Contenido

[Unidad 1: WampServer – Guía de descarga e instalación. 1](#_Toc440136386)

[Unidad 2: El lenguaje de programación del lado del servidor PHP. 4](#_Toc440136387)

[2.1. ¿Qué es PHP? 4](#_Toc440136388)

[2.2. Características del lenguaje 4](#_Toc440136389)

[2.2.1. Características generales de PHP 4](#_Toc440136390)

[2.2.2. Impresión de datos 5](#_Toc440136391)

[2.3. Utilización de variables en PHP 6](#_Toc440136392)

[2.3.1. Variables predefinidas 6](#_Toc440136393)

[2.3.2. Variables locales, globales y superglobales 6](#_Toc440136394)

[2.3.3. Concatenación de cadenas de texto 7](#_Toc440136395)

[2.3.4. Adición de más texto a una variable ya existente 8](#_Toc440136396)

[2.3.5. Utilización y manipulación de *arrays* 9](#_Toc440136397)

[2.3.5.1. Recorriendo los elementos de un *array* 10](#_Toc440136398)

[Unidad 3: El servidor de bases de datos MySQL 17](#_Toc440136399)

[3.1. Organización de MySQL 17](#_Toc440136400)

[3.1.1. Columnas 17](#_Toc440136401)

[3.1.1.1. Tipo Numérico 17](#_Toc440136402)

[3.1.1.2. Tipo Fecha 19](#_Toc440136403)

[3.1.1.3. Tipo Cadena 20](#_Toc440136404)

[3.1.2. Datos 21](#_Toc440136405)

[3.1.3. Índices 22](#_Toc440136406)

[3.2. Manipulación y utilización de MySQL 22](#_Toc440136407)

[3.2.1. Creación de una base de datos 23](#_Toc440136408)

[3.2.2. Creación de una tabla para la base de datos 24](#_Toc440136409)

[3.2.3. Inserción de información 28](#_Toc440136410)

[3.2.4. Selección de registros de la tabla 29](#_Toc440136411)

[3.2.5. Modificación de registros almacenados en una tabla 33](#_Toc440136412)

[3.2.6. Eliminación de registros almacenados en la tabla 36](#_Toc440136413)

[Unidad 4: MySQL y PHP. 37](#_Toc440136414)

[4.1. Ficheros de configuración y acciones repetitivas 37](#_Toc440136415)

[4.2. Obtener siempre una respuesta 38](#_Toc440136416)

[4.3. Uso de MySQL con PHP 38](#_Toc440136417)

[4.3.1. Establecer una conexión con MySQL 38](#_Toc440136418)

[4.3.2. Selección de la base de datos 39](#_Toc440136419)

[4.3.3. Creación de los ficheros de configuración 39](#_Toc440136420)

[4.3.4. Realización de consultas a MySQL 41](#_Toc440136421)

[4.3.5. Impresión de resultados 41](#_Toc440136422)

[4.3.5.1. Impresión de resultados usando un *array* 43](#_Toc440136423)

[4.3.5.2. Impresión de resultados usando un *array* de lista 44](#_Toc440136424)

[4.3.5.3. Impresión de resultados usando un objeto 45](#_Toc440136425)

[4.3.6. Liberación de memoria 48](#_Toc440136426)

[4.3.7. Cierre de la conexión 49](#_Toc440136427)

[4.4. Modificación de la información de la base de datos 49](#_Toc440136428)

[4.4.1. Envío de variables usando POST y GET 50](#_Toc440136429)

[4.4.1.1. Envío de variables mediante GET 50](#_Toc440136430)

[4.4.1.2. Envío de variables mediante POST 53](#_Toc440136431)

[4.4.2. Bootstrap 54](#_Toc440136432)

[4.4.3. Listado de registros 55](#_Toc440136433)

[4.4.4. Obtención de detalles de un registro 60](#_Toc440136434)

[4.4.5. Modificación de un registro 65](#_Toc440136435)

[4.4.6. Inserción de un nuevo registro 75](#_Toc440136436)

[4.4.7. Borrado de un registro específico 80](#_Toc440136437)

[Bibliografía: 85](#_Toc440136438)

# Unidad 1: WampServer – Guía de descarga e instalación.

En este tutorial, verás cómo descargar e instalar *el servidor web Wamp,* necesario para poder convertir nuestra PC en un servidor y así poder programar en PHP, al mismo tiempo que vemos el resultado.

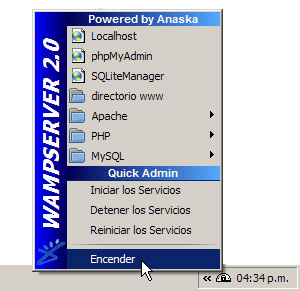
Lo primero es descargar el archivo, que podemos hacerlo accediendo a la siguiente dirección web: <http://www.wampserver.com/download.php> (aproximadamente 19.5mb).

Luego debemos instalar el ejecutable que acabamos de descargar y seguir los pasos de la instalación, en un momento determinado la instalación nos pedirá que definamos nuestro navegador por defecto, debemos buscar el archivo en nuestro disco (ej. C:\Archivos de Programas\Mozilla Firefox\firefox.exe, o el navegador que tú utilices).

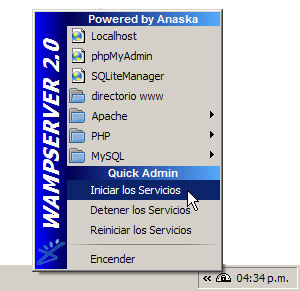
También nos pedirá que configuremos los datos de nuestro servidor SMTP, en caso de tener, ingresamos los datos, en caso contrario dejamos los datos que vienen por defecto y le damos a "Next >".

Una vez el programa esté instalado en nuestra PC, lo ejecutamos y para ponerlo en español, nos vamos con clic derecho sobre el iconito que generó en la barra de tareas (al lado del reloj), y definimos language > spanish.

Ahora, comencemos a utilizarlo… Lo primero es encender el servidor, para ello hacemos clic (normal) sobre el ícono en nuestra barra de tareas, y le damos clic a "Encender" (ver Figura 1.1). Y luego clic a "Iniciar los servicios" (ver Figura 1.2).



**Figura 1.1.**  
*Encender el WampServer.*

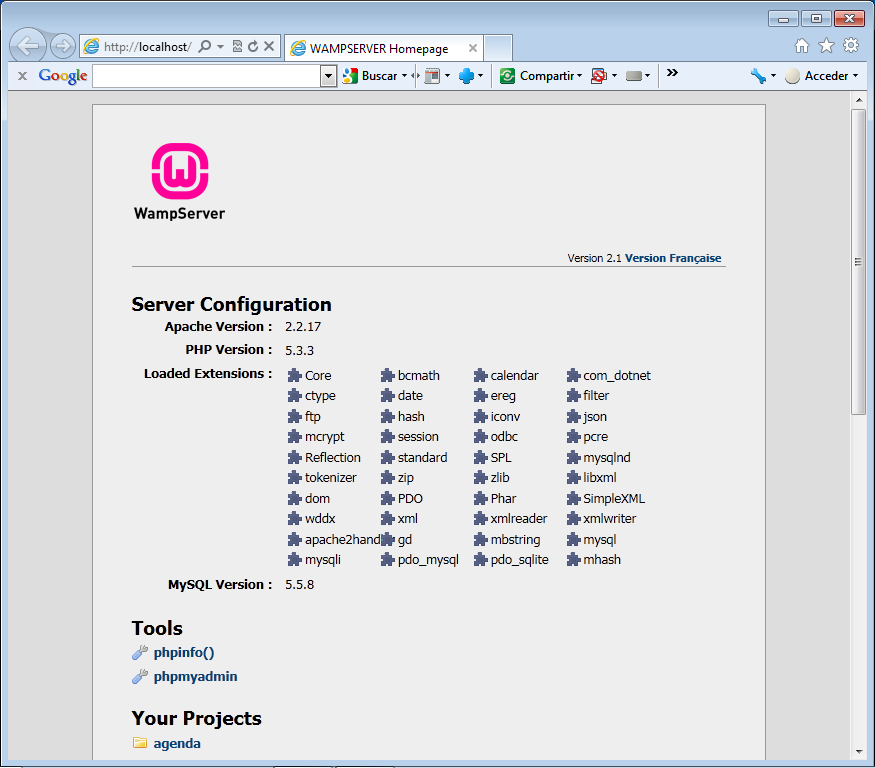


**Figura 1.2.**  
*Iniciar los servicios de WampServer.*

Enhorabuena, aquí ya tenemos montado nuestro servidor local!

Para probarlo, accederemos a nuestro localhost, para ello, abrimos nuestro navegador favorito y tecleamos la siguiente url: [*http://localhost/*](http://localhost/)

Nos debería abrir una página como la siguiente:



En esa página inicial de tu localhost, tienes el link para acceder al manejador de base de datos *phpmyadmin*.

Ahora solo resta que sepas que para probar tus scripts en tu servidor debes guardarlos en la carpeta */www* dentro del directorio donde instalaste el wampserver (por defecto C:\wamp\www), y para acceder a tus proyectos simplemente lo harás a través de la siguiente url: <http://localhost/_ruta_proyecto_/archivo.php> (sustituyendo *\_ruta\_proyecto\_* y *archivo.php* por tu ruta y archivos…).

(Pixelar.me, Marcelo, 2008)

# Unidad 2: El lenguaje de programación del lado del servidor PHP.

PHP, el "hijo" del Código Abierto (*Open Source*) es, en la actualidad uno de los lenguajes de programación que cuenta con más adeptos en la *Red*, que crecen cada día más por su sencillez a la hora de ser programado, su solidez y su constante innovación.

## 2.1. ¿Qué es PHP?

PHP son las siglas de *PHP Personal Hypertext Preprocessor* (Preprocesador Personal de Hipertexto), un lenguaje interpretado de alto nivel que es insertado en documentos con extensión .html y es ejecutado en el lado del servidor, a diferencia de otros lenguajes como Javascript, que son interpretados en el lado del cliente (el navegador del usuario). Cuando el código de las páginas se interpreta en el servidor, el usuario recibe en su navegador el producto de esos procesos sin poder determinar qué código lo ha producido.

La forma en que PHP se inserta en los documentos se produce mediante el uso de etiquetas especiales, como **<?php** y **?>** para indicar el comienzo y final del código PHP, respectivamente. Por ejemplo:

<?php  
 echo "hola desde PHP"  
?>

Para editar o escribir páginas o código que incluya PHP puede usar desde un editor de texto hasta un editor HTML de su preferencia.

## 2.2. Características del lenguaje

Como todo lenguaje, PHP dispone de sus propias especificaciones, sintaxis, variables, uso de éstas dentro y fuera de funciones, funciones predefinidas, etc. En este apartado estudiaremos aquéllas de más relevancia, intentando darle un enfoque práctico a la explicación, por lo que, antes de comenzar, cree en la raíz de su servidor una nueva carpeta con el nombre *phpBasicos*. En esta carpeta, alojaremos los distintos documentos .php que iremos creando a medida que avanzamos en los contenidos de este folleto.

### 2.2.1. Características generales de PHP

Cuando escribimos código en lenguaje PHP es muy importante recordar que al finalizar cada sentencia ha de agregarse un punto y coma:

$miVariable = 5

Esta expresión es incorrecta y generará un error. En cambio, la expresión:

$miVariable = 5;

Sí es correcta. A medida que escriba sus scripts de PHP, puede que desee añadir comentarios para una mejor comprensión del código. Esto puede realizarlo anteponiendo la cadena de texto // al comentario si éste es de una sola línea o bien colocarlo entre los caracteres /\* y \*/ si ocupan más de una línea:

// esto es un comentario de una sola línea

/\*  
esto es  
un comentario  
de varias líneas  
\*/

### 2.2.2. Impresión de datos

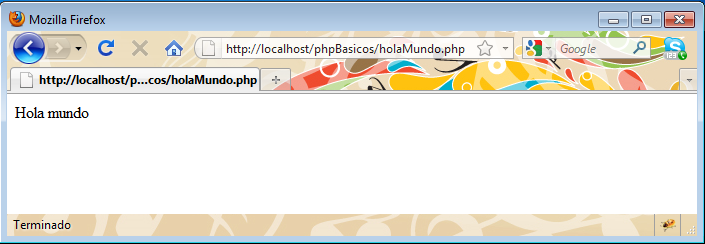
Por "impresión de datos" hacemos referencia a los datos que, mediante PHP, mostramos por pantalla, y no a un resultado impreso utilizando un dispositivo externo. En PHP existen varias formas de impresión de datos, siendo la más utilizada la función *echo*, cuya sintaxis es la siguiente:

<?php  
 echo "Hola mundo";  
?>

Este código nos mostrará la frase "Hola mundo" en la pantalla. Escriba el código utilizando el editor de texto que desee y guárdelo con el nombre *holaMundo.php*. Posteriormente, ábralo en su navegador tecleando la dirección <http://localhost/phpBasicos/holaMundo.php> para comprobar que el texto que se muestra en su navegador es:

*Hola mundo*

Tal y como puede ver en la figura 2.1. Puede encontrar el documento *holaMundo.php* en la carpeta *material/capitulo3/php/* del CD.



