit" />
</form>

```



**Figura 14.2.** Formulario en HTML.

El *script* siguiente crea una variable llamada **\$media**, que recoge los numeros que hemos introducido en el formulario y, mediante una sencilla operación aritmética, nos calcula la media. El siguiente paso es mostrar ese valor en pantalla, así como los valores que hemos introducido inicialmente.

```

<!-- calculol.php -->
<?
$media= ($a+$b+$c)/3 ;
?>
<html>

```

```

<body>
    Número 1: <?=$a?><br />
    Número 2: <?=$b?><br />
    Número 3: <?=$c?><br />
    Media: <?=$media?><br />
</body>
</html>

```



**Figura 14.3.** Resultado mediante el método "convencional".

Claro y sencillo, ¿verdad? No precisa de mayor explicación. De hecho, en este caso no merecería la pena plantearnos la programación orientada a objetos, pero hemos elegido un ejemplo tan sencillo a efectos didácticos.

```

<!-- formulario2.html -->
<form action="dom.php" method="get">
    Número 1: <input type="text" name="a" /><br />
    Número 2: <input type="text" name="b" /><br />
    Número 3: <input type="text" name="c" /><br />
    <input type="submit" />
</form>

```

En este caso, el formulario de inicio es identico al ejemplo anterior, con la diferencia de que los datos son enviados via get a un script llamado dom.php.

Pero antes, examinaremos el siguiente código:

```
<!-- media.php -->
<?php
    class media
    {
        var $a;
        var $b;
        var $c;
        function media($a, $b, $c)

            $this->a = $a;
            $this->b = $b;
            $this->c = $c;
        }
        function total()
        {
            return ($this->a + $this->b + $this->c)/3;
        }
    }
?>
```

Examinemoslo con sumo detenimiento, pues en él se resume todo lo que hemos visto a lo largo del presente capítulo en relación a la programación orientada a objetos.

```
class media
```

El primer paso: creamos una clase llamada **media**. Esta clase, como hemos visto con anterioridad, incluirá una serie de atributos, funciones y objetos.

```
var $a;
var $b;
var $c;
```

He aquí los tres atributos, que coincidirán con las tres variables que hemos recogido en el formulario de inicio.

```
function media($a, $b, $c)
{
    $this->a = $a;
    $this->b = $b;
    $this->c = $c;
}
```

A continuación, creamos la función `media()`, que incluye estos tres atributos, y usamos el puntero `$this->`, para usar estos atributos dentro de unas variables.

```
function total()
{
    return ($this->a + $this->b + $this->c)/3;
}
```

La última de las funciones, llamada `total()`, es la que realiza el trabajo final. Coge estas variables y opera con ellas para hallar la medida aritmética.

Veamos ahora el último de los *scripts*:

```
<!-- dom.php -->
<?php
    require('media.php');
    $r= new media($a,$b, $c);
?>
<html>
    <body>
        Número 1: <?=$r->a?><br />
        Número 2: <?=$r->b?></p>
        Número 3: <?=$r->c?></p>
        <p>Media: <?=$r->total()?><br /></p>
    </body>
</html>
```

El primero de los pasos es abrir el *script* `media.php`. Aquí vemos una de las características más claras de la POO: la modularidad. El *script* `media.php` actúa de manera independiente, y puede ser abierto y utilizado por cualquier otro *script*, dentro de este código o dentro de cualquier otro.

El siguiente paso es crear, mediante la sentencia `new`, un objeto perteneciente a la clase `media`, y que utilizará tres variables, (`$a`, `$b` y `$c`).

```
<p>Media: <?=$r->total()?><br />
```

Y por último, este objeto invoca la función `total()` que, como recordamos, es la que realiza la operación de media aritmética, empleando en este caso los valores que le hemos pasado mediante el formulario.

El resultado en ambos casos es el mismo. E incluso el ejemplo primero de programación convencional es más corto y más sencillo de realizar. Pero las ventajas de la

POO tambien son claras. Aparte de la gran elegancia del código, si en un futuro tuviésemos que crear un programa, que entre muchas otras cosas realizase la media aritmética de tres cifras distintas, simplemente cogeríamos el modulo **media.php** y lo incluiríamos tal cual en el nuevo código, mientras que de la otra manera tendríamos que rescribirlo prácticamente desde cero.



**Figura 14.4.** Resultado mediante POO.

# Trabajar con multiples bases de datos

## 15.1. Introducción

En el capitulo 7 vimos las ventajas de trabajar con diversas bases de datos. Dividiendo nuestros datos en diversas tablas, se evita la redundancia de datos, permite una busqueda mucho mas rapida, y las modificaciones son mas sencillas.

En el presente capitulo estudiaremos como se relacionan las diversas bases de datos entre si, desde un punto de vista mas riguroso y teorico, y lo pondremos en practica con una utilisima aplicacion que tiene cabida en cualquier pagina Web: un foro, en el que podremos crear diferentes temas de discusion, y cualquiera podra dejar sus mensajes en ellos.

## 15.2. Diseñar las relaciones entre nuestras bases de datos

Si bien este tema es objeto de numerosos tratados teóricos y asignatura fundamental en las mas importantes escuelas de informática del mundo, afrontaremos el diseño relacional de nuestras bases de datos desde un punto de vista ameno y práctico, plagado de ejemplos, sin renunciar en ningun caso al rigor.

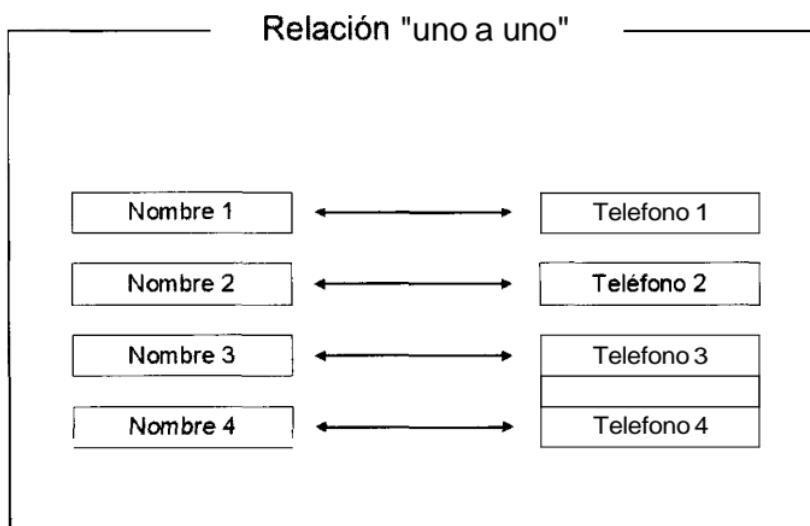
Las diferentes formas de relación entre diversas bases de datos que podemos encontrar son:

### 15.2.1. Relaciones de "uno a uno"

Estas relaciones entre bases de datos se dan cuando cada campo clave aparece solo una vez en cada una de las tablas.

Tomando un ejemplo del mundo real, una clara relación de "uno a uno" podría ser (o por lo menos así era hace años), el nombre de cualquier persona y su número de teléfono. Si partimos del supuesto en que cada persona tiene un solo número de teléfono, se podría hablar de una relación de "uno a uno".

Graficamente, se podría representar de la siguiente manera:



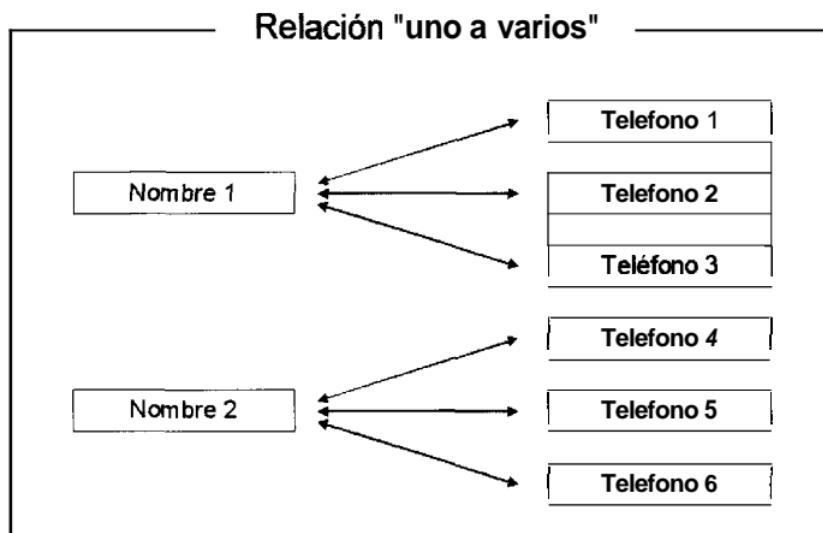
**Figura 15.1.** Esquema de relación "uno a uno".

Este tipo de relaciones se caracterizan porque cada uno de los campos define a aquel con el que se relaciona. Es decir, conociendo el nombre de una persona podemos conocer su número de teléfono. Si sabemos el número de teléfono, podemos identificar al dueño. En estos casos, se suele aconsejar incluir todos los datos dentro de una sola tabla.

### **15.2.2. Relaciones de "uno a varios"**

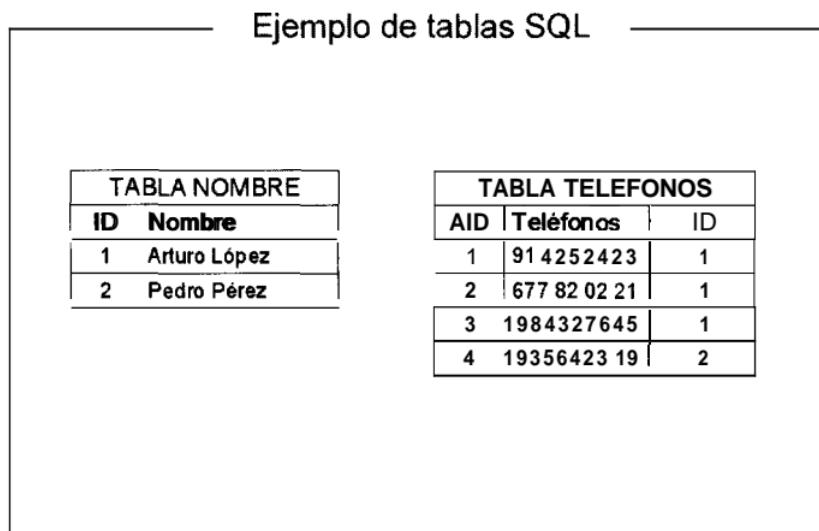
El ejemplo del caso anterior (cada persona, un teléfono), si bien es correcto teóricamente, es muy improbable desde el punto de vista de la realidad. Con la gran expansión de los teléfonos móviles, por lo general, cada persona tiene un número de teléfono fijo y, además, el del teléfono móvil. Debemos tener en cuenta que además del de su casa también tendrá un número de teléfono de empresa, y que quizás también sus teléfonos móviles estén divididos en ocio y trabajo.

Por ello, debemos tener nuestras bases de datos preparadas para ello. Este tipo de relaciones es conocido como de "uno a varios", y se podría representar de la siguiente manera.



**Figura 15.2. Esquema de relación "uno a varios".**

En este caso, lo aconsejable no es almacenar todos los datos en una sola tabla, sino lo eficiente es hacerlo en tablas separadas, utilizando el identificador ID para relacionarlas, como hemos visto numerosas veces a lo largo de este libro.



**Figura 15.3. Ejemplo de tablas SQL.**

Echemos un vistazo a la figura anterior. En la tabla **Nombre** almacenamos el nombre y apellidos, con su ID o número identificador. En la otra tabla, **Telefonos**, almacenamos únicamente números de teléfono, con su correspondiente número identificador, en este caso **AID**. La manera en que se relaciona una con otra es mediante el identificador **ID**, que está presente en ambas tablas.

A simple vista podemos advertir que la primera de las personas de la tabla de nombres, *Arturo López*, tiene tres números de teléfono, pues su ID, que en este caso es uno, aparece en tres de los teléfonos de la otra tabla.

De este modo, será mucho más sencillo cambiar, eliminar o ampliar los números de teléfono de cada una de las personas, que si las tuviésemos en la misma tabla.

Si estas tablas están creadas en MySQL (cosa que hemos aprendido a lo largo del presente libro), la sentencia que nos ayudaría a encontrar todos los teléfonos de una determinada persona sería:

```
select telefono from telefonos, nombres, where
nombre="Arturo Lopez" and nombres.id=telefonos.id;
```

### 15.2.3. Relaciones de "varios con varios"

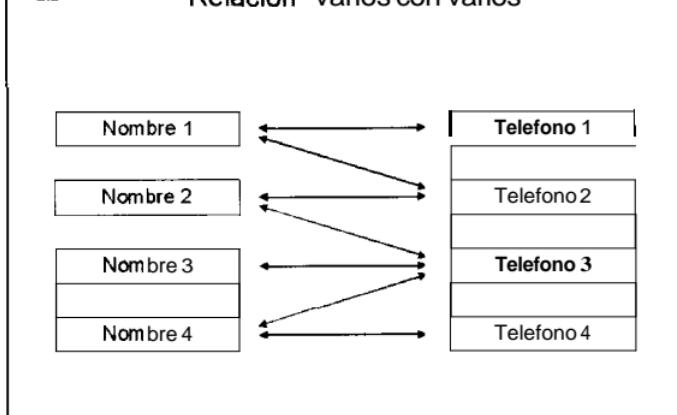
La última de las relaciones que podemos encontrarnos es la de "varios con varios". Dado que en la vida las cosas rara vez son sencillas, este será el tipo de relación que nos encontraremos más a menudo.

Volviendo al tema de los teléfonos, hemos encontrado la manera de relacionar cada una de las personas con sus diversos teléfonos: el de su casa, el de su empresa, el móvil. Pero no será extraño tener en nuestra base de datos diversas personas que trabajen en la misma empresa, por lo que su número de trabajo será el mismo, o miembros de una misma familia, por lo que compartirán el número de teléfono de su hogar.

¿Cómo tratar este tipo de relaciones? Si nos limitamos a repetir dicho número en la tabla, estaremos creando problemas de redundancia de datos, que a largo plazo lastrarán la rapidez y eficacia de nuestras tablas.

Este tipo de relaciones podría ilustrarse de la siguiente manera:

### Relación "varios con varios"



**Figura 15.4.** Ejemplo de relación "varios con varios"

Como vemos, cada elemento de la base de datos puede relacionarse libremente con uno o varios miembros de las distintas tablas.

En estos casos no hay una regla fija a la que podamos acogernos, pero lo aconsejable es aproximarse lo mas posible a la realidad, y no dudar en establecer tablas intermedias que nos ayuden a asociar mejor los datos.

Volviendo al tema de los telefonos, imaginemos que varias personas de nuestra tabla trabajan en la misma empresa, Acme Producciones. En este caso, su numero de trabajo seria el mismo. Pero para colmo, la centralita de Acme Producciones tiene varias líneas, por lo que los numeros de telefono de trabajo de estas personas serian varios. ¿Cómo representarlo en nuestra base de datos?

### Ejemplo de tablas SQL

Diagrama que muestra tres tablas SQL:

- TABLA NOMBRES**

ID	Nombre	AID
1	Arturo López	1
2	Pedro Pérez	1
- TABLA EMPRESAS**

EID	Empresa	AID
1	ACME	1
2	IDOWEB	2
- TABLA TELEFONOS**

ID	Teléfono	EID
1	91 425 24 23	1
2	677 82 02 21	1
3	98 432 76 45	2

**Figura 15.5.** Ejemplo de tablas SQL.

En este caso hemos creado una tabla intermedia llamada **Empresas**. En la tabla **Nombres** incluimos un nuevo campo **AID**, que se relaciona con la tabla **Empresas**, y es esta tabla la que se relaciona directamente con los teléfonos. De esta manera, podemos almacenar todos los datos con facilidad sin tener que repetir un solo numero de teléfono.

### 15.2.4. Conclusion

Por muy complicadas que parezcan las relaciones en el mundo real, tengamos por seguro que cuando queramos plasmarlas en nuestra base de datos correspondera a alguna de las tres opciones que hemos presentado. Por ello, no dudemos en invertir el tiempo que sea necesario hasta encontrar la combinación debases de datos optima que nos permita modelar la realidad sin repetir ninguno de los datos.

El tiempo que invirtamos en este proceso lo recuperaremos con creces durante el proceso de programacion, pues nos facilitara enormemente las cosas.

## 15.3. Creación de un foro. Ejemplo práctico

Los foros son una de las aplicaciones mas utilizadas y populares que se pueden encontrar en Internet.

Se trata de una aplicacion en la que alguien (normalmente el moderador o *Web master*) propone un tema de discusion, y la gente puede dejar sus opiniones sobre dicho tema para que todo el mundo las lea. Tambien suele darse la libertad de que las personas creen sus propios temas de discusibn, sin necesidad de supervision del *Web master*, que es el ejemplo que vamos a realizar nosotros.

### 15.3.1. Caracteristicas del foro

- Cualquier usuario podrá crear su propio tema de discusion, o responder a alguno ya existente.
- Cada vez que alguien introduzca un tema o una respuesta a alguno de los temas, deberá grabarse su dirección de correo electrbnico, así como la hora en que ha sido introducido.

- Deberemos recoger en una sola tabla todos los temas existentes, así como el número de respuestas que incluya cada tema. Desde esa tabla deberemos ver el número de respuestas a cada tema y, pinchando sobre el, leer todas las respuestas
- Todo ello con un diseño sencillo y claro, pero elegante a la vez.

¿Se atreve?

### 15.3.2. Bases de datos

Analizando las necesidades de un foro, podemos darnos cuenta que se trata de una relación de "uno a varios" entre las bases de datos. Esto es así debido a que cada tema del foro puede tener varias respuestas, pero cada respuesta solo pertenece a un tema. Y tal y como hemos visto en este capítulo, la mejor manera de mostrar este tipo de relaciones es mediante dos bases de datos.

A la primera de ellas la llamaremos `temas_foro` y tendrá el siguiente código:

```
create table temas_foro (
    tema_id int not null primary key
    auto_increment,
    tema_titulo varchar (150),
    tema_fecha datetime,
    tema_creador varchar (150)
);
```

Analicémoslo línea por línea:

```
create table temas_foro (
```

Instrucción en SQL para crear una tabla llamada `temas_foro`.

```
tema_id int not null primary key auto-increment,
```

El identificador de cada tema se llamará `tema_id`, y será un número único que se incrementa automáticamente. No podrá ser nulo.

```
tema_titulo varchar (150),
```

En este campo almacenaremos el título del tema. Longitud máxima, 150 caracteres.

```
tema_fecha datetime,
```

Mediante la sentencia `datetime`, almacenamos automáticamente la fecha en que se da de alta el nuevo tema de discusion.

```
tema_creador varchar (150) ;
```

En este campo se almacenara la dirección de correo electrónico de la persona que crea este nuevo tema de discusion.

La siguiente de las tablas, la que almacenara las respuestas a cada tema de discusion, se llamara `respuesta-foro`, y tendra el siguiente codigo:

```
create table respuesta_foro (
respuesta-id int not null primary key
auto-increment,
tema-id int not null,
respuesta-texto text,
respuesta-fecha datetime,
respuesta-creador varchar (150)
);
```

Analicemoslo linea por linea:

```
create table respuesta_foro (
```

Instrucción en SQL para crear una tabla llamada `respuesta-foro`.

```
respuesta-id int not null primary key
auto-increment,
```

El identificador de cada respuesta se llamara `respuesta-id`, y sera un numero unico que se incrementa automáticamente. No podra ser nulo.

```
tema-id int not null,
```

Con esta linea asociamos a cada respuesta con el tema de discusion al que pertenece. Fijemonos que estamos almacenando un campo llamado `tema-id`, que coincide con el identificador unico de cada tema de discusion. Esta es la mejor manera para relacionar las bases de datos en los casos de "uno con varios".

```
respuesta-texto text,
```

En este campo almacenaremos las distintas respuestas que pueda tener cada tema.

```
respuesta-fecha datetime,
```

Gracias a la sentencia `datetime`, almacenamos automáticamente la fecha en que esta respuesta ha sido realizada.

```
respuesta_creador varchar (150)
);
```

En este campo almacenaremos la dirección de correo electrónico del que realice cada respuesta.

### 15.3.3. *Scripts*

Una vez que hemos creado las bases de datos, y visto cómo se relacionan entre ellas, el siguiente paso será definir los distintos *scripts* que permitirán realizar todo el proceso.

Los *scripts* que necesitaremos son los siguientes:

- `nuevotema.html`
- `nuevotemal.php`
- `listatemas.php`
- `mostrar tema.php`
- `respuesta.php`

Examinemoslos uno por uno:

```
<!-- nuevotema.html -->
<html>
<head>
<title>Añadir un tema de discusión</title>
</head>
<body bgcolor="#FFFFFF">
    <table width="90%" border="0" cellspacing="2"
cellpadding="2"
        bgcolor="#999999">
        <tr bgcolor="#CCCCCC">
            <td>
                <div align="center"><font face="Arial, Helvetica,
                    sans-serif" size="5">Añadir un tema de
discusión
                </font></div>
            </td>
        </tr>
        <tr bgcolor="#FFFFFF">
            <td>
                <form method=post action="nuevotemal.php">
                    <table width="520" border="0" cellspacing="0"
cellpadding="0" align="center">
```

```
    <tr>
<td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
<td>
    <font size="2" face=
        "Arial, Helvetica, sans-serif">
        <b>Dirección de correo:</b></font>
    </td>
<td>
<input type="text" name="tema_creador" size=40
        maxlength=150>
</td>
</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td>
    <font face=
        "Arial, Helvetica, sans-serif" size="2">
        <b>Título del tema:</b></font>
    </td>
<td>
<input type="text" name="tema_titulo" size=40
        maxlength=150>
</td>
</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td valign="top">
    <font face=
        "Arial, Helvetica, sans-serif"
        size="2"><b>Texto:</b></font>
    </td>
<td>
<textarea name="respuesta_texto" rows=8 cols=40
        wrap=virtual></textarea>
</td>
```

```

</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td colspan="2">
<div align="center">
<input type="submit" name="submit" value=
"Añadir Tema">
</div>
</td>
</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
</table>
</form>
</td>
</tr>
</table>
</body>
</html>

```

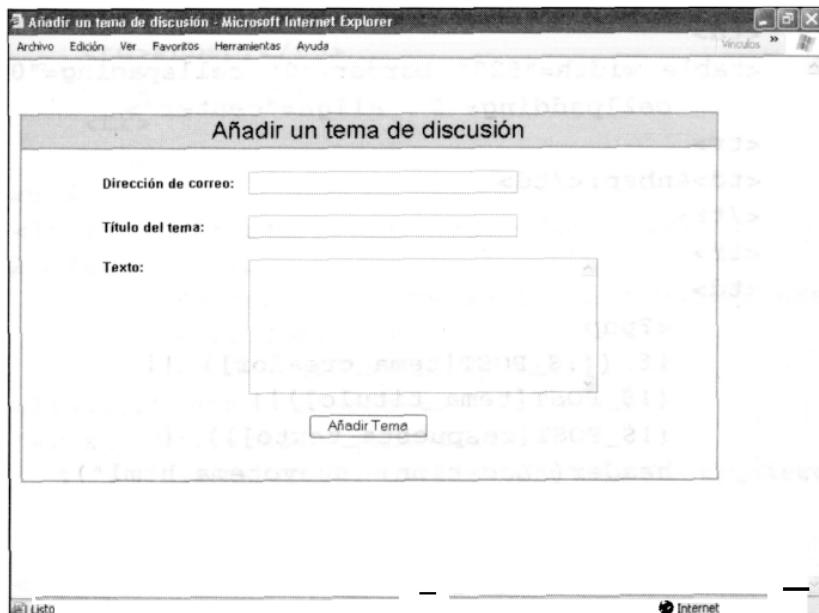


Figura 15.6. Pantalla inicial del foro.

Este primer *script*, realizado en HTML, no es mas que un sencillo formulario en el que recogeremos los datos necesarios para crear un nuevo tema de discusion: el título del tema, el texto, y la dirección de correo electrónico del que lo introduce.

Estos datos son enviados via **post** al *script* nuevo-tema1.php, que veremos a continuación.

```
<!-- nuevotema1.php -->
<html>
<head>
<title>Nuevo tema creado</title>
</head>
<body>
  <table width="90%" border="0" cellspacing=
    "2" cellpadding="2"
    bgcolor="#999999">
    <tr bgcolor="#CCCCCC">
      <td>
        <div align="center"><font face=
          "Arial, Helvetica, sans-serif"
          size="5">Nuevo tema creado</font></div>
      </td>
    </tr>
    <tr bgcolor="#FFFFFF">
      <td>
        <table width="520" border="0" cellspacing="0"
          cellpadding="0" align="center">
          <tr>
            <td>&nbsp;</td>
          </tr>
          <tr>
            <td>
              <?php
                if (( !$_POST[tema_creador]) ||
                  (!$_POST[tema_titulo]) ||
                  (!$_POST[respuesta_texto])) {
                  header("Location: nuevotema.html");
                  exit;

                  $conn = mysql_connect("localhost",
                    "tigreton", "anais") or
                  die(mysql_error());
                }
              </td>
            </tr>
          </table>
        </td>
      </tr>
    </table>
  </body>
</html>
```

```

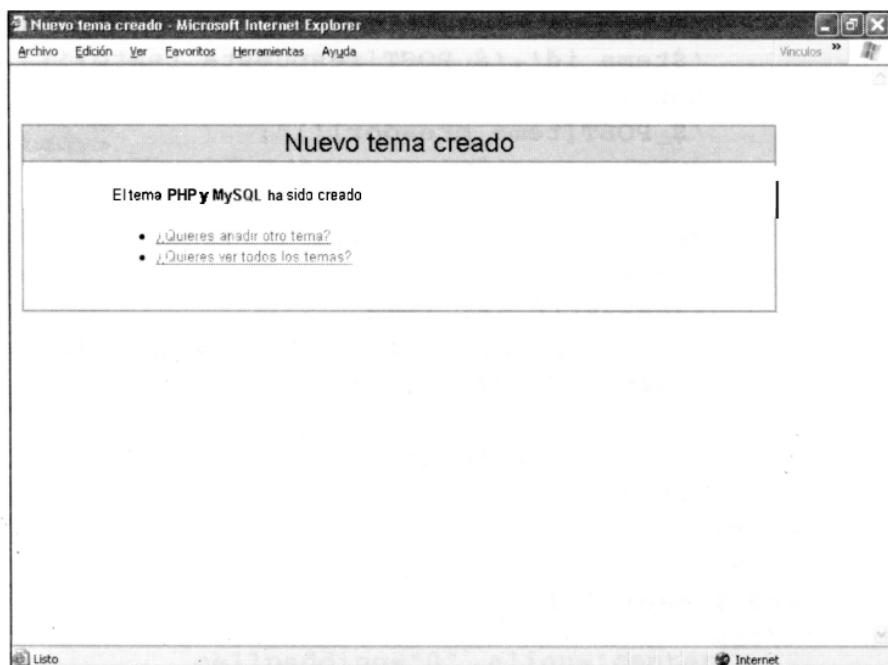
mysql_select_db("foro",$conn) or
die(mysql_error());
$nu_tema =
"insert into temas_foro values ('',
'$_POST[tema_titulo]', now(),
'$_POST[tema_creador]')";
mysql_query($nu_tema,$conn) or
die(mysql_error());
$tema_id = mysql_insert_id();
$nu_respuesta = "insert into
respuesta_foro values ('',
'$tema_id','$_POST[respuesta_texto]',
now(),
'$_POST[tema_creador]')";
mysql_query($nu_respuesta,$conn) or
die(mysql_error());
$msg = "<P><font size='2' face=
'Arial, Helvetica,
sans-serif'>El tema
<strong>$tema_titulo</strong> ha sido
creado.</font></p>" ;
?>
<?php print $msg; ?>
</td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>
<ul>
<li><a href="nuevotema.html"><font size=
"2" face="Arial,
Helvetica, sans-serif">&quest;Quieres
a&tilde;adir
otro tema?</font></a></li>
<li><font size="2" face=
"Arial, Helvetica, sans-serif">
<a href="listatemas.php">&quest;Quieres
ver todos
los temas?</a></font></li>
</ul>
</td>
</tr>

```

```

<tr>
<td>&nbsp;</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>

```



**Figura 15.7.** Script nuevotemal.php.

Este *script* es el encargado de almacenar los datos introducidos anteriormente en la base de datos. Una vez terminado este proceso, nos preguntara si queremos introducir un nuevo tema o ver la lista de todos los temas. En caso de elegir esta ultima opción, se nos mandara al *script* listatemas.php.