## 2.3. Utilización de variables en PHP

Para crear variables en PHP, anteponga el carácter $ al nombre de la variable que usted haya elegido:

$miVariable  
$GET['miVariable']

PHP es un lenguaje *case sensitive*; esto es, sensible a la diferencia entre mayúsculas y minúsculas, por lo que habrá de prestar atención puesto que, por ejemplo, $miVariable y $mivariable son dos variables distintas.

### 2.3.1. Variables predefinidas

PHP proporciona una enorme cantidad de variables predefinidas que están disponibles para cualquier *script* de código que se ejecute. Muchas de estas variables dependen de la configuración del servidor, pudiendo encontrar las que vamos a utilizar dentro de matrices, variables almacenadas en *cookies* y sesiones o enviadas vía POST o GET. La sintaxis para acceder a estas variables que se encuentran dentro de las matrices es la siguiente:

$\_GET['miVariable'];

Donde *miVariable* es una variable pasada a un *script* en forma GET.

$\_POST['miVariable'];

Donde *miVariable* es una variable pasada a un *script* en forma POST.

$\_COOKIE['miVariable'];

Donde *miVariable* es una variable almacenada en una *cookie*.

$\_SESSION['miVariable'];

Donde *miVariable* es una variable almacenada en una sesión de usuario.

$\_GLOBALS['miVariable'];

Donde *miVariable* es una variable almacenada por nosotros en la matriz de variables globales.

### 2.3.2. Variables locales, globales y superglobales

Podemos clasificar las variables en PHP en función del lugar en el que se utilizan. Por ejemplo, cuando estamos utilizando funciones, las variables que declaramos dentro de las mismas se conocen como variables locales y son eliminadas o borradas cuando la función termina de ejecutarse:

function sumar ($n, $m) {  
 $resultado = $n + $m;  
 return $resultado;  
}  
echo sumar(5,6);

En este ejemplo la variable $resultado es local para la función sumar.

Si desea utilizar variables externas a la función, ha de hacerlas globales previamente, para informar a la función de que va a necesitar valores que están alojados fuera de ella:

$n = 5;  
function sumar ($m) {  
 global $n;  
 $resultado = $n + $m;  
 return $resultado;  
}  
echo sumar(6);

En este ejemplo, la función sumartoma el valor de una variable externa ($n) para realizar las operaciones correspondientes.

Las variables autoglobales o superglobales son aquéllas para las que no es necesario hacer referencia en el interior de la función, puesto que dicho proceso se ha realizado ya de una forma que podríamos denominar "automática".

Estas variables son $\_GET, $\_POST, $\_COOKIE, $\_SESSION y $\_GLOBALS.

$\_GLOBALS['n'] = 5;  
function sumar ($m) {  
 $resultado = $n + $m;  
 return $resultado;  
}  
echo sumar(6);

En la dirección Web <http://www.php.net/manual/es> tiene disponible la última versión del manual de PHP, donde puede encontrar más información acerca del modo en que se manejan las variables en este lenguaje.

### 2.3.3. Concatenación de cadenas de texto

Para concatenar dos cadenas de texto en PHP se utiliza el carácter . , tal y como se muestra en el siguiente ejemplo:

<?php  
$variable1 = "hola";  
$variable2 = $variable1 . " mundo";  
echo $variable1 . "<br>". $variable2;  
?>

Mediante este *script* asignamos como contenido de *$variable1* la cadena de texto "hola", que concatenamos con " mundo" para generar el contenido de *$variable2*, "hola mundo".

PHP es un lenguaje capaz de acceder al valor de las variables que se encuentran en el interior de la cadenas de texto delimitadas por comillas dobles, por lo que las expresiones:

echo $variable1 . "<br>" . $variable2;

y:

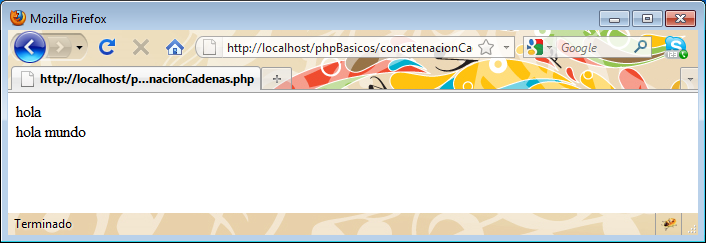
echo "$variable1 <br> $variable2";

son equivalentes. Escriba el *script* anterior en su editor de texto y guárdelo como *concatenacionCadenas.php* para posteriormente probarlo en su navegador tecleando <http://localhost/phpBasicos/concatenacionCadenas.php>.

Tal y como puede ver en la figura 2.2, el texto que aparece es:

hola  
hola mundo

ya que, además de los contenidos de las dos variables, el *script* contiene el texto "<br>", que es un *tag* HTML utilizado para generar un salto de línea. Puede encontrar el documento *concatenacionCadenas.php* en la carpeta *material/capitulo3/php/* del CD.



**Figura 2.2.**  
*Concatenación de cadenas de texto.*

### 2.3.4. Adición de más texto a una variable ya existente

Si desea agregar más información a una variable ya existente que además ya dispone de contenido, puede usar igualmente el carácter . :

<?php  
$variable1 = "esto es algo de texto";  
echo $variable1 . "<br>";  
$variable1 .= " ahora variable1 también tiene este texto";  
echo $variable1;  
?>

En este ejemplo, la expresión:

$variable1 .= " ahora variable1 también tiene este texto";

Es equivalente a:

$variable1 = $variable1 . " ahora variable1 también tiene este texto";

Por lo que finalmente el contenido de la variable $variable1 es "esto es algo de texto ahora variable1 también tiene este texto", lo que puede comprobar copiando el código en su editor de texto, guardarlo con el nombre que desee y añadirlo a la carpeta *phpBasicos* de su servidor para poder posteriormente abrirlo en su navegador y ver el resultado.

### 2.3.5. Utilización y manipulación de *arrays*

Un array es un conjunto de variables de una misma clase. Se accede a cada elemento individual del array mediante un número entero denominado índice. 0 es el índice o localizador del primer elemento y n-1 es el índice del último elemento, siendo n la dimensión, tamaño o longitud del array. Así $empleado[23] representaría al empleado con índice 23.

Las variables tienen el inconveniente de que sólo pueden almacenar un valor en un momento determinado, por lo que PHP también dispone de *arrays* como estructura para almacenar datos. Para crear un nuevo *array* en PHP, usaremos la palabra reserva *array* siguiendo la sintaxis:

$miArray = array(elemento1, elemento2, …, elementoN);

Por ejemplo:

$miArray = array(0,1,2,3,4,5,6);

Sirve para crear un *array* de siete elementos, en el que el acceso a cada posición se realiza utilizando un índice cuyo valor está entre 0 y el número de elementos menos uno, es decir, entre 0 y 6, por lo que $miArray[6] almacena el valor 6.

Sin embargo, también se pueden crear *arrays* en los que el acceso a cada posición se realice mediante una cadena de texto, como por ejemplo:

$miArray = array("nombre"=> "Juan", "apellido"=> "Pérez");

Donde los contenidos del *array* ("Juan" y "Pérez") se referencian por $miArray["nombre"] y $miArray["apellido"], respectivamente.

Para añadir más elementos al final del *array* utilice la función *array\_push* siguiendo la sintaxis:

array\_push(nombre\_del\_array, nuevo\_valor1, …, nuevo\_valorN);

Por ejemplo, para añadir dos nuevos elementos al final del *array* $miArray utilizaríamos:

array\_push($miArray,7,8);

#### 2.3.5.1. Recorriendo los elementos de un *array*

Utilizaremos bucles para recorrer un *array*, extrayendo los valores que tiene almacenados mediante uno de los dos métodos a nuestra disposición. El primero de ellos consiste en utilizar como índice el valor de una variable que se incrementa en cada iteración del bucle, mientras que el segundo consiste en utilizar funciones predefinidas de PHP (each, next), que extraen los valores mientras incrementan el apuntador del *array*.

Para entender mejor qué son los apuntadores y cómo funcionan, vamos a crear un archivo en el que definiremos un *array*, del que extraeremos los datos primeramente mediante un bucle for y una variable índice, y posteriormente mediante una función predefinida.

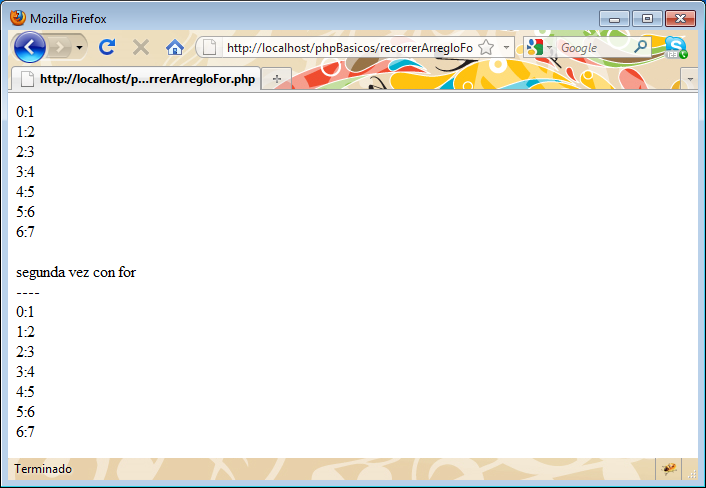
El cómo recorrer un *array* utilizando un bucle for puede encontrarlo en el documento *material/capitulo3/php/recorrerArregloFor.php* del CD:

<?php  
$myArray = array(1,2,3,4,5,6,7);  
for ($n=0; $n < sizeOf($myArray); $n++) {  
 echo $n . ":". $myArray[$n]. "<br>";  
}  
echo "<br>segunda vez con for<br>----<br>";  
for ($n=0; $n < sizeOf($myArray); $n++) {  
 echo $n . ":". $myArray[$n]. "<br>";  
}  
?>

Teclee el código en un archivo nuevo con su editor de texto, guárdelo como *recogerArregloFor.php* y ábralo en su navegador tecleando <http://localhost/phpBasicos/recogerArregloFor.php>. La información que debe aparecer en su pantalla ha de ser la misma que se muestra en la figura 2.3:

0:1  
1:2  
2:3  
3:4  
4:5  
5:6  
6:7

segunda vez con for  
----  
0:1  
1:2  
2:3  
3:4  
4:5  
5:6  
6:7



**Figura 2.3.**  
*Extrayendo datos de un array usando un bucle for.*

En el ejemplo anterior, hemos extraído en dos ocasiones los valores almacenados en el *array* para mostrar que el apuntador no fue alterado tras la primera vez.

Ahora vamos a recorrer el *array* usando each, pero antes vamos a analizar esta función; each devuelve dos pares de valores en un *array* cuyos campos pueden accederse de dos formas, usando los índices 0 y 1 o bien las claves *key* y *value*. Este *array* suele asignarse a una variable que es la que se maneja para extraer los datos.

Por ejemplo, si creamos la siguiente variable:

$temp = each($myArray);

*temp* dispondrá de los siguientes valores:

$temp[0]; (índice)  
$temp[1]; (valor)  
$temp['key']; (índice)  
$temp['value']; (valor)

Copie el siguiente código en un nuevo documento de su editor de texto y guárdelo con el nombre *recogerArregloEach.php*:

<?php  
$myArray = array(1,2,3,4,5,6,7);  
while ($temp = each($myArray)) {  
 echo $temp['key']. ":". $temp['value']. "<br>";  
}

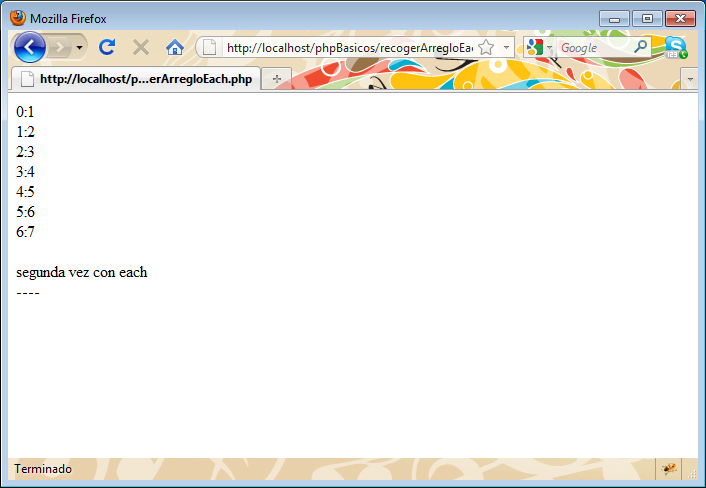
echo "<br>segunda vez con each<br>----<br>";  
while ($temp = each($myArray)) {  
 echo $temp['key']. ":". $temp['value']. "<br>";  
}  
?>

Abra del documento en su navegador tecleando <http://localhost/phpBasicos/recogerArregloEach.php>.

La información que debe aparecer en su pantalla ha de ser la misma que se muestra en la figura 2.4:

0:1  
1:2  
2:3  
3:4  
4:5  
5:6  
6:7

segunda vez con each  
----



**Figura 2.4.**  
*Extrayendo datos de un array usando each.*

En este caso sólo se muestra el contenido del *array* una vez, ya que al recorrerlo la primera vez se iba modificando el apuntador. Al intentar recorrer la estructura por segunda vez, el apuntador ya se encontraba al final del *array*, por lo que no se encuentra ningún contenido que mostrar. Puede encontrar el código de este *script* en el documento *material/capitulo3/php/recorrerArregloEach.php* del CD.