```

<!-- listatemas.php -->
<html>
<head>
<title>Listado de Temas</title>
</head>
<body bgcolor="#FFFFFF">

```

```

<table width="90%" border="0" cellspacing=
"2" cellpadding="2"
bgcolor="#999999">
<tr bgcolor="#CCCCCC">
<td>
<div align="center"><font face=
"Arial, Helvetica, sans-serif"
size="5">Listado de temas</font></div>
</td>
</tr>
<tr bgcolor="#FFFFFF">
<td>
<table width="600" border="0" cellspacing="0"
cellpadding="0" align="center">
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>
<?php
$conn = mysql_connect("localhost",
"tigreton", "anais") or
die(mysql_error());
mysql_select_db("foro",$conn) or
die(mysql_error());
$get_topics = "select tema_id,
tema_titulo,
date_format(tema_fecha, '%b %e %Y
at %r') as
fmt_tema_fecha, tema_creador from
temas_foro order by
tema_fecha desc";
$get_topics_res =
mysql_query($get_topics,$conn) or
die(mysql_error());
if (mysql_num_rows($get_topics_res)
< 1) {
$display-block =
"<P><em>No existe ningún tema.</em></p>";}
else {
//create the display string
$display-block = "
<table cellpadding=0 cellspacing=
0 border=0>

```

```

<tr height='25' bgcolor="#CCCCCC>
<th><font size='2' face=
'Arial, Helvetica, sans-serif'>
<b>Titulo del tema</b></font></th>
<th><font size='2' face=
'Arial, Helvetica, sans-serif',
<b># de Respuestas</b></font></th>
</tr>" ;
while ($topic_info =
mysql_fetch_array($get_topics_res)) {
$tema_id = $topic_info['tema_id'];
$tema_titulo =
stripslashes($topic_info['tema_titulo']);
$tema_fecha = $topic_info['fmt_tema_fecha'];
$tema_creador =
stripslashes($topic_info['tema_creador']);
$get_num_posts =
"select count(respuesta_id) from
respuesta_foro where tema_id =
$tema_id";
$get_num_posts_res =
mysql_query($get_num_posts,$conn) or
die(mysql_error());
$num_posts =
mysql_result($get_num_posts_res,0,'count(respuesta_id)');
$display-block .= "
<tr>
<td><font size='2' face=
'Arial, Helvetica, sans-serif'>
<a href=
\"mostrar tema.php?tema_id=$tema_id\">
<strong>$tema_titulo</strong></a><br>
creado el $tema_fecha por
$tema_creador<font></td>
<td align=center><font size='2' face=
'Arial, Helvetica,
sans-serif'>$num_posts</font></td>
</tr>" ;
}
$display-block .= "</table>";
}
?>
<?php print $display-block; ?></td>

```

```

</tr>
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>
<ul>
<li><a href="nuevotema.html"><font size=
"2" face="Arial,
    Helvetica, sans-serif">&quest;Quieres
    a&ntilde;adir
    otro tema?</font></a></li>
</ul>
</td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>

```

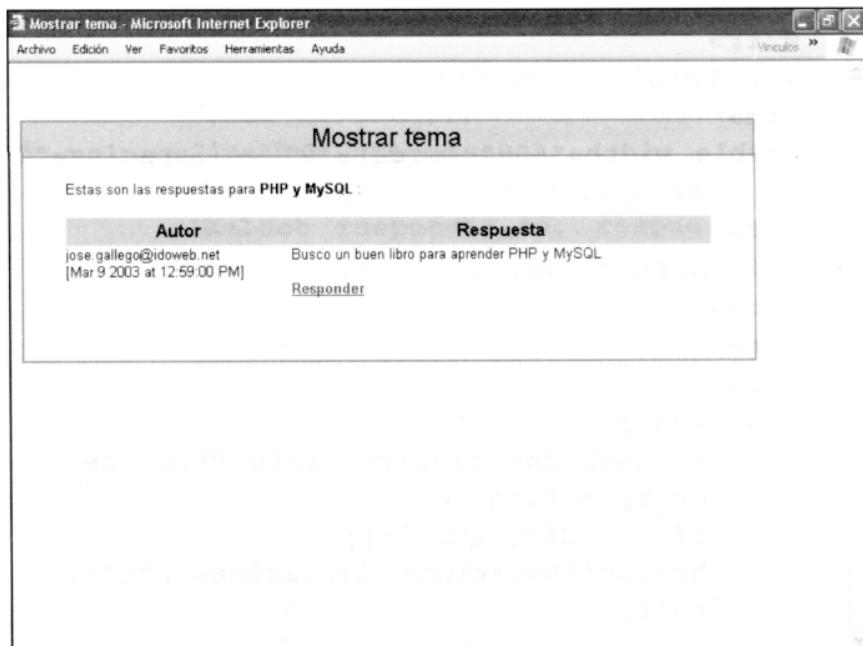


Figura 15.8. Script listatemas.php.

Este *script* es el encargado de crear la tabla donde figuran todos los temas de discusion que se han creado hasta el momento, la fecha en que fueron creados, y la dirección de correo electrónico de quien lo ha creado.

Tambien nos muestra el numero de respuestas que hasta el momento ha tenido cada tema de discusion y, si pinchamos sobre alguno de los temas, se nos enviara al *script* **mostrar tema.php**, donde nos indicaran las respuestas que existen.

```
<!-- mostrar tema.php -->
<html>
<head>
<title>Mostrar tema</title>
</head>
<body bgcolor="#FFFFFF">
    <table width="90%" border="0" cellspacing="2" cellpadding="2" bgcolor="#999999">
        <tr bgcolor="#CCCCCC">
            <td>
                <div align="center"><font face="Arial, Helvetica, sans-serif" size="5">Mostrar tema</font></div>
            </td>
        </tr>
        <tr bgcolor="#FFFFFF">
            <td>
                <table width="600" border="0" cellspacing="0" cellpadding="0" align="center">
                    <tr>
                        <td>&nbsp;</td>
                    </tr>
                    <tr>
                        <td>
                            <?php
                                //check for required info from the
                                query string
                                if (!$_GET[tema_id]) {
                                    header("Location: listatemas.php");
                                    exit;
                                }
                                //connect to server and select database
                            </?php>
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
    </table>
</body>
</html>
```

```

$conn = mysql_connect("localhost",
"tigreton", "anais") or
die(mysql_error());
mysql_select_db("foro", $conn) or
die(mysql_error());
//verify the topic exists
$verify_topic = "select tema_titulo
from temas_foro where
tema-id = $_GET[tema_id]";
$verify_topic_res =
mysql_query($verify_topic, $conn) or
die(mysql_error());
if (mysql_num_rows($verify_topic_res) < 1) {
//this topic does not exist
$display_block = "<P><font size='2' face='Arial, Helvetica,
sans-serif'><b>Ha seleccionado un tema
incorrecto. Por
favor <a href=\"listatemas.php\">vuelva
a intentarlo</a>.
</b></font></p>" ;
} else {
//get the topic title
$tema_titulo =
stripslashes(mysql_result($verify_topic_res, 0,
'tema_titulo'));
//gather the posts
$get_posts =
"select respuesta-id, respuesta_texto,
date_format(respuesta_fecha,
'%b %e %Y at %r') as
fmt_respuesta_fecha, respuesta-creador
from respuesta_foro
where tema-id = $_GET[tema_id] order
by respuesta_fecha
asc";
$get_posts_res =
mysql_query($get_posts, $conn) or
die(mysql_error());
//create the display string
$display_block = "

```

```
<P><font size='2' face=
'Arial, Helvetica, sans-serif'>
```

Estas son las respuestas para

```
<strong>$tema_titulo</strong>:
</font></p>
<table width=100% cellpadding=
0 cellspacing=0 border=0>
<tr height='26' bgcolor='#CCCCCC'>
<th><font size='3' face=
'Arial, Helvetica, sans-serif'>
<b>Autor</b></font></th>
<th><font size='3' face=
'Arial, Helvetica, sans-serif'>
<b>Respuesta</b></font></th>
</tr>";
while ($posts_info =
mysql_fetch_array($get_posts_res)) {
$respuesta_id = $posts_info['respuesta_id'];
$respuesta_texto =
nl2br(stripslashes
$posts_info['respuesta_texto']));
$respuesta_fecha =
$posts_info['fmt_respuesta_fecha'];
$respuesta_creador =
stripslashes($posts_info['respuesta_creador']);
//add to display
$display-block .= "
<tr>
<td width=35% valign=top><font size='2'
face='Arial,
Helvetica, sans-serif's
$respuesta_creador<br>
[$respuesta_fecha]</font></td>
<td width=65% valign=top><font size=
'2' face='Arial,
Helvetica, sans-serif'>
$respuesta_texto<br><br>
<a href=
\"respuesta.php?respuesta_id=
$respuesta_id\">
<strong>Responder</strong>
</a></font></td>
```

```

        </tr>";
    }
    //close up the table
    $display_block .= "</table>";
}
?>
<?php print $display_block; ?></td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
<tr>
<td>&nbsp;</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>

```

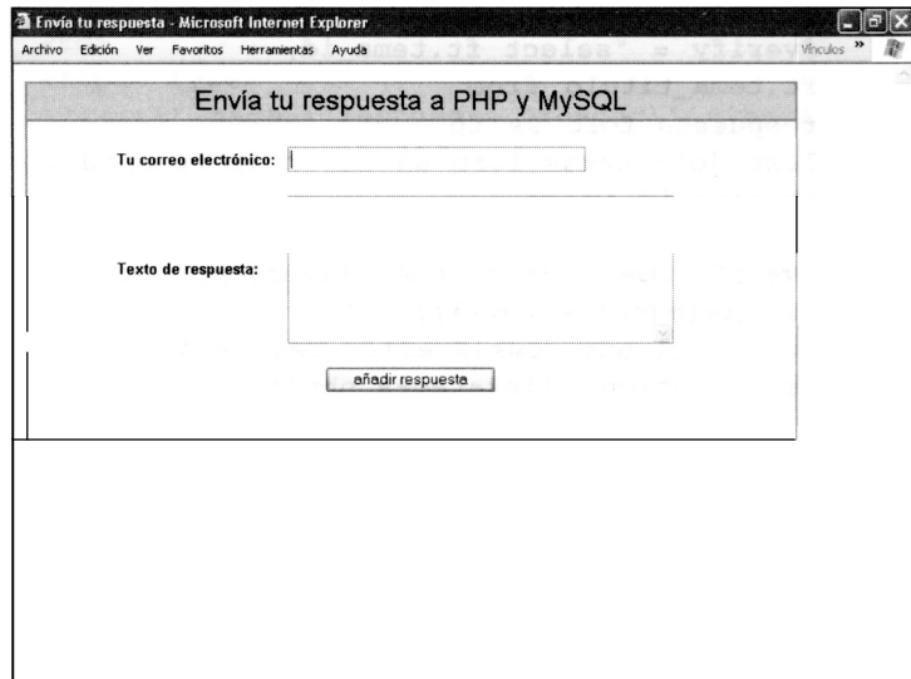


Figura 15.9. Script mostrartema.php.

Este *script* realiza una consulta a la base de datos y nos muestra todas las respuestas correspondientes a cada tema. Recordemos que la relación entre ambas bases de datos se produce gracias al identificador **tema-id**, que relaciona temas de discusion y respuestas.

Y en el caso de que queramos añadir alguna respuesta mas, se nos enviara al *script* **respuesta.php**.

```
<!-- respuesta.php -->
<html>
<head>
<title>Envía tu respuesta</title>
</head>
<body>
<?php
$conn = mysql_connect("localhost",
"tigreton", "anais") or
die(mysql_error());
mysql_select_db("foro", $conn) or
die(mysql_error());
if ($_POST[op] != "añadir respuesta") {
if (!$_GET[respuesta-id]) {
header("Location: listatemas.php");
exit;
}
$verify = "select ft.tema-id,
ft.tema-titulo from
respuesta_foro as fp
left join temas_foro as ft on fp.tema_id =
ft.tema_id where
fp.respuesta-id = $_GET[respuesta-id]";
$verify_res = mysql_query($verify, $conn)
or die(mysql_error());
if (mysql_num_rows($verify_res) < 1) {
header("Location: listatemas.php");
exit;
} else {
$tema_id = mysql_result($verify_res, 0, 'tema_id');
$tema_titulo =
stripslashes(mysql_result($verify_res,
0, 'tema_titulo'));
print "
<table width='90%' border='
'0' cellspacing='2'>
```

```

 cellpadding='2' bgcolor='#999999'>
 <tr bgcolor='CCCCCC'>
 <td>
<div align='center'>
 <font face=
 'Arial, Helvetica, sans-serif' size='5'>
 Envía tu respuesta a $tema_titulo</
font></div>
 </td>
 </tr>
 <tr bgcolor='#FFFFFF'>
 <td>
<form method=post action=
 \$_SERVER[PHP_SELF]\">





```

```

<input type="hidden" name=
  \"tema_id\" value=\"$tema_id\">
</td>
</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td colspan='2'>
<div align='center'>
  <input type="submit" name="submit"
         value="añadir respuesta"></div>
</td>
</tr>
<tr>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
</table>
</form>
</td>
</tr>
</table>

</body>
</html>
";
}
} else if ($_POST[op] == "añadir respuesta") {
if (( !$_POST[tema_id] ) ||
( !$_POST[respuesta_texto] ) ||
( !$_POST[respuesta_creador] )) {
header( "Location: listatemas.php" );
exit;
}
$nu_respuesta = "insert into respuesta_foro
values ('',
'$_POST[tema_id]',
'$_POST[respuesta_texto]', now(),
'$_POST[respuesta_creador]' );
mysql_query($nu_respuesta,$conn) or
die(mysql_error());
header("Location: respuesta.php?tema_id=
Sterna-id");
}

```

```
exit;  
}  
?>
```

#### 15.3.4. Conclusion

He aquí una aplicación sencilla y que podemos integrar con suma facilidad en cualquier página Web. Al no ser más que una extensión de temas ya vistos a lo largo de la presente guía, hemos limitado al máximo los comentarios para cada *script*. Si ha seguido con atención los capítulos anteriores, se encontrará en disposición de realizar cualquier tipo de modificación que necesite para adaptar el presente foro a sus gustos y necesidades.

# Como hacer nuestras aplicaciones seguras

## 16.1. Introducción

Sin duda, el tema de la seguridad es una palabra que siempre esta presente cuando hablamos de Internet. Si bien las ventajas de la "red de redes" son casi infinitas, a nadie escapa las dificultades tecnicas que supone la circulación a diario de datos confidenciales: tarjetas de credito, cuentas corrientes, datos personales, información secreta..., datos que nuestros clientes o visitantes nos comunican, y que un fallo de seguridad puede hacer que caigan en manos ajenas, con el consiguiente perjuicio economico y moral para los que confiaron en nosotros, y el correspondiente descredito de nuestro sitio Web, que indefectiblemente nos lleva a la quiebra o a la perdida absoluta de credibilidad.

Por ello, hemos decidido reservar este ultimo capitulo a estudiar a fondo el tema de la seguridad en Internet, y todos aquellos elementos que hemos de tener en cuenta para blindar nuestro sitio Web.

## 16.2. Autentificacion con PHP y MySQL

Ya en el capitulo 12 vimos un sencillo ejemplo para controlar el acceso de los visitantes a nuestras paginas mediante sesiones. Retomaremos este tema con diversos ejemplos sobre cómo podemos restringir el acceso a nuestras paginas a los usuarios que cumplan determinadas condiciones, o que sepan las contraseñas de acceso que les hemos proporcionado.

## 16.2.1. Sencillo mecanismo de control

A continuación estudiaremos una sencilla aplicación de control de acceso que chequeara nuestro nombre de usuario y contraseña y, en virtud de la respuesta que demos, nos mostrara dos diferentes mensajes: de bienvenida o de acceso restringido.

```
<!-- secreta.php -->
<?
    if(!isset($nombre)&&!isset($contraseña))
    {
?>
    <h1>Introduzca su nombre y contraseña</h1>
    Esta página es secreta
    <form method = post action =
    "secreta.php">
        <table border = 1>
            <tr>
                <th> Nombre de usuario: </th>
                <td> <input type = text name =
                nombre> </td>
            </tr>
            <tr>
                <th> Contraseña </th>
                <td> <input type = password name =
                contraseña> </td>
            </tr>
            <tr>
                <td colspan =2 align = center>
                    <input type = submit value =
                    "Registrarse">
                </td>
            </tr>
        </form>
<?
    }
    else if($nombre=
    ="Ricardo"&&$contraseña=="Maravillas")
    {
        echo "<h1>¡Bienvenido!</h1>" ;
        echo "Ha entrado a una página exclusiva
        para usuarios registrados";
    }
}
```

```
else
{
echo "<h1>Acceso no permitido</h1>";
echo "No esta autorizado a ver la siguiente
página .";
}
?>
```

Estudiémoslo punto por punto. La primera de las pantallas que se nos mostrara es un sencillo formulario en html, donde se nos pedira que introduzcamos nuestro nombre y contraseña.

---

*Nota: Fíjese que en el formulario, en el campo donde se introduce la contraseña, hemos indicado **input type = password**. Esto permitirá que a la hora de escribir ese dato, no aparezca directamente en pantalla y nadie pueda leerla por encima de nuestro hombro.*

---



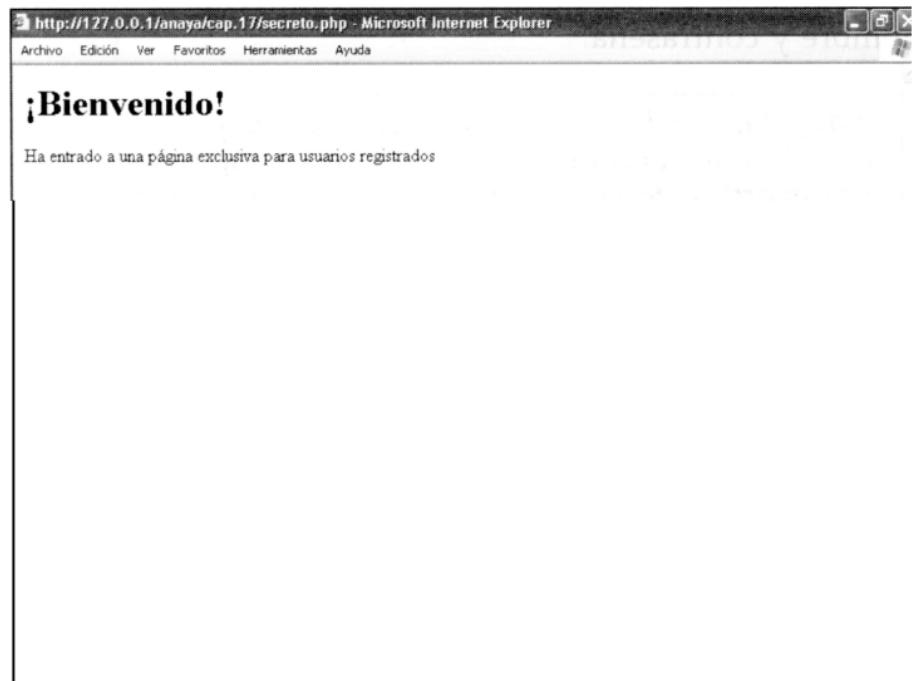
Figura 16.1. Ejemplo de mecanismo de control.

Estos datos son recogidos en forma de variables y enviados via **post** (hemos tratado el tema de formularios extensamente a lo largo de la presente guía, así que no nos extenderemos mas en el tema).

Estos datos son comparados con los que el propio *script* tiene almacenados en su código mediante la siguiente sentencia:

```
else if ($nombre=  
="Ricardo" && $contraseña=="Maravillas")
```

Si tanto el nombre como la contraseña introducidos por el visitante coinciden con estos, pasaremos a la siguiente pantalla.



**Figura 16.2.** Se introdujo la contraseña correcta.

En caso de que dichos datos no coincidan con los que tenemos almacenados, mostrara un mensaje de rechazo, y no podra seguir la navegacion.

El *script* anterior, aunque sumamente sencillo de establecer, presenta un inconveniente muy grande: tanto el nombre como la contraseña estan almacenados dentro del propio *script*, lo cual es un gran inconveniente desde el punto de vista de la seguridad.

Ademas, al ser una contraseña fija, si se la proporcionamos a un usuario autorizado, no podemos impedir que haga circular esta contraseña entre usuarios no autorizados, y que éstos puedan entrar en la pagina sin que podamos impedirlo. Ademas, este *script* solo nos protege una pagina.



**Figura 16.3.** No se introdujo la contraseña correcta.

Para evitar estos problemas, lo ideal es almacenar las distintas contraseñas en una base de datos, y dar a cada usuario una distinta. Así, en caso de que uno de ellos haga mal uso de esa información, solo tendremos que eliminarle de nuestra base de datos.

Para ello, no hay más que introducir una serie de modificaciones en el *script* anterior, mediante el cual se establezca una conexión con la base de datos y compare los datos que nos proporcione el usuario con los que tenemos almacenados en la base de datos. Para ello, utilizaremos el siguiente *script*.

## **16.2.2. Mecanismo de control que almacena las contraseñas en una base de datos**

El primer paso, lógicamente, será crear una base de datos en MySQL. Para simplificar el ejemplo, nos limitaremos a introducir dos campos en dicha base, el nombre de usuario y contraseña.

```
create table seguridad (
  usuario varchar (50),
  contraseña varchar (50)
);
```

A continuación, y a modo de prueba, introduciremos un nombre de usuario y una contraseña (los habituales: Ricardo y Maravillas), mediante la siguiente sentencia:

```
insert into seguridad values
('Ricardo', 'Maravillas')
```

El siguiente paso sera modificar el *script* anterior para que conecte con la base de datos y compruebe los datos que tiene almacenados.

```
<?
    if(!isset($nombre)&&!isset($contraseña))
    {
?>
    <h1>Introduzca su nombre y contraseña</h1>
        <form method = post action =
            "basesecreta.php">
            <table border = 1>
                <tr>
                    <th> Nombre de usuario </th>
                    <td> <input type = text name =
                        nombre> </td>
                </tr>
                <tr>
                    <th> Contraseña </th>
                    <td> <input type = password name =
                        contraseña> </td>
                </tr>
                <tr>
                    <td colspan =2 align = center>
                        <input type = submit value = "Registrarse">
                    </td>
                </tr>
            </form>
?>
    }
    else
    {
        $mysql = mysql_connect( 'localhost',
            'Ricardo', 'Maravillas' );
        if(!$mysql)
        {
            echo 'Imposible seleccionar la base de datos.';
            exit;
        }
    }
}
```

```

}

$mysql = mysql_select_db( 'contacto' );
if( !$mysql)
{
echo 'Imposible seleccionar la base de datos';
exit;
}
$query =
"select count(*) from contacto where
    nombre = '$nombre' and
    contraseña = '$contraseña' ";
$result = mysql_query( $query );
if(!$result)
{
echo 'Imposible efectuar la búsqueda.';
exit;
}
$count = mysql_result( $result, 0, 0 );
if ( $count > 0 )
{
echo "<h1>Bienvenido!</h1>";
echo "La contraseña introducida es correcta.
Bienvenido a la
página correcta";
}
else

echo "<h1>Acceso no autorizado</h1>";
echo "La contraseña introducida no es correcta.
No puede acceder
a la siguiente pagina." ;
}
}

?>

```

Este ejemplo ofrece un nivel de seguridad mucho mayor al almacenar las contraseñas en una base de datos.

### 16.2.3. Mecanismo de autentificación mediante sesiones

Retomaremos en este caso el ejemplo visto en el capítulo 12, de identificación de usuarios mediante sesiones,

introduciendo los cambios necesarios para que retome los valores de una base de datos, y que no vengan directamente en el código.

El primer paso, lógicamente, será crear la base de datos en la que estarán almacenados el nombre de usuario y las contraseñas.

```
create table contacto (
  usuario varchar (50),
  contraseña varchar (50)
);
```

A continuación, presentamos los tres *scripts* que conforman la validación mediante sesiones; esta vez, modificada para recoger la información de la base de datos.

```
<!-- home.php -->
<?
session_start();
if ($usuario && $contraseña)
{
$db_conn = mysql_connect('localhost', 'ricardo',
'maravillas');
mysql_select_db("contacto", $db_conn);
$query = "select * from contacto "
."where nombre='$nombre' "
." and contraseña='$contraseña'";
$result = mysql_query($query, $db_conn);
if (mysql_num_rows($result) >0 )
{
$valid_user = $contraseña;
session_register("valid_user");
}
?
?>
<html>
<body>
      <h1>Página de libre acceso</h1>
      <?
      if (session_is_registered("valid_user"))
      {
echo "Estás registrado, y tu contraseña es:
$valid_user <br>" ;
echo "<a href="
```

```

\"desregistro.php\">Salir</a><br>";
}
else
{
if (isset($contraseña))
{
echo "No te has registrado correctamente";
}
else
{
echo "No estas registrado.<br>";
}
echo "<form method=
post action=\"authmain.php\">" ;
echo "<table";
echo "<tr><td>usuario:</td>";
echo "<td><input type=text name=
usuariosc/&td></tr>" ;
echo "<tr><td>contraseña:</td>";
echo "<td><input type=password name=
contraseña></td></tr>" ;
echo "<tr><td colspan=2 align=center>";
echo "<input type=submit value=
\"Log in\"></td></tr>" ;
echo "</table></form>" ;
}
?>
<br>
<a href="solomiembros.php">Sección
exclusiva para miembros
registrados</a>
</body>
</html>
<!-- desregistro.php -->
<?
session_start;
$old_user = $valid_user;
$result = session_unregister("valid-user");
session_destroy();
?>
<html>
<body>
<h1>Log out</h1>
<?

```

```
if (!empty($old_user))
{
if ($result)
{
echo "Salir del registro.<br>" ;
}
else
{
echo "No ha podido salir del registro.<br>" ;
}
}
else
{
echo "Dado que no se había registrado, no
puede salir del
registro.
<br>" ;

?>
ca href=
"home.php">Vuelta a la pdgina principal</a>
</body>
</html>
<!-- solomiembros.php -->
c?
session_start();
echo "<h1>Sección exclusiva miembros
registrados</h1>" ;
if (session_is_registered("valid_user"))
{
echo "<p>Estás registrado y tu nombre es
$valid-user.</p>" ;
echo "<p>Contenidos exclusivos para vosotros</
p>" ;
}
else
{
echo "<p>No está registrado.</p>" ;
echo "<p>Sólo miembros registrados pueden
acceder a esta página</p>" ;
}
echo "<a href=
\"home.php\">Vuelta a la pdgina principal</a>" ;
?>
```

## 16.3. Encriptacion de contraseñas

Si bien hemos incrementado el nivel de seguridad incluyendo las contraseñas en una base de datos, todavía seguimos corriendo el riesgo de que alguien acceda a ellas. Si lo que estamos almacenando es, por ejemplo, números de tarjetas de crédito, una simple ojeada de un usuario malintencionado bastaría para echar por tierra la reputación de nuestro negocio, y la economía de nuestros clientes.

Para evitarlo, PHP incluye la función `crypt`, que transforma la contraseña que le hemos dado en un algoritmo codificado. Su forma de utilización es la siguiente

```
crypt (contraseña , 'clave')
```

Esta sentencia nos devolverá una palabra codificada, que será la que almacenaremos en la base de datos.

Veámoslo con un ejemplo. Teniendo la contraseña "Maravillas", creamos la clave 'ab', y ponemos en marcha la función `crypt`.