La solución a este inconveniente nos la proporciona la función *reset*, que sirve para reiniciar la posición del apuntador mediante la sintaxis:

reset ($nombre\_del\_array);

Cree un documento de nombre *recorrerArregloEachReset.php* y teclee de nuevo el código (puede encontrarlo en el documento *material/capitulo3/php/recorrerArregloEachReset.php*), añadiendo la línea que contiene la función *reset* antes de recorrer el *array* por segunda vez:

<?php  
$myArray = array(1,2,3,4,5,6,7);  
while ($temp = each($myArray)) {  
 echo $temp['key']. ":". $temp['value']. "<br>";  
}

echo "<br>segunda vez con each<br>----<br>";

reset ($myArray);

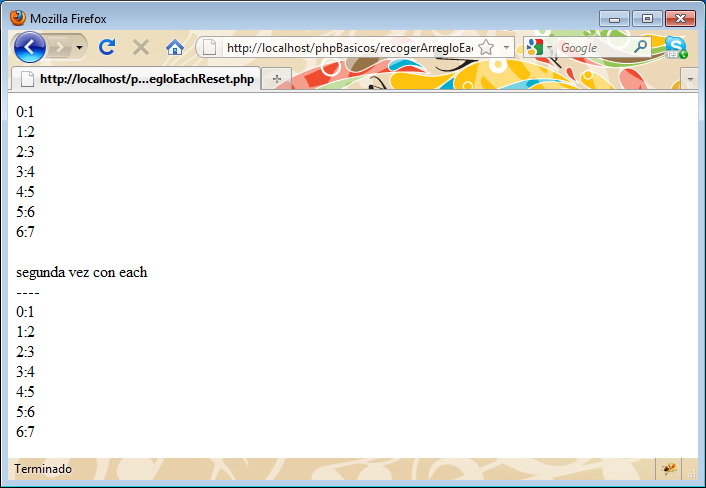
while ($temp = each($myArray)) {  
 echo $temp['key']. ":". $temp['value']. "<br>";  
}  
?>

Abra el documento en su navegador tecleando <http://localhost/phpBasicos/recogerArregloEachReset.php>.

La información que debe aparecer en su pantalla ha de ser la misma que se muestra en la figura 2.5:

0:1  
1:2  
2:3  
3:4  
4:5  
5:6  
6:7

segunda vez con each  
----  
0:1  
1:2  
2:3  
3:4  
4:5  
5:6  
6:7



**Figura 2.5.**  
*Extrayendo datos de un array usando each y reiniciando el apuntador.*

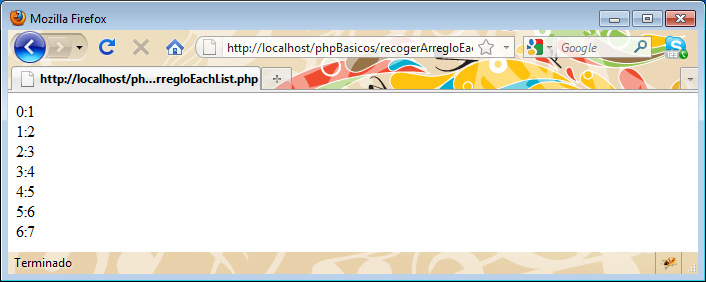
Existe una alternativa a la de utilizar las claves *key* y *value*, y es la de asignar el resultado a variables que definiremos en una lista:

<?php  
$myArray = array(1,2,3,4,5,6,7);  
while (list($clave, $valor) = each($myArray)) {  
 echo "$clave : $valor <br>";  
}  
?>

Copie el código en un nuevo documento de su editor de texto y guárdelo con el nombre *recorrerArregloEachList.php*, para posteriormente abrirlo en su navegador tecleando <http://localhost/phpBasicos/recorrerArregloEachList.php> (este código se encuentra disponible en el documento material/capitulo3/php/recorrerArregloEachList.php del CD).

La información que debe aparecer en su pantalla ha de ser la misma que se muestra en la figura 2.6:

0:1  
1:2  
2:3  
3:4  
4:5  
5:6  
6:7



**Figura 2.6.**  
*Extrayendo datos de un array usando each y list.*

# Unidad 3: El servidor de bases de datos MySQL

MySQL es uno de los servidores de bases de datos de código abierto (*Open Source*) más populares y conocidos en el mundo, un sistema de manejo de bases de datos sin igual en rapidez, estabilidad y facilidad de desarrollo. Dispone, además, de una arquitectura que lo hace extremadamente rápido y fácil de personalizar.

**Servidores de bases de datos**

Un servidor de bases de datos es un programa que almacena datos estructurados en forma de tablas. El servidor acepta conexiones de clientes a través de un puerto TCP/IP y admite consultas realizadas en lenguaje SQL (*Structured Query Language*) para posteriormente devolver los datos resultantes del procesamiento de aquellas al cliente a través de la Red.

## 3.1. Organización de MySQL

MySQL está conformada por distintas bases de datos, cada una de las cuales de una o varias tablas, que son las que contienen la información y que constan de tres elementos principales:

* Columnas.
* Datos.
* Índices.

### 3.1.1. Columnas

Como las tablas están constituidas por columnas, cada una de estas últimas ha de tener un nombre único, para que podamos referirnos a cada una de ellas sin riesgo de confundirla con alguna de las otras.

Al ser la tabla formada por columnas, cada columna debe tener un nombre único, para así referirnos a cada una de ellas específicamente. En MySQL existen tres tipos distintos de columnas, dependiendo de los datos que vayan a almacenar:

* Tipo Numérico.
* Tipo Fecha.
* Tipo Cadena.

#### 3.1.1.1. Tipo Numérico

Los tipos de datos numéricos pueden dividirse en dos grandes grupos: aquellos que tienen decimales (coma flotante) y aquellos que nos los tienen.

|  |  |
| --- | --- |
| **Tipo Numérico** | **Descripción** |
| *TinyInt* | Número entero con o sin signo. Con signo, el rango de valores válidos se encuentra entre -128 y 127. Sin signo, el rango de valores válidos se encuentra entre 0 a 255. |
| *Bit* | Número entero cuyo valor puede ser 0 ó 1. |
| *Bool* | Número entero cuyo valor puede ser 0 ó 1. |
| *SmallInt* | Número entero con o sin signo. Con signo, el rango de valores se encuentra entre -32768 y 32767. Sin signo, el rango de valores se encuentra entre 0 y 65535. |
| *MediumInt* | Número entero con o sin signo. Con signo, el rango de valores válidos se encuentra entre -8.388.608 y 8.388.607. Sin signo, el rango de valores válidos se encuentra entre 0 y 6777215. |
| *Integer* | Número entero con o sin signo. Con signo, el rango de valores válidos se encuentra entre -2147483648 y 2147483647. Sin signo, el rango de valores válidos se encuentra entre 0 y 429.4967.295. |
| *Int* | Número entero con o sin signo. Con signo, el rango de valores válidos se encuentra entre -2147483648 y 2147483647. Sin signo, el rango de valores válidos se encuentra entre 0 y 429.4967.295. |
| *BigInt* | Número entero con o sin signo. Con signo, el rango de valores válidos se encuentra entre -9.223.372.036.854.775.808 y 9.223.372.036.854.775.807. Sin signo, el rango de valores válidos se encuentra entre 0 y 18.446.744.073.709.551. |
| *Float* | Número pequeño en coma flotante de precisión simple. El rango de valores válidos se encuentra entre -3.402823466E+38 y -1.175494351E-38, 0 y entre -1.175494351E-38 y 3.402823466E+38[[1]](#footnote-1). |
| *xReal* | Número en coma flotante de precisión doble. El rango de valores válidos se encuentra entre -1.7976931348623157E+308 y -2.2250738585072014E-308, 0 y entre 2.2250738585072014E-308 y 1.7976931348623157E+308. |
| *Double* | Número en coma flotante de precisión doble. El rango de valores válidos se encuentra entre -1.7976931348623157E+308 y -2.2250738585072014E-308, 0 y entre 2.2250738585072014E-308 y 1.7976931348623157E+308. |
| **Tipo Numérico** | **Descripción** |
| *Decimal* | Número en coma flotante desempaquetado. En este caso el número se almacena como una cadena. |
| *Dec* | Número en coma flotante desempaquetado. En este caso el número se almacena como una cadena. |
| *Numeric* | Número en coma flotante desempaquetado. En este caso el número se almacena como una cadena. |

#### 3.1.1.2. Tipo Fecha

Cuando almacenamos fechas en nuestras tablas, hay que tener en cuenta el hecho de que MySQL no comprueba de forma concreta si una fecha es o no válida, sino si el mes indicado está comprendido entre 0 y 12 y el día del mes está comprendido entre 0 y 31. Atendiendo a este factor, disponemos de varios tipos que podemos utilizar para manejar fechas:

**Tabla 3.2.**

*Tipos de fecha disponibles para las tablas MySQL.*

|  |  |
| --- | --- |
| **Tipo Fecha** | **Descripción** |
| *Time* | Almacena una hora. El rango de valores válidos para la misma se encuentra entre -838 horas, 59 minutos y 59 segundos y 838 horas, 59 minutos y 59 segundos. El formato de almacenamiento es 'HH:MM:SS'. |
| *Year* | Almacena un año. El rango de valores válidos para el mismo se encuentra entre el año 1901 y el año 2155. El campo puede tener tamaño 2 o tamaño 4, dependiendo de si deseamos almacenar el año con dos o cuatro dígitos. |
| *Date* | Almacena una fecha. El rango de valores válidos de la misma se encuentra entre el 1 de enero de 1001 y el 31 de diciembre de 9999. El formato de almacenamiento es año-mes-día. |
| *DateTime* | Combinación de fecha y hora. El rango de valores se encuentra entre las 0 horas, 0 minutos y 0 segundos del 1 de enero de 1001 y las 23 horas, 59 minutos y 59 segundos del 31 de diciembre del 9999. El formato de almacenamiento es de año-mes-día horas:minutos:segundos. |
| *TimeStamp* | Combinación de fecha y hora. El rango de valores válidos de la misma se encuentra entre el 1 de enero de 1970 y el 31 de diciembre de 2037. El formato de almacenamiento depende del tamaño del campo como puede observar en la tabla 3.3. |

**Tabla 3.3.**

*Formatos de almacenamiento del tipo fecha TimeStamp.*

|  |  |
| --- | --- |
| **Tamaño** | **Formato** |
| 14 | AñoMesDíaHoraMinutoSegundo aaaammddhhmmss. |
| 12 | AñoMesDíaHoraMinutoSegundo aammddhhmmss. |
| 8 | AñoMesDía aaaammdd. |
| 6 | AñoMesDía aammdd. |
| 4 | AñoMes aamm. |
| 2 | Año aa. |

#### 3.1.1.3. Tipo Cadena

La principal distinción que se hace entre los tipos de cadena disponibles para las tablas MySQL es la de que las cadenas puedan disponer de longitud fija o variable:

**Tabla 3.4.**

*Tipos de cadena disponibles para las tablas MySQL.*

|  |  |
| --- | --- |
| **Tipo Cadena** | **Descripción** |
| *Char(n)* | Almacena una cadena de longitud fija cuyo número de caracteres puede estar comprendido entre 0 y 255. |
| *VarChar(n)* | Almacena una cadena de longitud variable cuyo número de caracteres puede estar comprendido entre 0 y 255. |

Pero, además de esta distinción, también pueden distinguirse otros dos subtipos dentro de los tipos de cadena, atendiendo al tratamiento que reciben a la hora de realizar ordenamientos y comparaciones. Estos dos tipos son el tipo Text y el tipo BLOB (*Binary Large Object*), en los que mientras que el primero se ordena sin tener en cuenta mayúsculas y minúsculas, el segundo se ordena teniéndolas en cuenta. Estos últimos, los tipos pertenecientes al subtipo BLOB, se utilizan para almacenar datos binarios, como pueden ser los de los ficheros, por ejemplo.

**Tabla 3.5.**

*Subtipos de tipo BLOB disponibles para las tablas MySQL.*

|  |  |
| --- | --- |
| **Tipo Cadena** | **Descripción** |
| *TinyText* | Columna con una longitud máxima de 255 caracteres. |
| *TinyBlob* | Columna con una longitud máxima de 255 caracteres. |
| *Blob* | Texto con una longitud máxima de 65535 caracteres. |
| *Text* | Texto con una longitud máxima de 65535 caracteres. |
| *MediumBlob* | Texto con una longitud máxima de 16.777.215 caracteres. |
| *MediumText* | Texto con una longitud máxima de 16.777.215 caracteres. |
| *LongBlob* | Texto con una longitud máxima de 4.294.967.295 caracteres. Hay que tener en cuenta que, debido a los protocolos de comunicación, los paquetes que se envían a través de la Red pueden tener un máximo de 16 Mb. |
| *LongText* | Texto con una longitud máxima de 4.294.967.295 caracteres. Hay que tener en cuenta que, debido a los protocolos de comunicación, los paquetes que se envían a través de la Red pueden tener un máximo de 16 Mb. |
| *Enum* | Campo que puede obtener un único valor de una lista de posibles valores donde se especifica. Este tipo acepta hasta 65535 valores distintos. |
| *Set* | Campo que puede contener ninguno, uno, o bien varios valores de una lista que puede tener un máximo de 64 valores. |

Las columnas, además del tipo al que pertenecen, también pueden tener propiedades o atributos:

* **Longitud:** Número de caracteres que acepta la columna para los valores que se introducen en ella.
* **Default:** Valor predeterminado, es decir, el valor que se asigna a la columna cuando no se le asigna un valor específico.
* **Null**-**Not Null:** Aceptación o no de valores nulos en caso de que la columna pueda recibir valores nulos.
* **Auto\_increment**: Autoincremental, aumenta en una unidad su valor por cada nuevo registro que se añada a la columna.

## 3.1.2. Datos

Los datos son la información que se almacena por filas dentro de las distintas tablas MySQL de las que disponemos, y cada uno de los registros es del tipo de datos de la columna a la que pertenece.

## 3.1.3. Índices

Las tablas de MySQL pueden utilizar índices, que sirven para mejorar el tiempo de respuesta de MySQL en las consultas cuando se realizan búsquedas en las distintas tablas. Puede establecer distintos tipos de índices para sus tablas:

**Tabla 3.6.**

*Tipos de índice para las tablas MySQL.*

|  |  |
| --- | --- |
| **Tipos de índice** | **Descripción** |
| *Primary* | Establece que el campo en el que se define sea primario. |
| *Index* | Crea un indexado del campo para que la búsqueda sea más rápida. |
| *Unique* | Establece que los valores de ese campo sean únicos para cada nuevo dato de la tabla. |

**Nota:** Es altamente recomendable que cada una de las tablas que cree posea un campo con valores únicos, lo que le ayudará a la identificación de ese registro. Por lo general, este registro único es un número cuyo valor aumenta en una unidad con cada registro nuevo que se añade a la tabla (un número autoincremental), y se le suele dar un nombre que sea identificativo de lo que representa, por ejemplo "id".

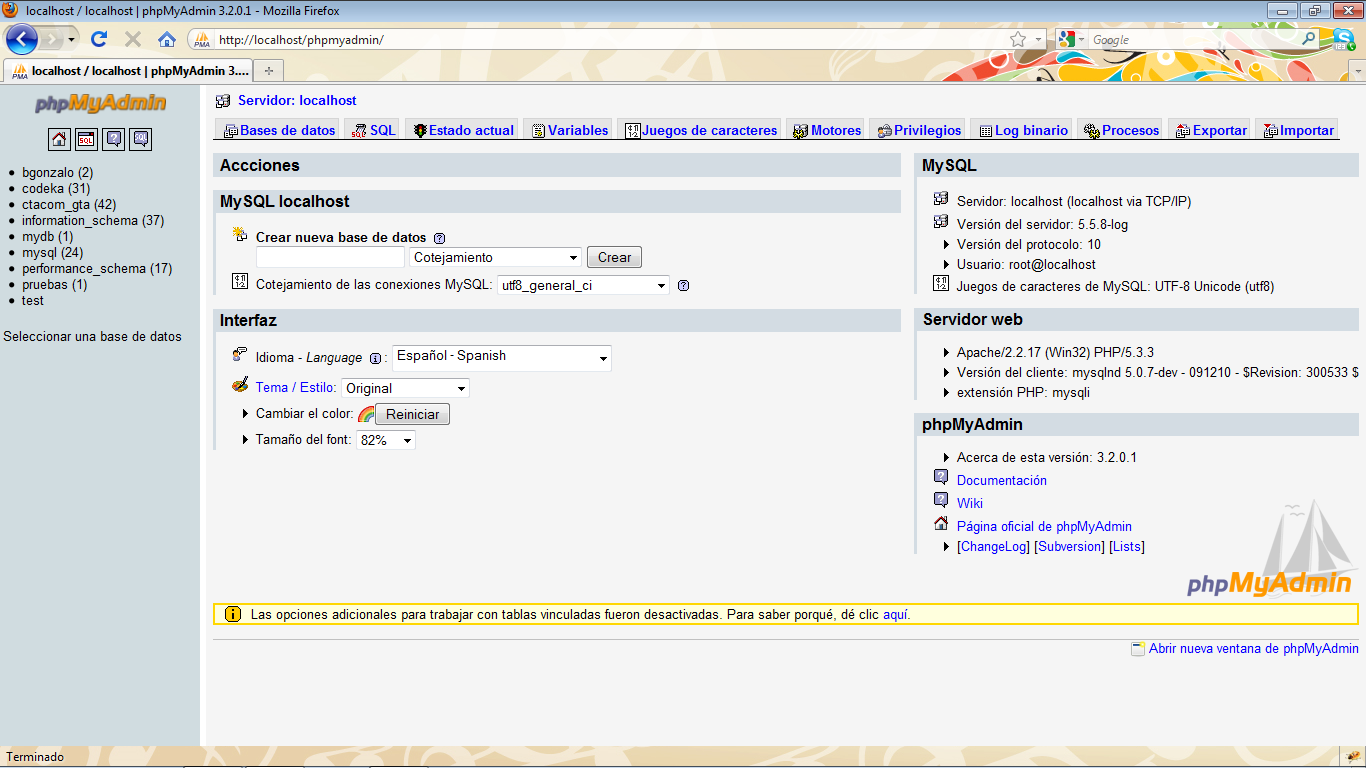
## 3.2. Manipulación y utilización de MySQL

Para manipular MySQL puede usted utilizar la terminal o consola del ordenador (una ventana del sistema MS-DOS), pero, por motivos de seguridad, la mayoría de los proveedores de hospedaje de sitios Web no permiten acceder a la existente en el servidor, por lo que suele ofrecerse a cambio el tener instalado un administrador o manipulador para la base de datos.

Para acceder al administrador de base de datos, abra una nueva ventana de su navegador y teclee:

[*http://localhost/phpMyAdmin*](http://localhost/phpMyAdmin)*[[2]](#footnote-2)*

tras lo que deberá ver en su navegador el administrador de bases de datos dispuesto para ser utilizado tal y como se muestra en la figura 3.1.

**Figura 3.1.**  
*Pantalla de bienvenida al administrador de bases de datos.*

Para irnos acostumbrando al funcionamiento de MySQL, vamos a crear primeramente una base de datos, y posteriormente, y dentro de ésta, crearemos una tabla sobre la que haremos inserciones, modificaciones y selecciones de datos, de modo que las estructuras a crear en los siguientes apartados le servirán además para ir realizando las distintas prácticas y ejercicios necesarios para familiarizarse con la manipulación de las bases de datos en su servidor.

### 3.2.1. Creación de una base de datos

Para comenzar, vamos a crear una nueva base de datos llamada *pruebas* en su administrador de bases de datos (*phpMyAdmin*). Para ello, teclee:

*pruebas*

en el cuadro de introducción de texto que se encuentra debajo de Crear nueva base de datos y pulse el botón **Crear.**

Si desea utilizar la consola (para lo que puede presionar el enlace Ventana de consulta del administrador) para crear la base de datos, teclee en la misma:

*CREATE DATABASE pruebas*

### 3.2.2. Creación de una tabla para la base de datos

Una vez que hemos creado la base de datos, necesitaremos al menos una tabla en la que almacenar información que posteriormente manipularemos. Para ello, vamos a crear una tabla de nombre *directorio* y cuya estructura puede observar en la tabla 3.7.

**Tabla 3.7.**

*Estructura de la tabla directorio de la base de datos pruebas.*

|  |  |
| --- | --- |
| **Nombre del campo** | **Tipo de datos** |
| *id* | Entero mediano, autoincremental. |
| *nombre* | Varchar de 20 caracteres. |
| *apellido* | Varchar de 20 caracteres. |
| *email* | Varchar de 30 caracteres. |
| *url* | Varchar de 30 caracteres. |
| *nick* | Varchar de 20 caracteres. |

El administrador de la base de datos nos ofrece dos posibilidades para crear una nueva tabla. La primera de ellas consiste en utilizar un formulario escrito en HTML, para lo cual hemos de comenzar seleccionando en el menú de bases de datos que se encuentra en la izquierda de su administrador la base de datos en la que vamos a crear la tabla; *pruebas* en nuestro caso.

Una vez seleccionada la base de datos *pruebas* accederá a una nueva pantalla en la que podrá observar dos cuadros de introducción de texto debajo de Crear nueva tabla en la base de datos pruebas, siendo el primero de ellos para el nombre de la nueva tabla y el segundo para el número de campos de que constará ésta última, como puede ver en la figura 3.2.

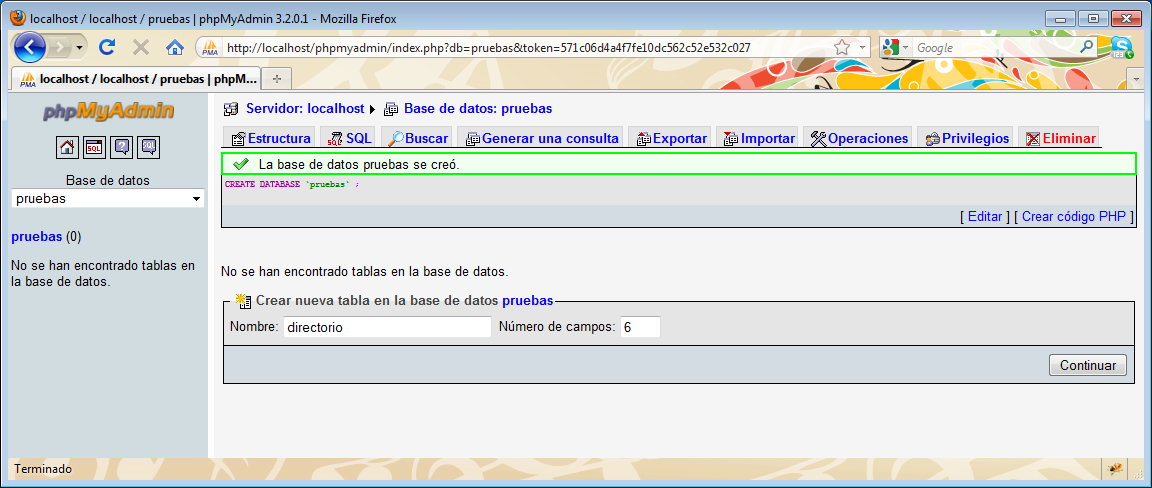
Tecleamos:

*directorio*

en el primer cuadro de introducción de texto y:

*6*

en el segundo, para posteriormente pulsar el botón **Continúe.**



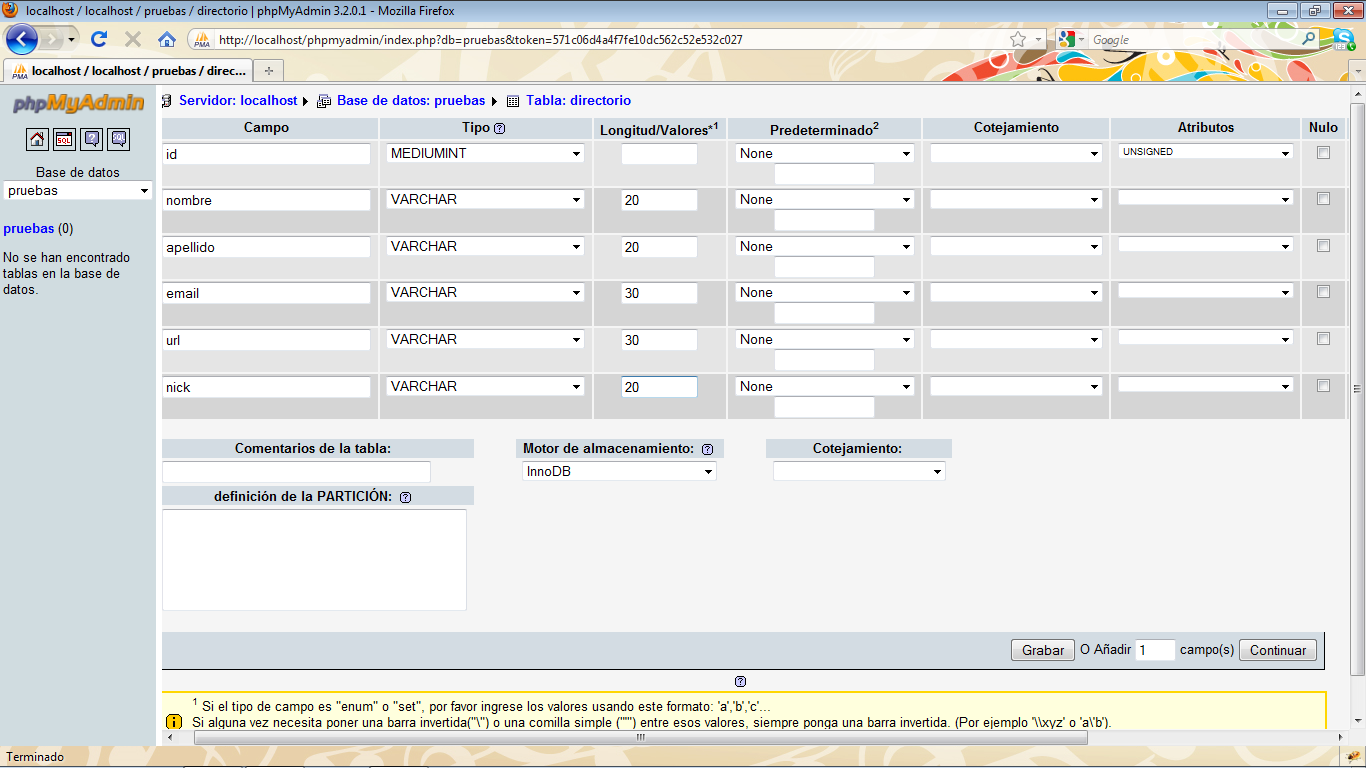
**Figura 3.2.**  
*Creación de la tabla directorio usando phpMyAdmin.*

Veremos entonces en nuestro navegador una serie de campos desplegables y de introducción de texto que deberemos rellenar de acuerdo a la estructura que decidimos que había de tener la tabla *directorio* y que podemos observar en la tabla 3.7. Una vez completados los campos del formulario (como se puede ver en la figura 3.3), pulsamos el botón **Grabar**, con lo que se creará la tabla *directorio* en MySQL, y veremos un mensaje que nos informa del éxito de la operación.