```
crypt (Maravillas, 'ab')
```

Dicha función nos devolverá el texto `cwExBBFWev3I`, el cual no puede ser descifrado de ninguna manera, pero podemos tener la seguridad de que la palabra "Maravillas", utilizando la clave 'ab', siempre dará como resultado `cwExBBFWev3I`, por lo que podemos utilizarlo de la siguiente manera.

En vez de la línea de código siguiente, que verifica el nombre y la contraseña:

```
if($nombre== "Ricardo" && $contraseña== "Maravillas")
```

lo sustituimos por la siguiente línea de código:

```
if($nombre== "Ricardo" && crypt ($contraseña,  
'ab'== " cwExBBFWev3I")
```

De esta sencilla manera, sin saber cuál era la contraseña original, podemos trabajar con ella con la seguridad de que nadie podrá descifrarla.

## 16.4. Proteger multiples páginas

Los ejemplos que hemos visto anteriormente nos sirven para proteger solamente una página. Iremos en la mayor

parte de los casos, lo que realmente necesitaremos es controlar el acceso a un conjunto de páginas determinadas. El tema en este caso se vuelve bastante mas complejo, pero al ser una aplicación necesitada por muchos usuarios, podemos emplear una serie de opciones predefinidas.

### 16.4.1. Autentificación basica

Existe una opción dentro del protocolo HTTP conocida como autentificación basica que podemos utilizar empleando para ello el lenguaje PHP o características propias de nuestro servidor Apache.

Inicialmente, la autentificación basica transmitía el nombre de usuario y la contraseña en formato de texto, lo cual le daba ciertos problemas de seguridad.

Pero en HTTP 1.1 se incluye un algoritmo, conocido como MD5, para proteger estos datos. Aun así, el grado de seguridad dista de ser perfecto, pero presenta la ventaja de su sencillez y su rapidez.

La ventaja fundamental de la autentificación basica es que nos permite proteger directorios enteros con el mismo nombre y contraseña, a diferencia de los ejemplos anteriores, que solo protegían una página.

### 16.4.2. Como utilizar la autentificación basica con PHP

He aquí un ejemplo básico de código en PHP que utiliza la autentificación basica incluida en HTTP, y que funciona en prácticamente todo tipo de servidores:

```
<?
if (substr($SERVER_SOFTWARE, 0, 9) =
= "Microsoft" &&
!isset($PHP_AUTH_USER) &&
!isset($PHP_AUTH_PW) &&
substr($HTTP_AUTHORIZATION, 0, 6) =
= "Basic"
)
{
list($PHP_AUTH_USER, $PHP_AUTH_PW) =
explode(":", base64_decode(substr
($HTTP_AUTHORIZATION, 6)));
}
```

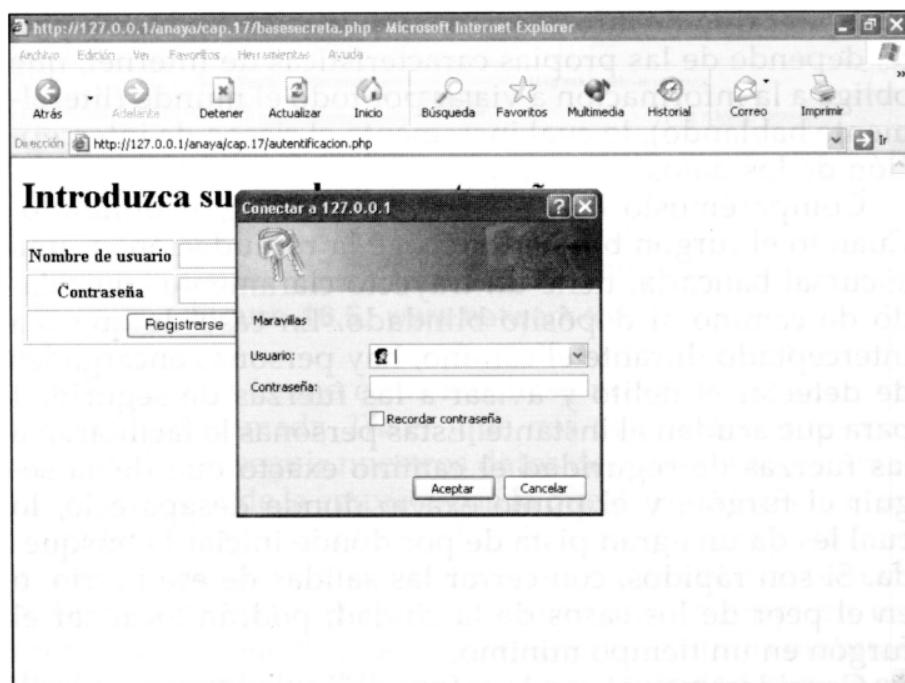
```