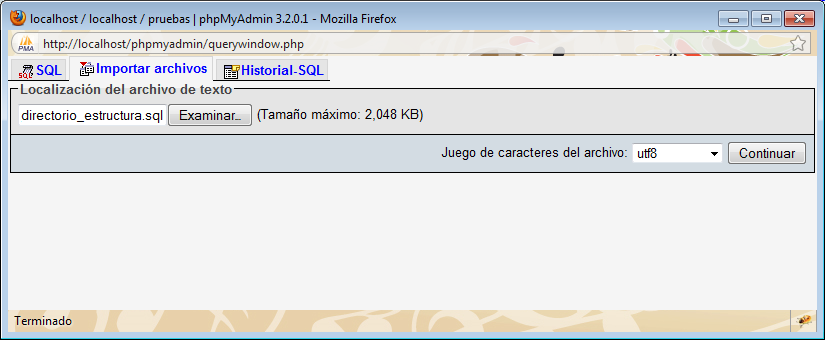
Esta forma de crear las distintas tablas de nuestra base de datos es sencilla de utilizar, pero tiene el inconveniente de que hemos de crear las tablas una por una y, además, nos obliga a rellenar tantos campos como le indiquemos al crear cada una de ellas, con lo que si éstas son grandes o tenemos un elevado número de tablas, su creación puede ser una tarea bastante incómoda. Por ello, MySQL ofrece la posibilidad de importar datos externos, es decir, leer un archivo con la información y estructura (e incluso con los datos) de una o varias tablas, de forma que podemos teclear los comandos que necesitemos en un documento de extensión .sql generado con un editor de texto y posteriormente importarlo desde MySQL para que se ejecuten dichos comandos y se creen las tablas con sus respectivas estructuras.

Como ejemplo práctico de lo que acabamos de leer, vamos a seleccionar la base de datos *pruebas* en el menú de la izquierda del administrador, y posteriormente vamos a pulsar en la pestaña que lleva el texto SQL en la parte derecha, lo que nos lleva a una pantalla con un área de texto y un campo de texto de tipo Archivo junto a un botón **Examinar.** Pulsaremos sobre éste para buscar y seleccionar el documento *material/cap4/sql/directorio\_estructura.sql* del CD, como puede ver en la figura 3.4.

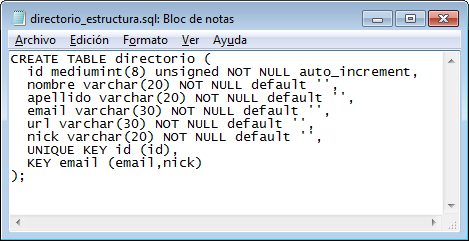
Este documento .sql contiene, como puede ver en la figura 3.5, los comandos necesarios para crear la tabla *directorio* en la base de datos *pruebas.*



**Figura 3.3.**  
*Estructura de la tabla directorio.*



**Figura 3.4.**  
*Carga de documentos .sql externos.*



**Figura 3.5.**  
*Contenidos del documento directorio\_estructura.sql.*

Seleccionado el documento, presionamos el botón **Continúe**, y veremos una pantalla que nos informa el éxito de la creación de la tabla (figura 3.6).



**Figura 3.6.**  
*Creación exitosa de la tabla directorio.*

Este último método es muy útil, no sólo para ganar tiempo a la hora de crear varias tablas, sino también para exportar la estructura de las bases de datos cuando se instalan módulos pre-programados.

### 3.2.3. Inserción de información

Antes de insertar datos en la tabla, vamos a importar un archivo con datos de ejemplo con el fin de que, a la hora de realizar selecciones de registros en ejercicios posteriores, la información sea la misma. Al igual que hicimos con anterioridad para crear una tabla utilizando un documento .sql externo, seleccionaremos la base de datos *pruebas* en el menú de la parte izquierda del administrador y luego la tabla *directorio* del menú de tablas dentro de la base de datos *pruebas*. A continuación, pulsamos sobre la pestaña de nombre SQL de la parte derecha. Utilizando el botón **Examinar** buscamos y seleccionamos el documento *material/cap4/sql/directorio\_datos.sql* del CD y luego pulsamos el botón **Continúe** para que se realicen las acciones que hemos tecleado en nuestro documento .sql externo.

Una vez almacenada en la tabla *directorio* la información del archivo importado, procederemos a insertar información. Para ello, podemos pulsar sobre el vínculo que reza Insertar, lo cual nos permitirá introducir registros nuevos a la tabla de uno en uno, con lo que esto puede conllevar pérdida de tiempo. Para evitar este problema, vamos a introducir la información mediante sentencias SQL. Con la tabla *directorio* seleccionada en la parte izquierda del administrador, pulse sobre la pestaña SQL de la parte derecha, en la que puede ver un área de texto en el que vamos a introducir nuestros comandos.

Para insertar información utilice la palabra reservada INSERT seguida del nombre de la tabla donde se realizará la inserción. Seguidamente, especificamos los campos que serán afectados cuando realizamos una nueva inserción de un registro y, por último, especificamos los valores que tendrán cada uno de esos campos. Resumiendo, la sintaxis debe ser como sigue:

INSERT INTO [nombre\_tabla] (campo1, campo2, campo3) VALUES ('valor\_campo1', 'valor\_campo2', 'valor\_campo3');

Si recuerda la estructura de la tabla *directorio* (tabla 3.7), ésta cuenta con los campos *id, nombre, apellido, email, url* y *nick*; hemos de tener en cuenta que *id* es autoincremental, con lo que no es necesario asignarle información, puesto que MySQL se encargará de insertar el registro con el número correspondiente. Para comprobar lo que acabamos de decir, vamos a insertar un nuevo registro:

**Tabla 3.8.**

*Datos del nuevo registro.*

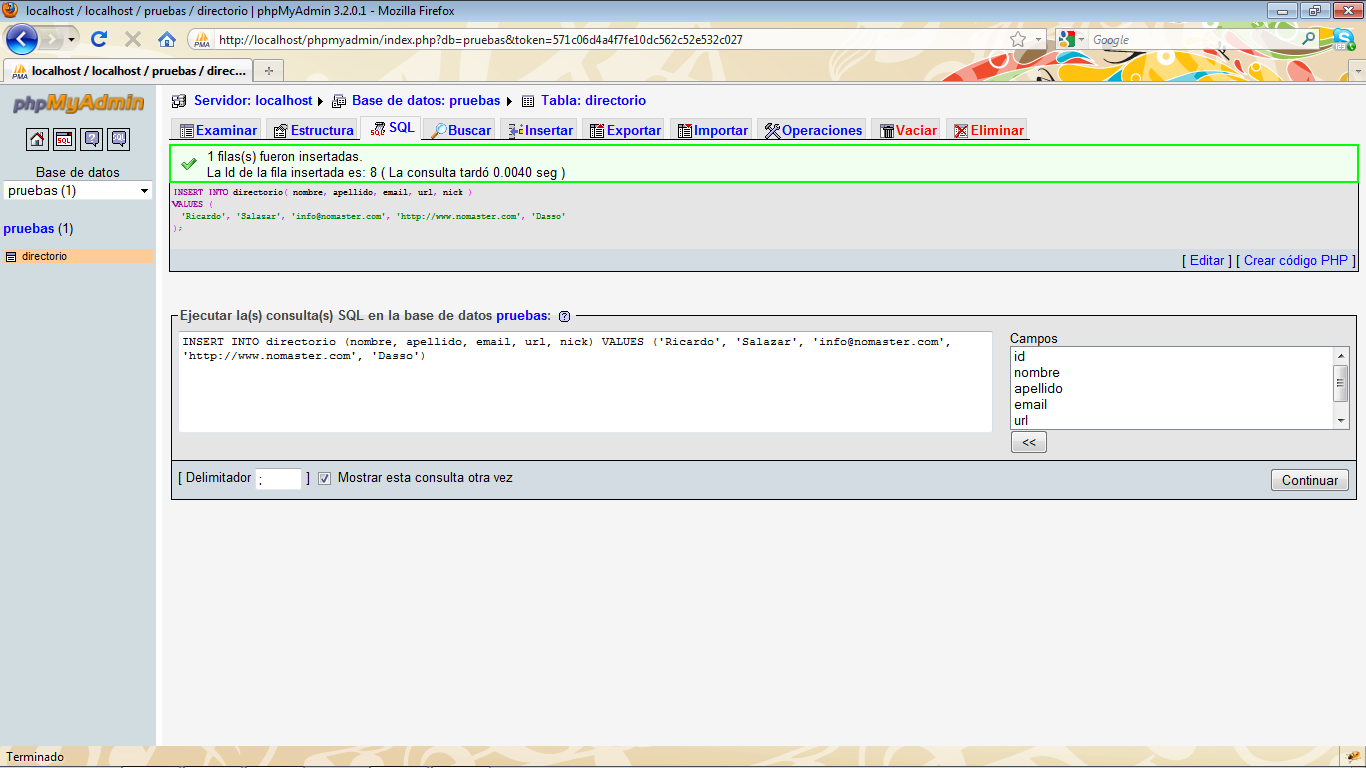
|  |  |
| --- | --- |
| **Campo** | **Contenido** |
| *nombre* | Ricardo |
| *apellido* | Salazar |
| *email* | [info@nomaster.com](mailto:info@nomaster.com) |
| *url* | <http://www.nomaster.com> |
| *nick* | Dasso |

Teclee en el área de texto la consulta para insertar el registro tal y como hemos especificado con anterioridad:

INSERT INTO directorio (nombre, apellido, email, url, nick) VALUES ('Ricardo', 'Salazar', 'info@nomaster.com', 'http://www.nomaster.com', 'dasso');

**Nota:** Tenga en cuenta que los valores para cada uno de los campos se teclean entre comillas simples, no así los nombres de los campos.

Pulsando el botón **Continúe** se agregará el nuevo registro a la tabla *directorio* de la base de datos *pruebas*, como puede ver en la figura 3.7.



**Figura 3.7.**  
*Datos insertados con éxito.*

De esta forma hemos realizado la inserción de un solo registro. Si deseamos insertar, por ejemplo, quince, no tenemos más que escribir una sentencia INSERT INTO … por cada uno de los registros a añadir en un documento .sql y, posteriormente, importarlo a MySQL, tal y como hemos realizado con los datos del documento *directorio\_datos.sql* que hemos visto al inicio de este apartado.

### 3.2.4. Selección de registros de la tabla

La selección de información es una de las actividades más realizadas cuando trabajamos con bases de datos. Cuando realizamos una búsqueda o desplegamos una serie de registros estamos realizando una selección. La sintaxis para realizar una selección de registros de una tabla es variable, siendo la forma básica como se indica a continuación:

SELECT \* FROM [nombre\_tabla];

En este caso, el carácter \* representa todas las columnas y todos los registros de la tabla.

Vamos a realizar diversas consultas de selección con los datos de nuestra tabla *directorio*. Seleccionamos dicha tabla del menú de tablas de la base de datos *pruebas* en la parte izquierda del administrador, y a continuación pulsamos sobre la pestaña SQL. En el área de texto teclee lo siguiente:

SELECT \* FROM directorio;

La siguiente pantalla mostrará todos los registros de la tabla, como puede observar en la figura 3.8.



**Figura 3.8.**  
*Selección de todos los registros.*

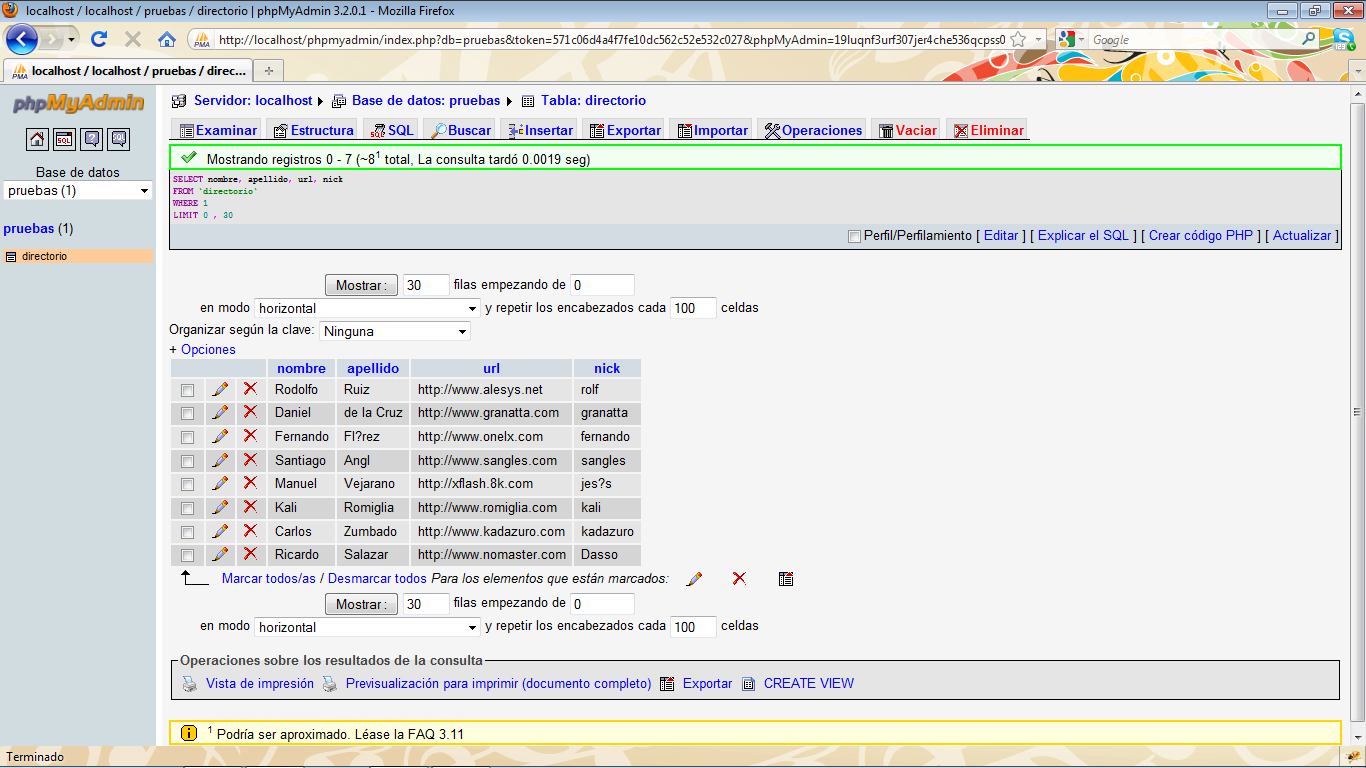
En cambio, si deseamos seleccionar sólo algunos campos de la tabla, habremos de separar los nombres de las columnas por comas, como se indica a continuación:

SELECT nombre\_columna1, nombre\_columna2, …, nombre\_columna\_N FROM [nombre\_tabla];

De esta forma, si deseamos seleccionar únicamente las columnas *nombre, apellido, url* y *nick* de la tabla *directorio* habremos de teclear en el área de texto de SQL lo siguiente:

SELECT nombre, apellido, url, nick FROM directorio;

La siguiente pantalla mostrará únicamente los campos *nombre, apellido, url* y *nick* de cada registro almacenado en la tabla, como puede observar en la figura 3.9.



**Figura 3.9.**  
*Selección de campos concretos de los registros.*

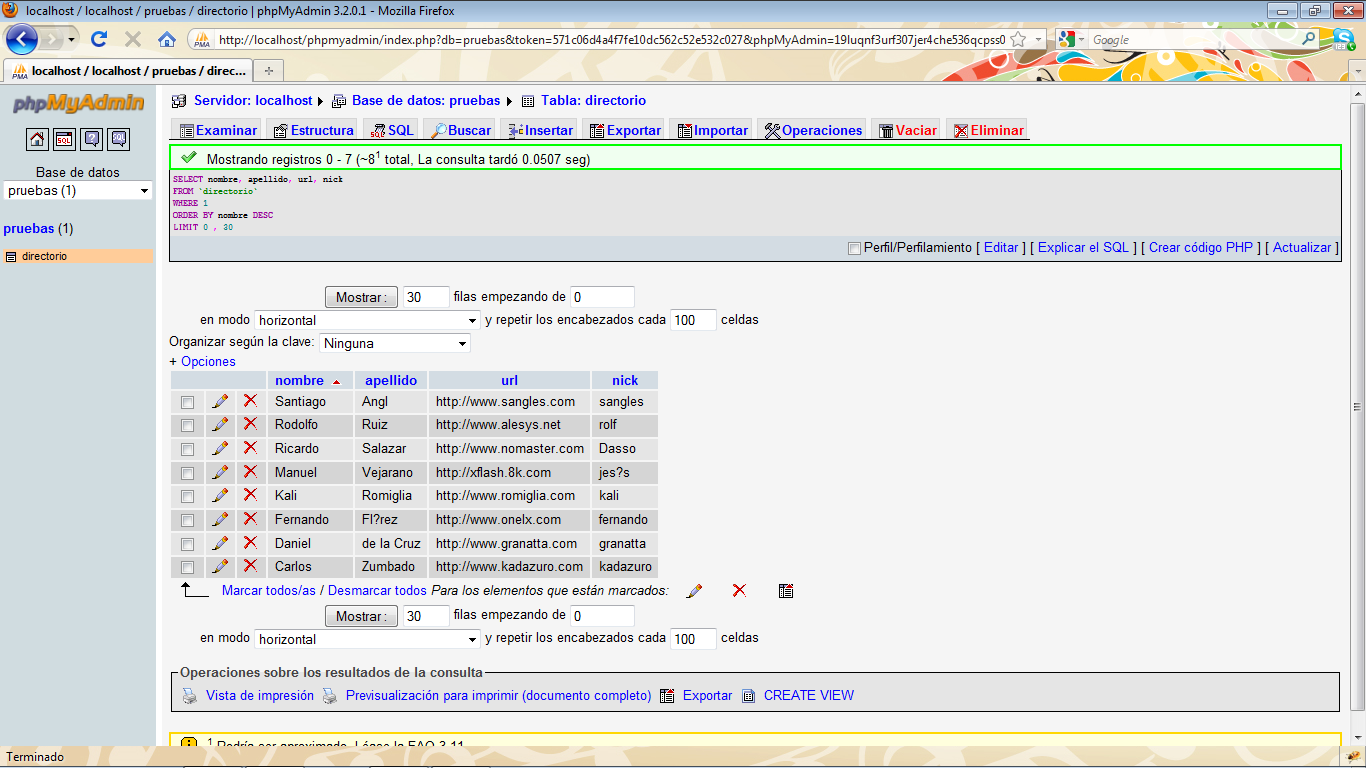
**Ordenación de los resultados**

Además de obtener los campos deseados en la consulta, también podemos ordenarlos en orden ascendente (ASC) o descendente (DESC), referenciando a una columna o a varias, para lo cual agregaremos ORDER BY [columna] [modo de ordenamiento] a la consulta, resultando entonces lo siguiente:

SELECT nombre\_columna\_1, nombre\_columna\_2, …, nombre\_columna\_N FROM [nombre\_tabla] ORDER BY [nombre\_columna] [modo de ordenamiento];

Si queremos seleccionar registros en nuestra tabla *directorio* tomando como referencia al campo *nombre*, y en orden descendente, teclearíamos:

SELECT nombre, apellido, url, nick FROM directorio ORDER BY nombre DESC;



**Figura 3.10.**  
*Selección y ordenamiento de resultados.*

Puede practicar las selecciones ordenando por distintas columnas. Para ordenar por más de una columna, puede adjuntar el nombre del campo y el orden, seguido cada uno de ellos por la instrucción ORDER BY, por ejemplo:

SELECT nombre, apellido, url, nick FROM directorio ORDER BY nombre ASC, url DESC;

**Selección de datos específicos**

En caso de que deseemos realizar una consulta a la base de datos buscando los registros que concuerden con un criterio específico, utilizaremos la cláusula WHERE, que se suele utilizar de la siguiente forma:

SELECT [nombre\_columnas] FROM [nombre\_tabla] WHERE [nombre\_columna] [criterio\_comparación] [valor];

donde *nombre\_columna* es una columna de la tabla, *criterio\_comparación* es el lugar en el que indicamos a la base de datos que seleccione los campos donde la información sea "igual", "mayor" o "menor" o que contenga cierto valor, y *valor* es el criterio de búsqueda de la selección que estamos llevando a cabo.

Por ejemplo, si quisiéramos seleccionar todas las columnas de la tabla *directorio* donde el nombre sea *Ricardo*, teclearíamos lo siguiente:

SELECT \* FROM directorio WHERE nombre = 'Ricardo';

Si deseamos seleccionar todas las columnas de la tabla *directorio* donde *nombre* comienza con la letra *R*, teclearíamos lo siguiente:

SELECT \* FROM directorio WHERE nombre LIKE 'R%';

Si deseamos seleccionar todas las columnas de la tabla *directorio* donde *nombre* contenga la letra *r*, teclearíamos lo siguiente:

SELECT \* FROM directorio WHERE nombre LIKE '%R%';

Observe los registros de la tabla *directorio* y practique realizando distintos tipos de consultas de selección como las anteriores.

**Mostrar un determinado número de registros**

Puede que cuando realizamos una consulta no queramos mostrar todos los registros resultantes en la búsqueda, sino simplemente un número determinado de ellos. Para este tipo de consultas se utilizan lo que se denominan *límites*, y la sintaxis para usarlos es la siguiente:

SELECT [campos] FROM [tabla] [WHERE] LIMIT [inicio],[cantidad];

siendo *inicio* el índice en el *array* resultante de respuestas y *cantidad* el número de registros a mostrar.

Por ejemplo, si queremos mostrar solamente los dos primeros registros de la tabla *directorio*, teclearíamos:

SELECT \* FROM directorio LIMIT 0,2;

Si queremos mostrar los dos registros siguientes, la consulta a realizar sería:

SELECT \* FROM directorio LIMIT 2,2;

**Nota:** Recuerde que puede pensar en un *array* como en uno de esos grandes ficheros con cajones, dentro de cada uno de los cuales hay información guardada sobre un tema concreto. Los *arrays* son estructuras que sirven para almacenar más de un valor al mismo tiempo, en distintas celdas, y disponen de un índice, un número identificador que se usa para acceder a cada uno de los datos almacenados. Este índice es un número que va desde 0 hasta *número\_valores\_del\_array – 1*; así, si un *array* guarda seis valores, el índice de dicho *array* va desde 0 hasta 5. Por ello, los dos primeros elementos de un *array* están almacenados en la posición 0 y 1, de forma que cuando queremos escoger los dos primeros registros de la consulta comenzamos por el número 0.

Obsérvese como se siguen mostrando dos registros tras la búsqueda, pero al variar el valor de *inicio*, el lugar a partir del cual se empiezan a mostrar los registros también varía. Este tipo de consultas (utilizando *límites*) es muy útil cuando deseamos realizar paginaciones de resultados.

### 3.2.5. Modificación de registros almacenados en una tabla

Para actualizar o modificar el contenido de alguno de los registros almacenados en la tabla se utiliza como referencia el campo cuyo identificador es único para cada registro. En nuestra tabla *directorio*, el campo *id* es único por ser autoincremental, por lo que nos ayuda a especificar cuál es el registro que deseamos modificar.

La sintaxis para realizar modificaciones es la siguiente:

|  |
| --- |
| UPDATE [nombre\_tabla] SET nombre\_campo1=valor1, nombre\_campo2=valor2, …, nombre\_campoN=valorN [WHERE] [nombre\_columna] [criterio\_comparación] [valor]; |

Inserte el siguiente registro en la tabla *directorio* tal y como hicimos con anterioridad:

|  |
| --- |
| INSERT INTO directorio (nombre, apellido, email, url, nick) VALUES ('Juan', 'Perez', 'info@juanperez.com', 'jperez'); |