}

if ($PHP_AUTH_USER != "Ricardo" || $PHP_AUTH_PW != "Maravillas")
{
header('WWW-Authenticate: Basic realm="Maravillas"');
if (substr($SERVER_SOFTWARE, 0, 9) =
= "Microsoft")
header("Status: 401 Unauthorized");
else
header("HTTP/1.0 401 Unauthorized");
echo "<h1>Acceso no permitido<h1>";
echo "No está autorizado a entrar en esta
web.";
}
else
{
echo "<h1>Bienvenido!</h1>";
echo "<p>Ha entrado usted en la zona
privada.</p>";
}

```

?>



**Figura 16.4.** Ejemplo de autentificacion básica.

Otra ventaja de esta pieza de código es que su interfaz es muy familiar para cualquier usuario acostumbrado a navegar por Internet, y dotara a nuestras páginas de un aspecto más profesional. Introduciendo este *script* en cada archivo del directorio lo protegeremos de miradas indiscretas.

## **16.5. Como establecer una política de seguridad en nuestra pagina Web**

Sea cual sea nuestra presencia en Internet, aunque se trate de algo puramente testimonial, debe estar respaldada por una política de seguridad adecuada que nos asegure la integridad y privacidad de nuestro material. Mas aún: si estamos hablando de una tienda virtual o cualquier tipo de negocio que implique cobro por Internet, debe estar respaldada por una política de seguridad adecuada.

### **16.5.1. Interrupcion en la transmision de datos confidenciales.**

Este es probablemente el mayor riesgo que acecha a cualquier tipo de información confidencial. En buena parte, depende de las propias características de Internet, que obliga a la información a viajar por todo el mundo (literalmente hablando), lo cual incrementa el riesgo de interceptación de los datos.

Comparemoslo con el caso de un furgón blindado. Cuando el furgón blindado recoge la recaudación en una sucursal bancaria, tiene un trayecto claramente controlado de camino al depósito blindado. En caso de que sea interceptado durante el camino, hay personas encargadas de detectar el delito y avisar a las fuerzas de seguridad para que acudan al instante. Estas personas le facilitaran a las fuerzas de seguridad el camino exacto que debía seguir el furgón, y el punto exacto donde desapareció, lo cual les da una gran pista de por donde iniciar la búsqueda. Si son rápidos, con cerrar las salidas de ese barrio, o en el peor de los casos de la ciudad, podrán localizar el furgón en un tiempo mínimo.

Consideremos ahora la información que enviamos vía Internet como un furgón blindado. Si acabamos de reali-

zar una compra en una tienda, y estamos enviando el numero de nuestra tarjeta de crédito o el numero de nuestra cuenta corriente, realmente es como si estuviéramos enviando dinero en metalico.

Y lo peor de todo es que el recorrido de nuestros datos no puede ser predecido ni controlado de antemano, ni mucho menos delimitado a un barrio o ciudad.

El camino que siguen nuestros datos desde que salen del ordenador hasta que llegan a la tienda puede dar varias veces la vuelta al mundo, y rara vez sera el mismo. Veamoslo graficamente ayudandonos de las herramientas que nos proporciona una pagina como [www.network-tools.com](http://www.network-tools.com).

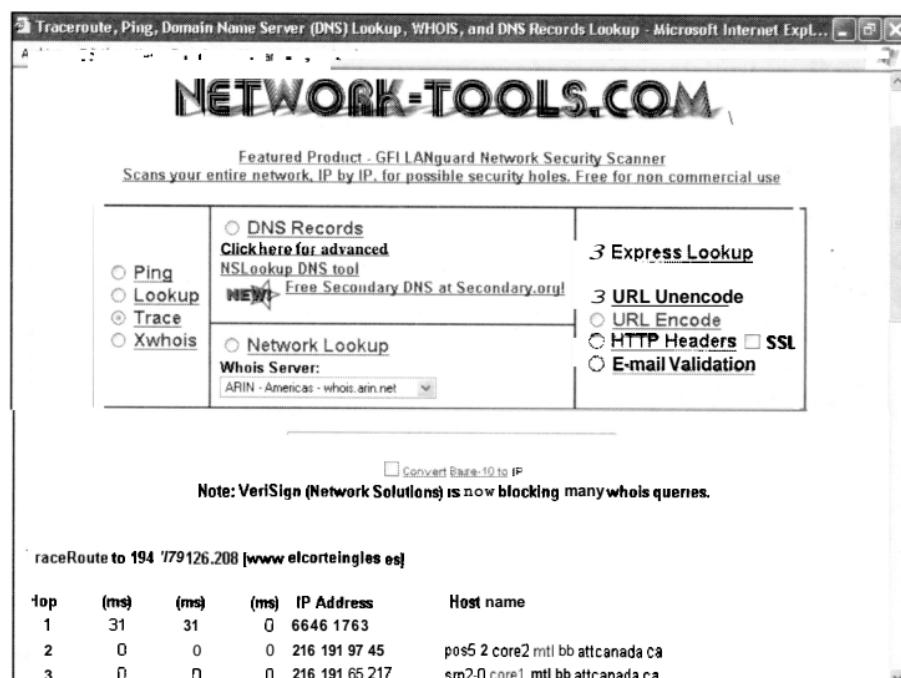


Figura 16.5. [www.network-tools.com](http://www.network-tools.com).

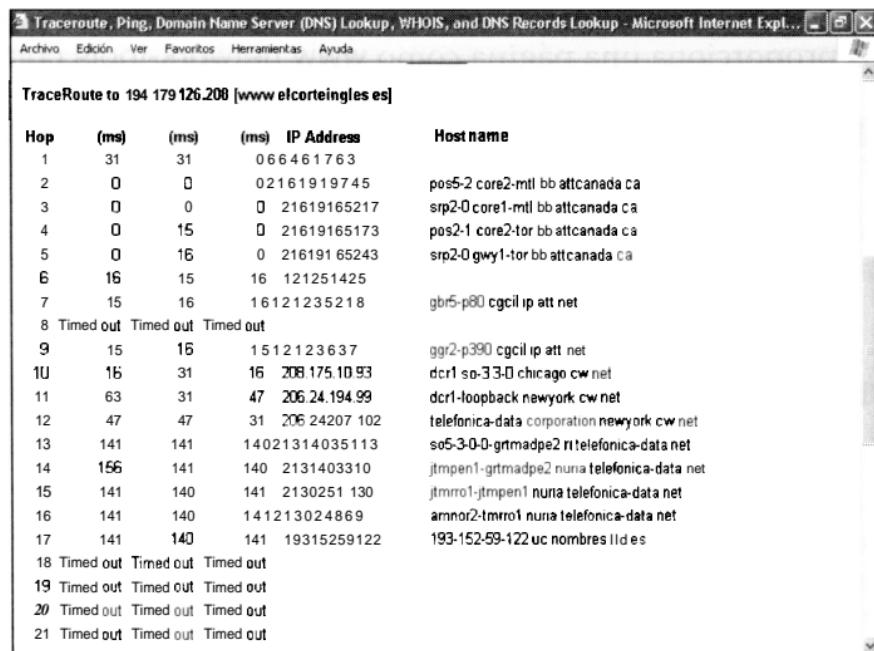
Esta pagina (como otras muchas), nos proporciona una herramienta llamada *Tracer*, que nos calcula todos los pasos que deben seguir nuestros datos desde nuestro ordenador al servidor de la empresa con la que deseamos contactar.

Empecemos con un sencillo ejemplo. En casi todas las grandes ciudades de España hay grandes centros comerciales. En mi caso, si quiero realizar una compra en un centro comercial, apenas tengo que caminar unos minutos desde mi domicilio. Ello me da una seguridad mayor en caso de llevar una gran cantidad de dinero o algún objeto

de valor (lo cual, desgraciadamente, nunca es mi caso), dado que el tiempo que permanecere expuesto a cualquier tipo de sustraccion sera escaso.

Ahora imaginemos que lo que quiero es comprar en dicho centro comercial, pero via Internet. Para saber que camino seguiran mis datos, empleo la herramienta *Tracer*. Introduzco la URL de dicho centro comercial en la página, y ¿qué obtengo?

Algo tan escalofriante como esto:



The screenshot shows a Microsoft Internet Explorer window with the title bar 'Traceroute, Ping, Domain Name Server (DNS) Lookup, WHOIS, and DNS Records Lookup - Microsoft Internet Explorer'. The menu bar includes 'Archivo', 'Edición', 'Ver', 'Favoritos', 'Herramientas', and 'Ayuda'. The main content area displays a table titled 'TraceRoute to 194.179.126.208 [www.elcorteingles.es]'. The table has columns for 'Hop', '(ms)', '(ms)', '(ms)', 'IP Address', and 'Hostname'. The data shows the path from the user's location to the target website, including stops in Canada, the United States, and Spain. Many hops are marked as 'Timed out'.

Hop	(ms)	(ms)	(ms)	IP Address	Hostname
1	31	31	0	0.66.4.61.763	
2	0	0	0.216.19.197.45	pos5-2.core2.mtl.bb.attcanada.ca	
3	0	0	0.216.19.165.217	srp2-0.core1.mtl.bb.attcanada.ca	
4	0	15	0.216.19.165.173	pos2-1.core2-tor.bb.attcanada.ca	
5	0	16	0.216.19.1.65243	srp2-0.gwyl-tor.bb.attcanada.ca	
6	16	15	16.121.251.425		
7	15	16	16.121.213.5218	gbr5-p80.cgcil.ip.att.net	
8	Timed out	Timed out	Timed out		
9	15	16	15.121.123.637	ggr2-p390.cgcil.ip.att.net	
10	16	31	16.208.175.10.93	dcr1.s0-3-0.chicago.cw.net	
11	63	31	47.206.24.194.99	dcr1-loopback.newyork.cw.net	
12	47	47	31.206.24207.102	telefonica-data.corporation.newyork.cw.net	
13	141	141	140.213.140.35113	so5-3-0-0.grtmadpe2.r.telefonica-data.net	
14	156	141	140.213.140.3310	jtmpen1.grtmadpe2.nuna.telefonica-data.net	
15	141	140	141.213.0251.130	jtmpen1.jtmpen1.nuna.telefonica-data.net	
16	141	140	141.213.024869	amnor2-tmro1.nuna.telefonica-data.net	
17	141	140	141.193.152.59.122	193.152.59.122.uc.nombres.ll.es	
18	Timed out	Timed out	Timed out		
19	Timed out	Timed out	Timed out		
20	Timed out	Timed out	Timed out		
21	Timed out	Timed out	Timed out		

**Figura 16.6.** Ejemplo de Tracer.

Este es el camino que siguen mis datos: desde mi domicilio, van directamente a Canada, de allí a Chicago, a Nueva York, vuelta a España..., y se pierde la pista. Ni siquiera una herramienta tan potente puede seguir la pista a la informacion mandada.

Evidentemente, el riesgo de perdida de informacion en semejante viaje es mucho mayor que si hago el viaje a pie. ¿Cómo minimizar este riesgo?

La primera de las cosas que debemos hacer es reducir la cantidad de informacion confidencial que hacemos viajar a traves de la Web. Es decir, no solicitar informacion confidencial a menos que sea estrictamente necesario. Y, realmente, lo estrictamente necesario para una empresa es el cobro.

Por lo tanto, decidido que el único dato sensible a pedir es el del cobro. El siguiente paso será lograr blindar ese proceso. Para ello tenemos dos soluciones:

## Establecer una conexión segura en nuestro servidor (SSL)

Cuando hablamos de conexión segura (*Secure Socket Layer*), tecnología desarrollada por Netscape, y que se ha convertido en el estándar del mercado, consiste en encriptar la información antes de ser enviada, y desencriptada al llegar al servidor. De esta manera nos aseguramos que aunque la información sea interceptada, no se podrá descifrar.

Esta opción presenta una serie de inconvenientes. Por un lado, el proceso de encriptación y desencriptación ralentiza el proceso y reduce el número de conexiones simultáneas en nuestro servidor. Por otro lado, el coste que implica la compra de un certificado digital por parte de una compañía de prestigio, como Verisign ([www.verisign.com](http://www.verisign.com)) o Thawte ([www.thawte.com](http://www.thawte.com)), que verifique que tanto nuestra página como nuestro servidor tienen implementada esta conexión.

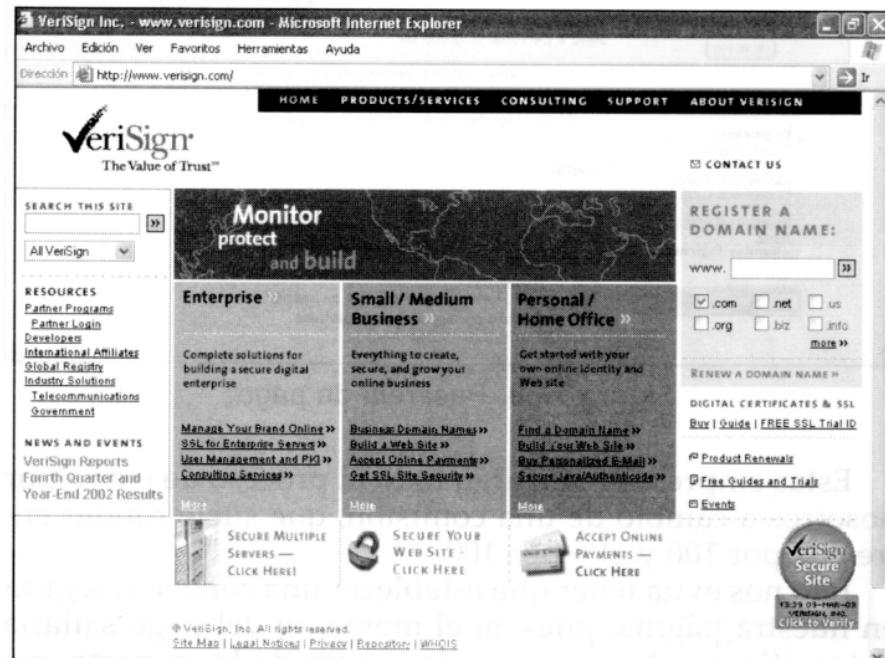


Figura 16.7. Verisign, una de las principales empresas de seguridad informática.

El coste anual de este servicio suele andar sobre los 600 € anuales.

## Confiar todo el tema de nuestros cobros a una empresa especializada

Por muy segura que sea la conexión, si no nos respalda el nombre de una marca de prestigio, el cliente puede dudar de la aplicación que hagamos de sus datos personales. Nuestra ética puede ser intachable, pero es de entender que cueste proporcionar datos confidenciales a unos desconocidos. Por ello, es recomendable delegar el tema de pagos a empresas especializadas, como pueden ser Paypal ([www.paypal.com](http://www.paypal.com)), o 2Checkout ([www.2checkout.com](http://www.2checkout.com)).

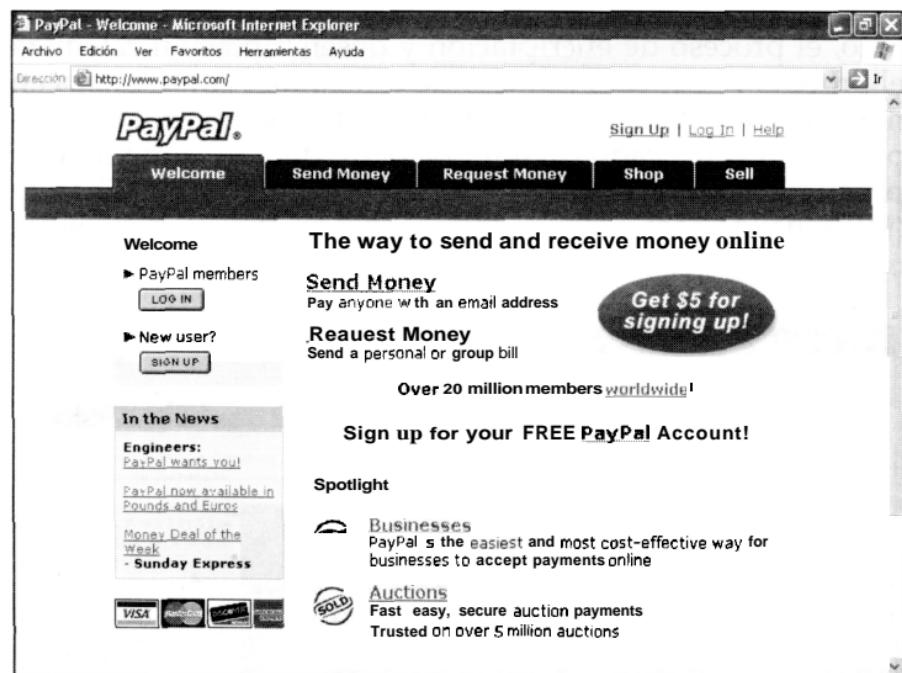


Figura 16.8. Pasarela de pago.

Estas empresas realizaran todo el proceso de cobro por nosotros a cambio de una comisión, que suele rondar entre el 2 por 100 y el 6 por 100.

Ello nos evita tener que establecer una conexión segura en nuestra página, pues en el momento del pago saltaría automáticamente a la pasarela segura de la empresa que hayamos contratado, que se encargara de verificar que los datos de la tarjeta son correctos, de efectuar el cobro y de

ingresarnos el dinero en nuestra cuenta. El cliente tiene la garantía de que sus datos de cobro no llegan a nuestras manos, sino que están en manos de una compañía de prestigio, y a nosotros nos evita trámites engorrosos.

También podemos delegar esta función en nuestro banco, a condición de que tengan establecida una pasarela de pago. Sus comisiones suelen ser un poco menores (¡pero no olvide que siempre serán bancos!), pero actualmente son pocos los bancos que ofrecen una pasarela de pago que realmente funcione y sirva para comercios virtuales de tamaño pequeño y mediano. Además, su aplicación a nuestra página Web será bastante más complicada que las aplicaciones de empresas como Paypal o 2checkout.

## Funciones de PHP que actuan con MySQL

A lo largo del presente libro hemos abordado de una manera practica la manera de crear una pagina Web dinámica utilizando el lenguaje "del lado del servidor" PHP, la base de datos MySQL y el *software* de servidor Apache.

En aras de una comprension mas facil, y del espíritu práctico y de "hagalo usted mismo" que inspira todo el libro, quizá hayamos pasado por alto una vision mas exhaustiva de las sentencias de estos lenguajes.

Para paliar en lo posible esta carencia, incluimos en estos apéndices una lista completa de sentencias en PHP y MySQL, que debe servir al lector que desee profundizar aún mas en el desarrollo de páginas Web dinámicas a aumentar sus conocimientos o, sencillamente, como elemento de consulta al que recurrir cuando tengamos dificultades a la hora de programar.

### A.1. Funciones de PHP que trabajan con MySQL

Lista de todas aquellas funciones incluidas en PHP que nos sirven para interactuar con la base de datos MySQL.

#### **mysql\_affected\_rows**

Versiones que la soportan: PHP 3>= 3.0.6 y PHP 4.

Descripción: int **mysql-affected-rows** (int query-identifier)

Nos devuelve el numero de filas involucradas en una consulta especifica mediante la sentencia **SELECT**; las filas borradas, mediante la sentencia **DELETE**, o actualizadas, mediante la sentencia **UPDATE**.

## **mysql-close**

Versiones que la soportan: PHP 3 y PHP 4.

Cierra la conexión con MySQL.

Descripción: int **mysql-close** (int link-identifier)

Devuelve **TRUE** si tiene éxito, **FALSE** en caso de error.

Cierra la conexión previamente abierta con la base de datos asociada al identificador de conexión (*link identifier*). En caso de no existir identificador de conexión, se cerrará la última conexión abierta.

La mayoría de las veces ni siquiera es necesario incluir esta sentencia, debido a que todas las conexiones no persistentes se cierran automáticamente cuando deja de ejecutarse el *script*.

## **mysql\_connect**

Versiones que la soportan: PHP 3 y PHP 4.

Abre una conexión MySQL.

Descripción: int **mysql-connect** (string hostname)

Devuelve **TRUE** en caso de conexión positiva, y **FALSE** en caso de fallo.

Esta sentencia se emplea para establecer conexiones con servidores MySQL. En caso de que no indiquemos el dominio, se sobreentenderá que se trata de la red local (*localhost*).

Si se emplea dos veces en el mismo *script*, no creará una nueva conexión, sino que nos remitirá a la ya existente. Esta conexión se cerrará cuando termine la ejecución del *script*, o incluyamos la sentencia **mysql-close**.

## **mysql\_create\_db**

Versiones que la soportan: PHP 3 y PHP 4.

Crea una base de datos MySQL.

Descripción: int **mysql\_create\_db** (string database name [, int link-identifier])

Devuelve **TRUE** si tiene éxito, **FALSE** en caso de fallo.

Su función es crear una nueva base de datos en el servidor al que nos referimos a través del identificador de conexión (*link-identifier*).

## **mysql\_data\_seek**

Versiones que la soportan: PHP 3 y PHP 4.

Desplaza el puntero interno de fila.

Descripción: int **mysql\_data\_seek** (int query-identifier, int row-number)

Mueve el puntero de resultados hasta la fila de la base de datos asociada con el identificador de consulta (*query identifier*).

Devuelve TRUE si tiene éxito, FALSE en caso de fallo.

### **mysql\_dbname**

Versiones que la soportan: PHP 3 y PHP 4.

Sirve para obtener el nombre de la base de datos con la que estamos trabajando.

Descripción: string **mysql\_dbname** (int *query-identifier*, int *i*)

Nos devuelve el nombre de la base de datos en la que actualmente se encuentra el puntero *i*.

### **mysql\_drop\_db**

Versiones que la soportan: PHP 3 y PHP 4.

Se utiliza para borrar una base de datos MySQL determinada.

Descripción: int **mysql\_drop\_db** (string *database-name*, int *link-identifier*)

Su función es la eliminar la base de datos cuyo nombre indicamos, asociada al identificador de conexión (*link identifier*), incluyendo todos los datos almacenados en ella.

### **mysql\_error**

Versiones que la soportan: PHP 3 y PHP 4.

Nos indica los mensajes de error asociados a la última operación MySQL que hayamos realizado.

Descripción: string **mysql\_error** ( )

Debido a que la información acerca de errores que proporciona el servidor de la base de datos de MySQL suele ser bastante escasa, podemos recurrir a esta sentencia para recuperar la cadena de caracteres del error.

Los errores que devuelve el servidor de base de datos MySQL no dan mucha información sobre el problema. Por este motivo, utilice estas funciones para recuperar la cadena de caracteres del error.

### **mysql\_fetch\_array**

Versiones que la soportan: PHP 3 y PHP 4.

Recupera el valor de una fila y lo convierte en elemento de una matriz, o *array*.

Descripción: int **mysql\_fetch\_array** (int query-identifier [, int result-type])

Devuelve un *array* o elemento de una matriz, que se corresponde con la fila recuperada, o **FALSE**, si no hay mas filas.

La sentencia **mysql\_fetch\_array()** devuelve la siguiente fila de una busqueda en MySQL y avanza automaticamente el puntero hasta la siguiente fila. Devuelve la fila como una matriz asociativa, una matriz numerica, o ambas, dependiendo del valor de **result-type**.

En definitiva, se trata de una version ampliada de la sentencia **mysql\_fetch\_row()**, con la ventaja de que, sin ser significativamente mas lento que esta ultima, nos proporciona mas informacion.

### **mysql\_fetch\_field**

Versiones que la soportan: PHP 3 y PHP 4.

Nos proporciona informacion sobre una determinada columna de la busqueda.

Descripción: object **mysql\_fetch\_field** (int query-identifier, int field-offset)

Esta sentencia nos devuelve un objeto que contiene information acerca de una determinada columna de nuestra consulta. Si no se especifica el desplazamiento del campo, se recuperara el campo siguiente que no haya sido aun recuperado por la sentencia **mysql\_fetch\_field()**.

Si suponemos que todos los resultados obtenidos mediante esta funcion se almacenan en la variable **!\$field**, podemos analizar las propiedades de cada campo recuperado en la siguiente tabla:

#### *Propiedades*   *Información contenida*

<b>name</b>	Nombre de la columna
<b>table</b>	Nombre de la tabla a la que pertenece la columna
<b>max_length</b>	Máxima longitud de la columna
<b>not-null</b>	Si la columna tiene un valor NOT NULL
<b>primary-key</b>	Si la columna tiene un valor de PRIMARY KEY
<b>unique-key</b>	Si la columna tiene un valor de UNIQUE
<b>multiple-key</b>	Si la columna no tiene un valor de UNIQUE

numeric	Si la columna es numérica
blob	Si la columna es un BLOB
type	El tipo de datos de la columna
unsigned	Si la columna es UNSIGNED
zerofill	Si la columna tiene valor de ZEROFILL

## **mysql\_fetch\_object**

Versiones que la soportan: PHP 3 y PHP 4.

Recupera una fila en forma de objeto.

Descripción: int mysql\_fetch\_object (int query-identifier [, int result-type])

Devuelve un objeto con las propiedades que corresponden a la fila recuperada, o FALSE, si no hay mas filas.

La sentencia **mysql\_fetch\_object()** es similar a la sentencia **mysql\_fetch\_array()**, con una diferencia: se devuelve un objeto en vez de un *array*. Esto significa que solo tiene acceso a los datos por los nombres de los campos, y no por sus desplazamientos.

## **mysql\_fetch\_row**

Versiones que la soportan: PHP 3 y PHP 4.

Devuelve la fila buscada como una matriz numerica.

Descripción: array mysql\_fetch\_row (int query-identifier)

Esta sencilla sentencia se limita a devolver la fila de resultados en forma de matriz numerica. Es idéntica a la sentencia **mysql\_fetch\_array()**, en la que cada columna devuelta se almacena en un desplazamiento de la matriz, comenzando en el desplazamiento 0.

## **mysql\_field\_seek**

Versiones que la soportan: PHP 3 y PHP 4.

Fija la posición del campo, por defecto, para una posterior llamada de la función **mysql\_fetch\_field()**.

Descripción: int mysql\_field\_seek (int query-identifier, int field-offset)

Se posiciona en el desplazamiento de campo especificado (*field offset*). Si la siguiente llamada a la sentencia **mysql\_fetch\_field()** no incluye un desplazamiento de campo, sera este ultimo valor el que se devuelva.

## **mysql\_fieldflags**

Versiones que la soportan: PHP 3 y PHP 4.

Nos muestra el puntero (*flag*) de un determinado campo.

Descripcion: string **mysql\_fieldflags** (int query-identifier, int i)

Esta sentencia nos sirve para obtener los punteros del campo que hayamos especificado. Los valores que podemos obtener son:

- Not null
- Primarykey
- Not null and primary key
- “ ” (Cadena vacia)

## **mysql\_fieldlen**

Versiones que la soportan: PHP 3 y PHP 4.

Nos sirve para obtener la longitud de un determinado campo.

Descripcion: int **mysql\_fieldlen** (int query-identifier,int i)

La sentencia **mysql\_fieldlen()** nos devuelve la longitud del campo especificado.

## **mysql\_fieldname**

Versiones que la soportan: PHP 3 y PHP 4.

Nos facilita el nombre de un determinado campo.

Descripcion: string **mysql\_fieldname** (int query-identifier, int field)