A continuación seleccionaremos el registro que acabamos de insertar. Esto podemos hacerlo de varias formas, seleccionando en la tabla al usuario cuyo nombre es *Juan*, por ejemplo:

|  |
| --- |
| SELECT \* FROM directorio WHERE nombre = 'Juan'; |

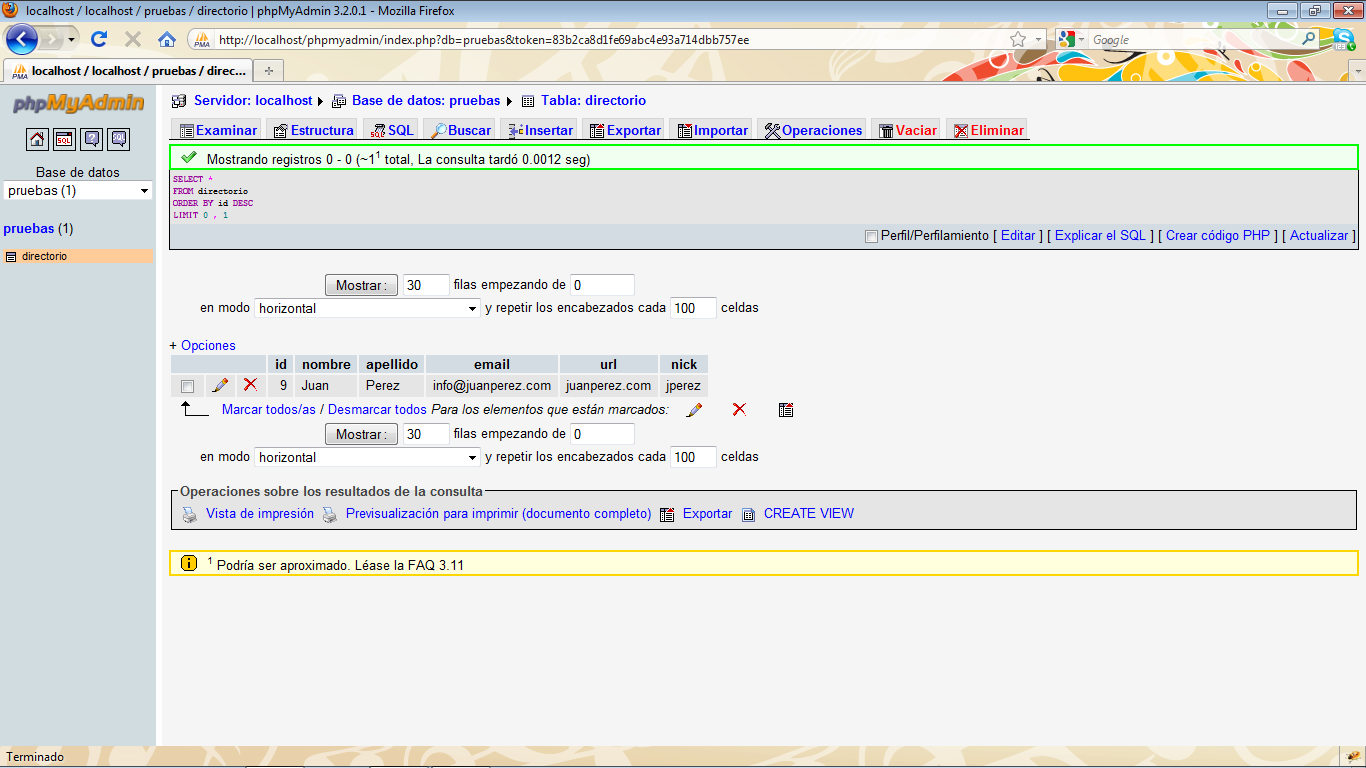
También podemos seleccionar de la tabla *directorio* el registro en el que el valor del campo *id* sea el mayor (al ser este campo autoincremental el último registro será el que tenga mayor *id*), como se muestra en la figura 3.11.

SELECT \* FROM directorio ORDER BY id DESC LIMIT 0,1;

Si se ha fijado, el nuevo registro que hemos añadido a la tabla tiene la particularidad de que algunos de los contenidos de sus campos están incompletos (en comparación con los datos almacenados de los demás registros); el apellido de *Juan* carece de tilde, y su *url* no contiene la cadena *http://www.* que sí tienen el resto de registros, por lo que hemos de modificar los datos que hemos almacenado previamente (concretamente los campos *apellidos* y *url*) utilizando el campo *id* como identificador único.

Para ello teclearemos:

UPDATE directorio SET  
apellido='Pérez', url='http://www.juanperez.com' WHERE id=9;

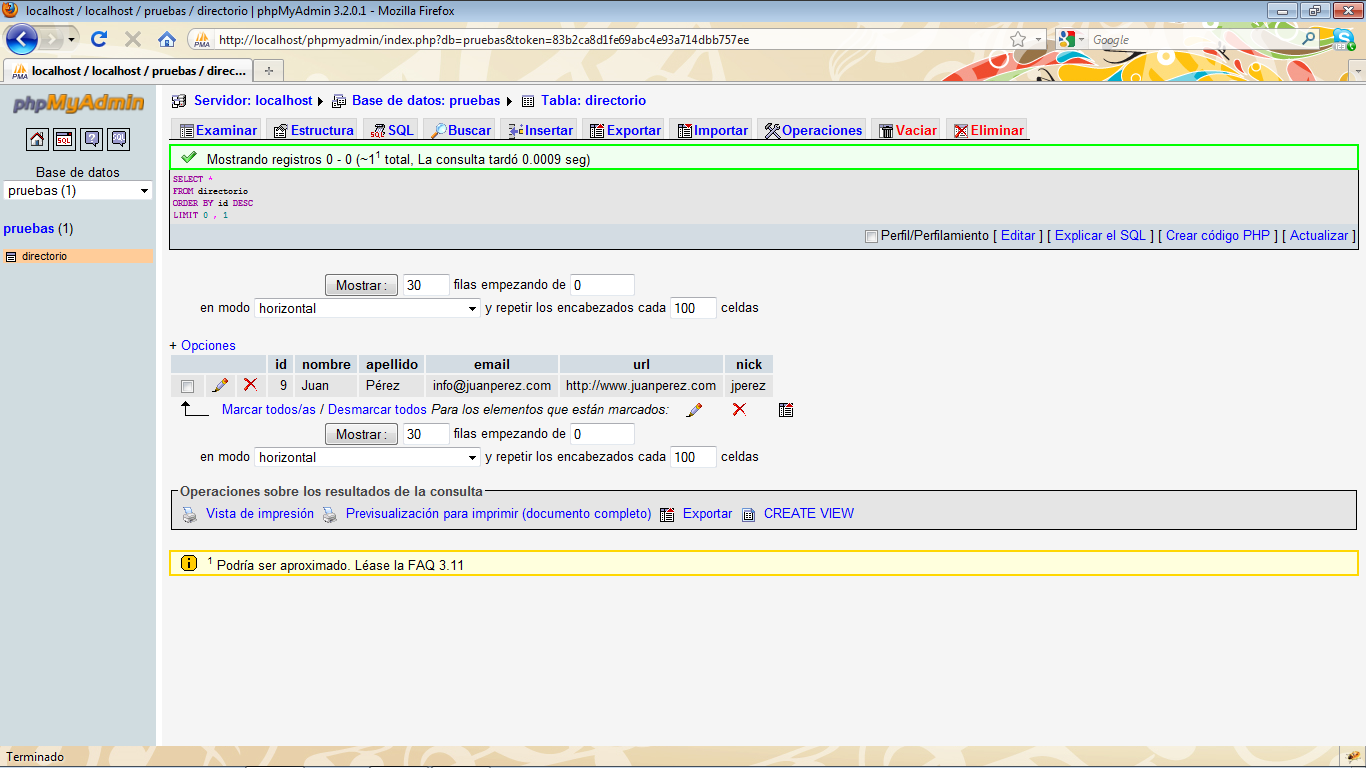


**Figura 3.11.**  
*Selección del último registro insertado.*

Ejecute la consulta a la base de datos, y seleccione de nuevo el registro utilizando como valor el campo *id* el del último registro insertado en la tabla:

SELECT \* FROM directorio ORDER BY id DESC LIMIT 0, 1;

Como puede observar en la figura 3.12, los datos se han modificado con éxito, con lo que el último registro añadido ya dispone de la información correcta.



**Figura 3.12.**  
*Modificación del último registro añadido a la tabla directorio.*

El utilizar un identificador único para realizar las modificaciones es fundamental. En el supuesto de que omitiera usar dicho identificador (*id* en este ejemplo), la consulta a la base de datos se haría de la siguiente forma:

UPDATE directorio SET  
apellido='Pérez', url='http://www.juanperez.com';

Y como resultado de esta acción, todos los registros almacenarían en su campo *apellido* el valor *Pérez* y en su campo *url* el valor [*http://www.juanperez.com*](http://www.juanperez.com).

### 3.2.6. Eliminación de registros almacenados en la tabla

Para realizar el borrado de un registro se utiliza la siguiente sintaxis:

DELETE FROM [tabla] WHERE [cláusula where];

Para borrar el registro cuyo identificador *id* es 9, la consulta a efectuar sería:

DELETE FROM directorio WHERE id = 9;

**Nota:** Es muy importante hacer referencia a una columna con un identificador único a la hora de realizar la consulta. De igual forma a como ocurría en una consulta de actualización, si se omite la cláusula *where* al efectuar una operación de borrado, la consulta afecta a todos los registros, por lo que en este caso serían eliminados todos ellos.

# Unidad 4: MySQL y PHP.

En las dos unidades anteriores hemos aprendido un uso básico de MySQL, así como algunas características del lenguaje PHP. Ahora aprenderemos a utilizar ambos de forma conjunta, realizando los mismos ejercicios que la unidad anterior pero utilizando PHP para comunicarnos con la base de datos y mostrar los resultados en la ventana de su navegador. Además utilizaremos el framework CSS Bootstrap para la presentación de las páginas.

PHP dispone de funciones predefinidas específicas para comunicarse de forma efectiva con MySQL mediante un proceso que por lo general suele desarrollarse pasando por las siguientes etapas:

* Conexión de PHP a la base de datos MySQL.
* Realización de una consulta a la base de datos MySQL (SQL *query*).
* Impresión en pantalla o procesamiento de los datos obtenidos.
* Liberación de resultados y cierre de la conexión con la base de datos MySQL.

MySQL cierra automáticamente las conexiones con la base de datos una vez procesado el documento .php que la abrió, pero para estar más seguros debemos siempre cerrar todas las conexiones que establezcamos.

## 4.1. Ficheros de configuración y acciones repetitivas

Cada vez que deseemos comunicarnos con MySQL a través de documentos escritos en PHP hay que establecer una conexión en la que necesitaremos cuatro variables, las cuales, por lo general, se utilizarán en todos los documentos PHP del sitio o proyecto que se esté realizando. Para utilizar estas variables disponemos de dos posibilidades:

* Definir las variables en cada documento PHP.
* Definir las variables en un fichero externo que es incluido para su uso en cada documento PHP.

Ambas opciones funcionan correctamente, y el uso es el mismo, pero a la hora de hacer modificaciones (en el nombre de la base de datos, la contraseña o el nombre de usuario) será más fácil y efectivo realizarlas una sola vez sobre el fichero externo que realizar la modificación una vez por cada documento PHP que vaya a utilizar.

Por otra parte, cada vez que necesite conectarse desde un documento .php a la base de datos MySQL, realizará el mismo proceso una y otra vez; de hecho, al transcurrir el tiempo, nos percataremos de que utilizamos una y otra vez las mismas funciones, por lo que es recomendable guardarlas en archivos externos de forma que nuestros proyectos dispongan de una mejor organización y sean más fáciles de actualizar o corregir en caso de que existan fallos (es un buen método para centralizar los posibles errores; puesto que agrupamos las funciones en un fichero concreto; en caso de producirse un fallo sólo habremos de buscar el error en dicho fichero y, cuando sea subsanado, todos los documentos .php que hacen uso de este fichero externo funcionarán correctamente).

Vamos a crear ahora en la raíz del servidor local una carpeta de nombre *phpmysql*, y dentro de ésta crearemos otra de nombre *includes*, en la que vamos a almacenar todos los archivos de uso común y de configuración.

## 4.2. Obtener siempre una respuesta

Es importante y conveniente que el documento .php siempre imprima algo, incluso en el caso de que no se haya encontrado resultado alguno o no haya podido establecerse conexión con la base de datos. Para tal efecto, podemos utilizar la función *die*, siendo su sintaxis como sigue:

die ("mensaje");

De esta forma, cuando se realice una consulta o conexión a la base de datos, podemos establecer el uso de esta función como alternativa en caso de error. Por ejemplo:

|  |
| --- |
| $resultado = mysql\_connect("servidor","usuario","clave") or die("No se pudo conectar al servidor");  $resultado = mysql\_query("SELECT \* FROM tabla") or die("No se pudo conectar al servidor"); |

Si se produce un error, ambos ejemplos mostrarán el texto "No se pudo conectar al servidor". Si lo que desea saber es el error exacto que se produjo, puede hacer uso de la función *mysql\_error( )*, que retornará el error que se produjo:

|  |
| --- |
| $resultado = mysql\_connect("servidor","usuario","clave") or die(mysql\_error( ));  $resultado = mysql\_query("SELECT \* FROM tabla") or die(mysql\_error( )); |

El uso de la función *die* le indica a PHP que termine de interpretar el documento en el que se encuentra después de imprimir el mensaje; es decir, una vez impreso, se terminan los procesos de la página.

Un aspecto a tener en cuenta es el hecho de que la función *mysql\_error( )* proporciona el error exacto que se produjo en MySQL, lo cual es de gran ayuda cuando se está desarrollando el proyecto. Sin embargo, no es necesario suministrar tanta información al usuario que esté navegando por nuestro sitio, por lo que es conveniente, una vez terminado el trabajo y comprobado que todo funciona correctamente, sustituir nuestros mensajes de error por otros más sencillos en los que no se den muchos detalles.