La sentencia **mysql\_fieldname()** devuelve el nombre del campo especificado mediante el identificador de consulta (*query-identifier*).

## **mysql\_fieldtable**

Versiones que la soportan: PHP 3 y PHP 4.

Nos indica el nombre de la tabla que contiene un determinado campo.

Descripcion: int **mysql\_fieldtable** (int query-identifier, int field)

Nos indica el nombre de la tabla que contiene el campo especificado (*intfield*), del resultado previamente especificado por (int query-identifier).

## **mysql\_fieldtype**

Versiones que la soportan: PHP 3 y PHP 4.

Nos indica el tipo de campo.

Descripción: string **mysql\_fieldtype** (int query-identifier, int i)

Realmente, esta sentencia es casi identica a la ya vista **mysql\_fieldname()**, con la diferencia de que ésta nos devolverá el tipo de campo seleccionado, que podrá ser **INT**, **CHAR** o **REAL**.

### **mysql\_free\_result**

Versiones que la soportan: PHP 3 y PHP 4.

Libera la memoria destruyendo todo registro de ese resultado.

Descripción: int **mysql\_free\_result** (int query-identifier)

Esta función libera la zona de memoria asociada con la búsqueda del identificador de consulta (*query-identifier*). Dado que PHP libera la memoria automáticamente cuando se completa una petición, realmente, solo será necesario llamar a esta función cuando se quiera liberar la memoria mientras se ejecuta el *script*.

### **mysql\_list\_dbs**

Versiones que la soportan: PHP 3 y PHP 4.

Nos da una lista de todas las bases de datos MySQL indicadas en el servidor.

Descripción: int **mysql\_list\_dbs** (void)

Devuelve un conjunto de resultados que contiene la lista de todas las bases de datos que hay disponibles en el servidor.

### **mysql\_list\_fields**

Versiones que la soportan: PHP 3 y PHP 4.

Devuelve todos los resultados contenidos en la lista de tablas de la base de datos específica.

Descripción: int **mysql\_list\_fields** (string database, string tablename)

La sentencia **mysql\_list\_fields()** recupera información sobre el nombre de tabla dado. Los argumentos son el nombre de la base de datos (*database name*) y el nombre de la tabla (*tablename*). Se devuelve un puntero al resultado. La función devuelve -1 si ocurre un error.

### **mysql\_list\_tables**

Versiones que la soportan: PHP 3 y PHP 4.

Lista las tablas de una base de datos MySQL.

Descripcion: int **mysql\_list\_tables** (string database)

Devuelve un conjunto de resultados que contiene una lista de las tablas existentes en la base de datos. Podemos usar la sentencia **mysql\_tablename()** para recuperar los nombres reales de cada tabla.

## **mysql\_num\_fields**

Versiones que la soportan: PHP 3 y PHP 4.

Devuelve el numero de campos de una busqueda en MySQL.

Descripcion: int **mysql\_num\_fields** (int query-identifier)

La sentencia **mysql\_num\_fields()** devuelve el numero de campos de la busqueda que hemos realizado en MySQL.

## **mysql\_num\_rows**

Versiones que la soportan: PHP 3 y PHP 4.

Devuelve el numero de filas en una busqueda en MySQL.

Descripcion: int **mysql\_num\_rows** (int query-identifier)

La sentencia **mysql\_num\_rows()** devuelve el numero de filas implicadas en una busqueda realizada en MySQL.

## **mysql\_pconnect**

Versiones que la soportan: PHP 3 y PHP 4.

Crea una conexion MySQL persistente.

Descripcion: int **mysql\_pconnect** (string hostname)

Crea una conexion persistente con un servidor MySQL.

Su funcionamiento es muy similar a la sentencia **mysql\_connect()**, con la diferencia de que la conexion no puede ser cerrada mediante la sentencia **mysql\_close()**. A este tipo de conexiones se les llama "persistentes".

## **mysql\_query**

Versiones que la soportan: PHP 3 y PHP 4.

Realiza una consulta a la base de datos.

Descripcion: int **mysql\_query** ( string query, int link-identifier)

Realiza una consulta a la base de datos del servidor MySQL asociada con el identificador de conexion (*link identifier*). Es casi identica a la sentencia **mysql\_db\_query()**, con la excepcion de que no cambia la base de datos para la busqueda.

En caso de exito, devuelve un identificador de consulta MySQL positivo, o **FALSE** en caso de error.

## **mysql\_regcase**

Versiones que la soportan: PHP 3 y PHP 4.

Construye una expresion regular para una busqueda que no distinga entre mayusculas y minusculas.

## **mysql\_result**

Versiones que la soportan: PHP 3 y PHP 4.

Obtiene el valor de un campo especifico en una fila determinada.

Descripcion: int **mysql\_result** (int query-identifier, int i, mixed field)

Devuelve el contenido de la celda en la fila y desplazamiento del conjunto resultado MySQL especificado.

Nos retorna el valor de un determinado campo en la fila especificada. Si el nombre de la fila tiene un alias, debemos sustituir el nombre de la columna por la del alias que le hemos aplicado.

## **mysql\_select\_db**

Versiones que la soportan: PHP 3 y PHP 4.

Selecciona una base de datos determinada de nuestro servidor MySQL.

Descripcion: int **mysql\_select\_db** (string database\_name, int link-identifier)

Devuelve **TRUE** si tiene exito, **FALSE** en caso contrario.

Establece conexion con la base de datos que tengamos activa, asociada al identificador de conexion (*link-identifier*). En caso de no especificar este identificador de conexion, se limitara a utilizar la ultima conexion abierta.

## **mysql\_tablename**

Versiones que la soportan: PHP 3 y PHP 4.

Obtiene el nombre de la tabla de un campo

Descripcion: string **mysql\_tablename** (int query\_identifier, int field)

Tomando el valor devuelto mediante la utilizacion de la sentencia **mysql\_list\_tables()**, devolvera el nombre de la tabla listado en la fila especificada mediante el campo especificado (*intfield*), empezando por la fila cero.

# Tipos de columnas en MySQL

A lo largo del presente libro, hemos utilizado las bases de datos MySQL para almacenar diversos tipos de datos, ya sea fechas, cadenas de texto, numeros, direcciones de correo electronico, etc.

Con ello, ha quedado claro que estas bases de datos pueden almacenar tipos de datos muy diversos, incluyendo imagenes, musica, video, etc.

De cara al usuario avanzado que quiere seguir estudiando la creación de páginas Web dinamicas, incluimos este anexo, en el que profundizamos en los diferentes tipos de datos que pueden almacenarse en una base de datos MySQL, y en los diferentes tipos de columnas que podemos crear para hacer el almacenamiento mas efectivo.

Dado que muchas de estas columnas pueden tener parametros opcionales, estableceremos una serie de convenciones a la hora de explicar las diferentes columnas.

## Parametros

**M.** Este parametro se utiliza para indicar el numero maximo de caracteres que pueden tener los valores que incluyamos en esta columna. M puede ser cualquier numero entero entre 1 y 255. El almacenamiento de un numero de caracteres mayor, indefectiblemente, nos acabara causando problemas, sobre todo cuando las relaciones entre las distintas bases de datos sean mas complejas.

**D.** Este parametro nos permite especificar cuantos numeros decimales pueden ser almacenados en valores de punto flotante. El valor maximo de este parametro es 30, siempre y cuando el parametro M nos lo permita.

## Atributos

**ZEROFILL.** Esta opcion consigue que la columna siempre ocupe su maxima longitud, dado que en el caso de que no le asignemos ningun valor, el sistema automaticamente lo completara con ceros. De esta manera nos aseguramos de que en la busqueda siempre encontraremos un valor, aunque sea cero. Al activar esta opcion, automaticamente se activa la opcion **UNSIGNED**.

**UNSIGNED.** Esta opcion consigue que la columna solo acepte valores positivos, o cero. Hay que tener precaucion y, antes de activarla, tener completamente seguro que no queremos aceptar valores negativos.

**BINARY.** Por defecto, los caracteres de comparacion en MySQL no distinguen entre mayusculas y minusculas. Sin embargo, los caracteres de comparacion en columnas **BINARY** si admiten mayusculas y minusculas.

## Valores numericos

**TINYINT[(M)] [UNSIGNED] [ZEROFILL]**

Descripcion: diminuto valor entero.

Rango: de -128 a 127 (o de 0 a 255, si activamos la funcion **UNSIGNED**).

**SMALLINT[(M)] [UNSIGNED] [ZEROFILL]**

Descripcion: diminuto valor entero.

Rango: de -32768 a 32767 (0 a 65535, si activamos la funcion **UNSIGNED**).

**MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]**

Descripcion: entero de tamaño medio.

Rango: de -8388608 a 8388607 (0 a 16777215, si activamos la funcion **UNSIGNED**).

**INT[(M)] [UNSIGNED] [ZEROFILL]**

Descripcion: numero entero de tamaño normal.

Rango: de -2147483648 a 2147483647. (0 a 4294967295, si activamos la funcion **UNSIGNED**).

**INTEGER[(M)] [UNSIGNED] [ZEROFILL]**

Descripcion: es un sinonimo de INT.

**BIGINT[(M)] [UNSIGNED] [ZEROFILL]**

Descripcion: numero entero de gran tamaiio.

Rango: de -9223372036854775808a 9223372036854775807. (0 a 18446744073709551615, si activamos la funcion UNSIGNED).

**FLOAT(precision) [UNSIGNED] [ZEROFILL]**

Descripcion: numero de punto flotante.

Rango: de 0 a +/- 1.175494351E - 38 hasta +/- 3.402823466E+38.

**Nota:** Si especificamos el parámetro precision (en bits), debe ser menor o igual a 24, pues de otra manera se creara una columna DOUBLE.

**DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]**

Descripcion: numero de punto flotante de gran precision.

Rango:

de 1.7976931348623157E+308 a -2.2250738585072014E-308, 0, y de 2.2250738585072014E-308 hasta 1.7976931348623157E+308.

**Nota:** La precision (en bits), si se especifica, debe ser mayor o igual a 25; o si no, se creara una columna FLOAT. El valor de precision no debe ser mayor de 53.

También se puede definir como DOUBLE PRECISION [(M, D)] o REAL[(M,D)].

**DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]**

Descripcion: un numero de punto flotante que puede almacenarse como una cadena de texto.

Rango: como para DOUBLE, pero ateniéndose a los parametros M y D.

**Nota:** Si no especificamos el valor de D, por defecto, sera igual a cero, y los números en esta columna no tendrán parte decimal. Si no especificamos el valor de M, su valor sera igual a 10.

**DEC[(M[,D])] [UNSIGNED] [ZEROFILL]**

Descripcion: idéntico a decimal.

**NUMERIC[ (M[ ,D] ) ] [UNSIGNED] [ZEROPILL]**

Descripcion: identico a decimal.

## Valores de fecha y tiempo

### DATE

Descripcion: una fecha.

Rango: de '1000-01-01' hasta '9999-12-31'.

---

**Nota:** El valor en que las fechas son almacenadas en las tablas MySQL corresponde a 'aaaa-mm-dd' (año-mes-día).

---

### DATETIME

Descripcion: combinación de fecha y hora.

Rango: de '1000-01-01 00:00:00' hasta '9999-12-31 23:59:59'.

---

**Nota:** El valor en que las fechas son almacenadas en las tablas MySQL corresponde a 'AAAA-MM-DD HH:MM:SS'

---

### TIMESTAMP[ (M) ]

Descripcion: almacena la fecha y la hora en un momento determinado.

Rango: Desde '1970-01-01 00:00:00' a algun momento en el año 2037.

---

**Nota:** Una columna con formato TIMESTAMP tiene gran utilidad para almacenar la fecha y la hora de una operación de INSERT o UPDATE porque, automáticamente, fija la fecha y la hora de la operación más reciente si no le damos un valor nosotros mismos. También podemos fijar la fecha y la hora actual, asignándole un valor NULL.

---

### TIME

Descripcion: una hora.

Rango: desde '-838:59:59' hasta '838:59:59'

---

**Nota:** El valor en que las horas son almacenadas en las tablas MySQL corresponde a 'HH:MM:SS'.

---

## YEAR [ (2|4) ]

Descripcion: un año, en version de dos o cuatro digitos (el valor por defecto es cuatro).

Rango: desde 1901 hasta 2155 en version de cuatro digitos, y desde 1970 hasta 2069 en version de dos digitos.

## Valores de caracteres

### CHAR (M) [BINARY]

Descripcion: una cadena de caracteres de longitud fija.

Longitud máxima: M caracteres.

---

**Nota:** Las cadenas de caracteres son almacenadas hasta la longitud máxima indicada de M (de 0 a 255 caracteres). Los espacios infermedios son eliminados cuando se supera este valor. Asimismo, si no ocupamos el valor completo de M, automaticamente se almacenaran espacios al final de la cadena de texto hasta completar esta longitud.

---

### VARCHAR (M) [BINARY]

Descripcion: una cadena de caracteres de longitud variable.

Longitud maxima: M caracteres.

---

**Nota:** Los espacios al principio y al final de la cadena de texto son eliminados a la hora del almacenamiento. Dado que es una cadena de longitud variable, no es necesario consumir el máximo de espacio permitido.

---

### TINYBLOB

### TINYTEXT

Descripcion: una pequeña cadena de texto de longitud variable.

Longitud maxima: 255 caracteres.

---

**Nota:** Muy similar a VARCHAR; sin embargo, en este caso no se borran los caracteres del principio y del final de las líneas.

---

**BLOB**

**TEXT**

Descripción: una cadena de texto de longitud variable.  
Longitud máxima: 65535 caracteres.

**MEDIUMBLOB**

**MEDIUMTEXT**

Descripción: una cadena de texto de tamaño medio y longitud variable.

Longitud máxima: 16777215 caracteres.

**LONGBLOB**

**LONGTEXT**

Descripción: una cadena de texto de gran tamaño y longitud variable.

Longitud máxima: 4294967295 caracteres.

**ENUM('valor1','valor2',...)**

Descripción.: una enumeración de valores de los cuales uno ha de ser escogido de cada fila.

Longitud máxima: puede contener hasta un total de 65535 valores distintos.

---

*Nota: Los valores en este tipo de campos son almacenados como números enteros que representan el elemento seleccionado. 1 representa el primer elemento; 2, el segundo; y así sucesivamente. El valor 0 representa una cadena vacía, es decir, que no seleccionamos ninguno de los valores que se nos ofrecen.*

---

**SET('valor1','valor2',...)**

Descripción: un conjunto de valores, cada uno de los cuales puede ser seleccionado o no.

Longitud máxima: podemos dar hasta 64 valores para seleccionar.

# Guía de referencia rápida de HTML

A lo largo de la presente guía, hemos hecho numerosas referencias al lenguaje HTML. Como ya sabemos, una de las características de PHP es su capacidad para embeberse dentro de una página escrita en HTML e incrementar sus posibilidades de manera exponencial. Podríamos decir, incluso, que una página en HTML es el lugar donde un *script* en PHP se siente más a gusto.

Por ello, hemos creído imprescindible incluir en nuestra guía una referencia rápida que incluya las principales etiquetas (*tags*) de este lenguaje, para que los principiantes tengan un punto de apoyo, y los usuarios avanzados puedan consultar algunas dudas puntuales.

## C.1. Etiquetas de HTML

### Etiquetas básicas

`<html></html>`

Crea un documento de HTML.

`<head></head>`

Indicamos el nombre del documento. Esta información no aparecerá en el documento.

`<body></body>`

Indica el texto que conforma el documento.

**<title></title>**

Pone el nombre del documento en la barra de titulo.

## **Etiquetas de atributos**

**cbody bgcolor=?>**

Designa el color del fondo de la pagina, usando el nombre del color en ingles o hexadecimal.

**cbody text=?>**

Designa el color del texto, usando el nombre del color en ingles o hexadecimal.

**cbody link=?>**

Designa el color de los *links*, usando el nombre del color en inglés o hexadecimal.

**cbody vlink=?>**

Designa el color de los *links* visitados, usando el nombre del color en ingles o hexadecimal.

**cbody alink=?>**

Designa el color de los *links* en *click*, usando el nombre del color en ingles o hexadecimal.

## **Etiquetas de texto**

**<pre></pre>**

Crea texto preformatado.

**<h1></h1>**

Crea el encabezado mas grande.

**<h6></h6>**

Crea el encabezado mas pequeño.

**<b></b>**

Crea el texto en "**negrita**".

**<i></i>**

Crea texto en *"cursiva"*.

**<tt></tt>**

Crea texto teletipo, o maquina de escribir.

**<cite></cite>**

Crea una cita, usualmente en cursivas.

**<em></em>**

Da enfasis a palabras (con cursivas).

**<strong></strong>**

Da enfasis a palabras (con negritas).

**<font size=?></font>**

Indicamos el tamaño de las letras o fuentes (de 1 a 7).

**<font color=?></font>**

Le da color a la fuente, usando el nombre del color en ingles o hexadecimal.

## **Etiquetas de *links***

**<a href=>URL</a>**

Crea un *hyperlink* a una URL determinada.

**<a href=>mailto:EMAIL</a>**

Crea un *link* que nos sirve para mandar un correo electronico.

**<a name="NAME"></a>**

Crea el objetivo de un *link* en un documento.

**<a href=>#NAME</a>**

Enlaza a un objetivo dentro de un documento.

## Etiquetas de formato

**<p></p>**

Crea un nuevo parrafo.

**<p align=?>**

Alinea un parrafo: a la izquierda (*left*), derecha (*right*), o al centro (*center*).

**<br>**

Inserta un salto de linea.

**<blockquote></blockquote>**

Crea una sangria en ambos lados del texto.

**<dl></dl>**

Crea una lista de definiciones.

\*

**<dt>**

Precede a cada termino a definir.

**<dd>**

Precede a cada definición.

**<ol></ol>**

Crea una lista numerada.

**<li></li>**

Preceda a cada termino de la lista, y añade un numero.

**<ul></ul>**

Crea una lista punteada.

**<div align=?>**

Un *tag* genérico usado para dar formato a un bloque grande de HTML, también usado para los *stylesheets*.

## **Etiquetas de elementos graficos**

**<img src=>name»>**

Aiiade una imagen.

**<img src=>name» align=?>**

Alinea una imagen: a la izquierda (*left*), derecha (*right*), centro (*center*);abajo (*bottom*),arriba (*top*),en medio (*middle*).

**<img src=>name» border=?>**

Asigna el tamaño del borde de una imagen.

**<hr>**

Inserta una linea horizontal.

**<hr size=?>**

Asigna el tamaño de la linea.

**<hr width=?>**

Asigna el ancho de la linea, en porcentaje o en valor absoluto.

**<hr noshade>**

Crea una linea sin sombra.

## **Etiquetas de tablas**

**<table></table>**

Crea una a tabla.

**<tr></tr>**

Crea cada linea en una tabla.

**<td></td>**

Crea cada columna en una tabla.

**<th></th>**

Crea cada encabezamiento en una tabla.

## Atributos de tabla

**ctable border=#>**

Crea el ancho del borde de la tabla.

**ctable cellspacing=#>**

Crea un espacio entre las columnas de las tablas.

**<table cellpadding=#>**

Crea un espacio entre los marcos de las columnas y su contenido.

**ctable width=# or %>**

Asigna el ancho de la tabla bien en pixeles, o bien en porcentaje del ancho del documento.

**<tr align=?> or <td align=?>**

Alinea las columnas: a la izquierda (left), derecha (right), o al centro (center).

**ctr valign=?> or ctd valign=?>**

Alinea verticalmente las columnas: arriba (top), en medio (middle), o abajo (bottom).

**ctd colspan=#>**

Indica el numero de columnas que abarca una celda.

**ctd rowspan=#>**

Indica el numero de columnas que abarca una celda (*default=1*).

**ctd nowrap>**

Previene que se rompa el texto dentro de una celda.

## Frames

**<frameset></frameset>**

Define un documento con frames; tambien puede ser incluido dentro de otros *framesets*.

```
<frameset rows=>value,value>>
```

Define que los documentos dentro de un *frameset* serán acomodados en líneas horizontales. El tamaño de las líneas se determina bien en pixeles, o bien en porcentajes.

```
<frameset cols=>value,value>>
```

Define que los documentos dentro de un *frameset* serán acomodados en columnas. El tamaño de las columnas se determina en pixeles o porcentajes.

```
<frame>
```

Define un frame dentro de un frameset.

```
<noframes></noframes>
```

Define que aparecerá en un navegador que no soporte frames.

## Atributos de los *Frames*

```
<frame src=>URL>>
```

Especifica el documento HTML del *frame*.

```
<frame name=>name>>
```

Define nombre o region, para que pueda ser objetivo de otro frame.

```
<frame marginwidth=#>
```

Define los márgenes derecho e izquierdo de un frame; debe ser mayor o igual a 1.

```
<frame marginheight=#>
```

Define los márgenes de arriba y abajo de un frame; debe ser mayor o igual a 1.

```
<frame scrolling=VALUE>
```

Indica si el frame tiene una barra de desplazamiento. Los valores pueden ser yes (si), no (no), o auto (el navegador lo determina automáticamente). El valor por omisión, como en los documentos ordinarios, es auto.

```
<frame noresize>
```

Previene que el usuario cambie de tamaño el *frame*.

## Formularios

```
<form></form>
```

Se utiliza para crear el formulario propiamente dicho.

```
<select multiple name=>NAME> size=?></select>
```

Crea un menu con barra de desplazamiento. El tamaño asigna el numero de opciones que se pueden ver al mismo tiempo sin mover la barra.

```
<options
```

Defina cada una de las opciones dentro del menu.

```
<select name=>NAME>></select>
```

Crea un menu que ofrece una serie de opciones para elegir.

```
<textarea name=>NAME> cols=40 rows=8></textarea>
```

Crea un area de texto. La cantidad de columnas determinan el ancho, la cantidad de las filas (*rows*)determinan la altura.

```
<input type="checkbox" name="NAME">
```

Crea un *checkbox*. El texto sigue al *tug*.

```
<input type="radio" name="NAME" value="x">
```

Crea un boton de radio. El texto sigue al *tug*.

```
<input type="text" name="foo" size=20>
```

Crea un area de texto de una linea. El tamaño crea el largo en caracteres.

```
<input type="submit" value="NAME">
```

Crea un boton de envio.

```
<input type="image" border=0 name= "NAME" src= "name .gif">
```

Crea un boton de envio usando una imagen.

```
<input type=>reset>>
```

Crea un boton para reestablecer los datos de la forma a su estado original.

# Índice alfabetico

## A

AddSlashes; 94  
All; 105  
Alter; 104  
Amazon; 18  
Anaya Multimedia; 109  
Apache; 26, 29, 30, 31, 35,  
135,140,148,152  
ASCII; 152  
**ASP**; 43  
Atkinson, James; 161  
Autentificacion; 179,245,  
250,251

## B

Base de datos; 57, 59, 60, 75,  
76,107,117,213,219,243

## C

C, lenguaje; 51  
C++, lenguaje; 44, 51  
Chop; 90  
Content Management  
System; 135  
Cookies; 21, 27, 169, 171, 185  
Count; 67,68  
Cracker; 99  
Create; 69, 103, 104

## D

Date; 44, 63  
Dbm; 27  
Delete; **104**  
Describe; 63, 64, 69  
Dmoz; 32,142

## **E**

El Corte Ingles; 18  
El País; 24  
Elvis; 45  
Encriptacion; 249,250  
eMule; 185,186,187, 188,  
193,194  
Ereg; 95, 96  
Ereg\_replace; 96  
Eregi; 95, 96  
Eregi\_replace; 96

## **F**

Ferrigno, Lou; 199  
FilePro; 27  
Flash; 23  
Formularios; 26, 40, 152, 163,  
207,241  
FTP (File Transfer Protocol);  
26,140,152,163

## **G**

Gacik, Milan; 33  
GET; 80  
GIF; 142  
Google; 18, 21, 32, 142  
GPL, Licencia; 32, 33, 140  
Grant; 105  
Gutsman, Andy; 28  
Gzip, (formato de compre-  
sion); 101

## **H**

HTML; 22, 23, 24, 26, 30, 42,  
44, 45, 48, 71, 81, 90, 91  
HTTP; 169

## **I**

If; 51,174  
Include; 125,126  
Interbasem; 27  
Internet; 15, 16, 239  
Internet Explorer; 71  
Insert; 65, 66, 67, 69, 104

## **J**

Java; 51,197

## **L**

Lerdorf, Rasmus; 28  
Linux; 27, 33, 139, 152, 162  
Ltrim(); 90

## **M**

Microsoft; 29, 30, 152  
Microsoft Access; 162  
Microsoft SQL  
Server; 162  
Murphy, ley de; 103  
MySQL; 17, 18, 25, 26, 28, 97,  
111, 135, 146, 167, 197,  
216  
Mysql\_connect; 72, 73, 74  
Mysql\_fetch\_array; 76  
Mysql\_pconnect; 73  
Mysql\_query; 75, 76, 80  
Mysql\_select\_db; 75

## **N**

Napster; 185, 186  
Netscape; 71  
NL2br; 90

## O

ODBC; 27  
ODP; 32  
Open Source; 21, 28, 147,  
161,186  
Operadores; 47  
Operadores aritmeticos; 47  
Operadores de texto; 47  
Oracle; 27  
OsCommerce;  
21,147,148,149,154,160

## P

Perl; 51  
PHP; 17, 18, 20, 21, 25, 26, 28,  
94, 113, 125, 135, 146, 167,  
185,193,197  
phpBB; 21,161,163,165,  
194  
PHP Home Edition 2; 33,135  
phpMyadmin; 36, 37, 59, 84,  
100,110  
PHP-Nuke; 137  
Platón, 200  
Ponce de Leon, Harald; 148  
POST; 50, 51, 113, 130, 133  
Postgre\_SQL; 27,162  
PostNuke; 21, 135, 136, 137,  
138,139,141,145,163  
Programacion Orientada a  
Objetos (POO); 197,198,  
199,200  
Puerto; 35,192

## R

Require; 125,126  
Revoke; 105

## S

Script; 39, 45, 46, 51, 55, 78,  
81, 95, 112,117,125, 153,  
210,211,221,234,237  
Seguridad; 100  
Select; 65, 66, 67, 68, 69, 104  
Sendmail; 174  
Sesiones; 27, 169, 174, 175,  
176,185,245  
Show; 63, 69  
Siglo de Oro Espaiiol; 58  
Solaris; 27  
SQL; 60, 62, 63, 66, 68, 71, 84,  
94, 100, 102, 110,217,219,  
220,243  
SSL; 150,255  
StripSlashes; 94  
Strtolower; 92, 93  
Strtoupper; 92, 93  
Suraski, Zeev; 28

## T

TCP/IP, protocolo; 16, 192  
Terra; 24  
The New York Times; 24  
Trim; 90

## U

Ucfirst; 92, 93  
Ucwords; 92, 93  
UNIX; 27, 29, 30, 39  
Update; 104  
Url; 50,153

## V

Variables; 46

**W**

Web; 26, 89, 95, 161, 167, 237  
WinMySQLAdmin; 38  
Windows; 29, 30, 41, 101, 139

**X**

XML; 43

**Y**

Yahoo; 142

**Z**

Zend; 28  
Zip (formato de compren-  
sion); 101,162