## 4.3. Uso de MySQL con PHP

### 4.3.1. Establecer una conexión con MySQL

Para establecer una conexión con MySQL se requieren el nombre del servidor, el nombre del usuario y una contraseña, información que proporciona el administrador del servicio cuando se contrata un servicio de hospedaje en la Web. La función que se utiliza es *mysql\_connect*, cuya sintaxis es la siguiente:

|  |
| --- |
| $cnx = mysql\_connect ("servidor", "usuario", "contraseña"); |

Si hemos de especificar que el servidor es accedido por un puerto en concreto, hemos de adjuntarlo al nombre del servidor separado por el carácter ":" :

|  |
| --- |
| $cnx = mysql\_connect ("servidor:3306", "usuario", "contraseña"); |

El identificador del enlace que se establece se almacena, en este caso, en la variable *cnx*, aunque puede establecer el nombre de variable que usted desee. En caso de no poder conectar con la base de datos es conveniente que tanto PHP como MySQL nos mantengan informados de lo que está ocurriendo, utilizando para ello la función *die* de la que hablamos con anterioridad, cuya sintaxis, mostramos de nuevo:

|  |
| --- |
| $cnx = mysql\_connect ("servidor", "usuario", "contraseña") or die (mysql\_error( )); $cnx = mysql\_connect ("servidor", "usuario", "contraseña") or die ("No se pudo conectar con el servidor); |

Por ejemplo, si es usted el administrador del sitio local en el que está haciendo las pruebas puede tratar de conectarse al servidor utilizando la siguiente línea de código:

|  |
| --- |
| $cnx = mysql\_connect ("localhost", "root", "root") or die (mysql\_error( )); |

### 4.3.2. Selección de la base de datos

Una vez establecida con éxito la conexión con MySQL, hemos de seleccionar la base de datos con la que vamos a trabajar, para lo que utilizaremos la función *mysql\_select\_db*, cuya sintaxis es la siguiente:

|  |
| --- |
| mysql\_select\_db (base de datos, identificador de conexión); |

Por ejemplo, para seleccionar la base de datos de nombre *pruebas* (que utilizamos con anterioridad en la unidad 3 dedicada a MySQL) utilizando el identificador de conexión almacenado en la variable *$cnx* utilizaríamos la siguiente línea:

mysql\_select\_db ("pruebas", $cnx);

### 4.3.3. Creación de los ficheros de configuración

Por lo general (no siempre) trabajaremos conjuntamente con una base de datos y un servidor de bases de datos, por lo que es conveniente crear archivos de configuración, como ya se comentó anteriormente en este mismo capítulo. Vamos a utilizar dos ficheros, uno de ellos almacenará las variables de configuración y el otro lo utilizaremos para agrupar las funciones de uso común.

En el primero de ellos, que puede encontrar con el nombre *config.php* en la carpeta *material/cap5/php/includes/* del CD, especificaremos los valores de *servidor, usuario, contraseña* y *base de datos* en distintas variables, y aunque posteriormente agreguemos más información a este fichero, de momento el contenido será el siguiente:

|  |
| --- |
| <?php $HOSTNAME = "localhost"; //SERVIDOR $USERNAME = "root"; //USUARIO $PASSWORD = "root"; //CONTRASEÑA $DATABASE = "pruebas"; //BASE DE DATOS ?> |

Guarde este documento con el nombre *config.php* en la carpeta *phpmysql/includes* de su servidor local, que creó anteriormente, y abra un nuevo documento de texto para crear el segundo archivo.

Este segundo archivo, y aunque con posterioridad le añadamos más funciones que utilicemos a medida que las necesitemos, va a comenzar conteniendo funciones relativas al hecho de que cada vez que queramos obtener o modificar información de la base de datos vamos a tener que conectarnos con ella, lo que se producirá en todos los documentos .php que utilicemos, por lo que vamos a declarar una función que nos sirva para establecer una conexión con la base de datos:

|  |
| --- |
| <?php /\*\*\*  función conectar que se conecta a MySQL y devuelve el identificador de conexión \*\*\*/ function conectar ( ) {  global $HOSTNAME, $USERNAME, $PASSWORD, $DATABASE;  $idcnx = mysql\_connect ($HOSTNAME, $USERNAME, $PASSWORD) or  die (mysql\_error( ));  mysql\_select\_db ($DATABASE, $idcnx);  return $idcnx; }  ?> |

Este archivo, del que puede encontrar una copia de nombre *funciones.php* en la carpeta *material/cap5/php/includes/* del CD, tiene algunas características que pasamos a comentar:

* La función contiene comentarios, lo cual es de utilidad en proyectos grandes, ya que con leer el comentario sabemos para qué sirve la función y no tenemos por qué analizarla para averiguarlo.
* La función toma los valores de las variables del archivo *config.php*, que son globales y externas a la función.

Guarde ahora el documento con el nombre *funciones.php* en la carpeta *phpmysql/includes* de su servidor local. Ahora que dispone de los dos ficheros, y teniendo en cuenta que *funciones.php* necesita a *config.php* para utilizar las variables, es fácil hacerse una idea del orden en el que hay que incluir estos ficheros en las llamadas para que sean utilizados: primero el de configuración y, seguidamente, el de funciones.

### 4.3.4. Realización de consultas a MySQL

Establecida la conexión con MySQL y seleccionada la base de datos con la que vamos a trabajar podemos realizar las consultas que queramos a MySQL, para lo que usamos la función *mysql\_query*, cuya sintaxis es la siguiente:

*$resultado = mysql\_query (Consulta SQL, identificador de conexión);*

Si no especificamos un identificador de conexión, MySQL asume que se utiliza el último creado.

Por ejemplo, para realizar una consulta que obtenga todas las filas de la tabla *directorio* y las ordene en orden descendente en base al valor *id*, utilizaríamos:

*$res = mysql\_query ("Select \* FROM directorio ORDER BY id desc") or die (mysql\_error( ));*

El resultado de la consulta se almacena en la variable *$res*.

### 4.3.5. Impresión de resultados

Hay varias formas de procesar los resultados que devuelve MySQL, pero antes de hacerlo siempre es conveniente verificar si se obtuvieron resultados y no se produjo ningún error. Al retornar MySQL el resultado en forma de filas, nos basta con comprobar si dicho número de filas es mayor que cero, para lo que utilizamos la función *mysql\_num\_rows*, cuya sintaxis es:

*mysql\_num\_rows (identificador de resultado);*

Puede ver un ejemplo en el documento *material/cap5/php/mysql\_num\_rows.php* del CD, en el que encontrará el siguiente código:

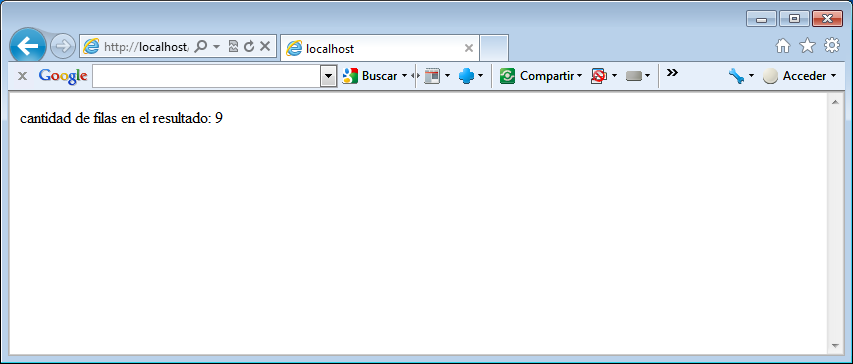
|  |
| --- |
| 1 <?php  2 include ("includes/config.php");  3 include ("includes/funciones.php");  4  5 //nos conectamos a mysql  6 $cnx = conectar ();  7 //consulta.  8 $sql = "SELECT \* FROM directorio ORDER BY id DESC";  9 $res= mysql\_query($sql) or die (mysql\_error());  10 if( mysql\_num\_rows($res) >0){  11 echo "cantidad de filas en el resultado: " . mysql\_num\_rows($res);  12 }else{  13 echo "no se obtuvieron resultados";  14 }  15 mysql\_close($cnx);  16 ?> |

**Nota:** Hemos numerado las líneas para poder hacer referencia más rápidamente a las actividades que desarrolla cada fichero .php, pero omítalas a la hora de crear los suyos propios.

En las líneas 2 y 3 se incluyen los ficheros de configuración, y en la 6 se realiza una llamada a la función *conectar* (que se encuentra declarada en el fichero *funciones*, como vimos al crear los ficheros de configuración), asignando entonces el identificador de conexión a una variable de nombre *$cnx*.

En la línea 8 almacenamos la consulta en una variable llamada *$sql*, que utilizamos en la línea 9 para realizar la consulta SQL a MySQL. En caso de que se produzca un fallo, se imprime cuál fue el error concreto ("or die(mysql\_error( )").

En las líneas 10-14 encontramos una estructura de condición en la que se revisa mediante una sentencia *if* si el número de columnas obtenido una vez realizada la consulta es mayor que cero. De ser así, se imprime un mensaje con la cantidad de columnas obtenidas (figura 4.1), y si no hay resultado alguno, se muestra otro mensaje informando de tal circunstancia.



**Figura 4.1.**  
*Cantidad de filas obtenidas en el resultado de la consulta SQL.*

Finalmente, en la línea 15 cerramos la conexión con la base de datos, utilizando para ello el identificador de conexión.

Podemos proceder de varias maneras si encontramos resultados para imprimir, utilizando para ello las distintas funciones predeterminadas que PHP proporciona para tal efecto. A continuación mostraremos las de uso más generalizado, aunque el uso de las mismas se hace totalmente al gusto de cada cual.

La extracción de datos suele realizarse utilizando un bucle de tipo *while*, asignando para las filas una variable que es accedida para imprimir los resultados; dependiendo del método que se utilice, la variable podrá disponer de la información de la fila en forma de *array* (*array* de lista u objeto), siendo en este aspecto donde radica la mayor diferencia entre las distintas formas de imprimir resultados. Como ejemplo, obtendremos los contenidos de las columnas *nick, email* y *url* de la tabla *directorio* de la base de datos de nombre *pruebas* utilizando cada uno de los métodos disponibles, que iremos explicando al darles uso.

Como base para la impresión, vamos a utilizar un fichero cuyo contenido es el siguiente:

|  |
| --- |
| 1 <?php  2 include ("includes/config.php");  3 include ("includes/funciones.php");  4  5 //nos conectamos a mysql  6 $cnx = conectar ();  7 //consulta.  8 $sql = "SELECT nick,email,url FROM directorio ORDER BY id ASC";  9 $res= mysql\_query($sql) or die (mysql\_error());  10 if( mysql\_num\_rows($res) >0){  11 //impresión de los datos.  12 }else{  13 echo "no se obtuvieron resultados";  14 }  15 mysql\_close($cnx);  16 ?> |

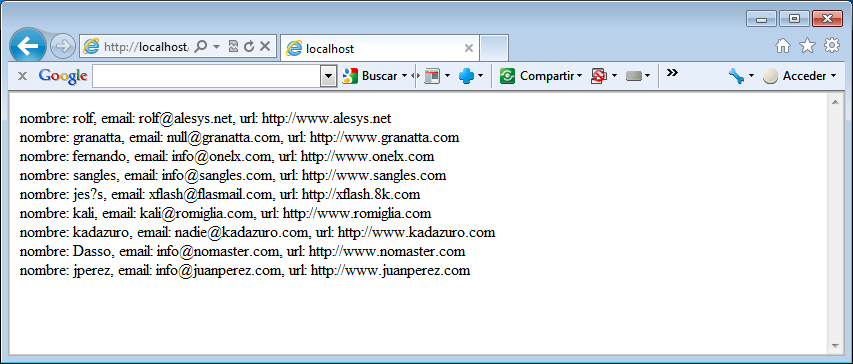
A partir de la línea 11 de este fichero (del que puede encontrar una copia de nombre *baseImpresionResultados.php* en la carpeta *material/cap5/php/* del CD), es el lugar en el que insertaremos las distintas formas de impresión de datos.

#### 4.3.5.1. Impresión de resultados usando un *array*

Mediante la función *mysql\_fetch\_row* se procesa el resultado como un *array*, donde cada uno de los índices del *array* es una columna del resultado. Para conseguir esto, es útil saber la cantidad de columnas de la tabla (en el caso de que en la consulta se hayan seleccionado todas ellas), siendo el orden en el que aparezcan los campos tras la consulta lo que defina el índice del *array*:

|  |
| --- |
| 1 <?php  2 include ("includes/config.php");  3 include ("includes/funciones.php");  4  5 //nos conectamos a mysql  6 $cnx = conectar ();  7 //consulta.  8 $sql = "SELECT nick,email,url FROM directorio ORDER BY id ASC";  9 $res= mysql\_query($sql) or die (mysql\_error());  10 if( mysql\_num\_rows($res) >0){  11 //impresión de los datos.  12 while ($fila = mysql\_fetch\_row($res)) {  13 echo "nombre: ". $fila[0] . ", email: " . $fila[1] . ", url: " . $fila[2] . "<br>";  14 }  15 }else{  16 echo "no se obtuvieron resultados";  17 }  18 mysql\_close($cnx);  19 ?> |

Observe cómo en la línea 12 se accede a las columnas en el resultado mediante la expresión *$fila[número\_columna]*. Posteriormente, se concatenan las cadenas de texto obtenidas y un salto de línea mediante el *tag* *<br>* de HTML para imprimir un resultado por cada fila. Si escribe el código en un nuevo documento de texto o modifica el fichero base para añadir código a partir de la línea 11 y lo guarda con el nombre de *mysql\_fetch\_row.php* en la carpeta *phpmysql* de su servidor (también puede utilizar el documento *material/cap5/php/mysql\_fetch\_row.php* del CD), al abrir el documento en su navegador tecleando [*http://localhost/phpmysql/mysql\_fetch\_row.php*](http://localhost/phpmysql/mysql_fetch_row.php), obtenemos una impresión en pantalla como la de la figura 4.2.



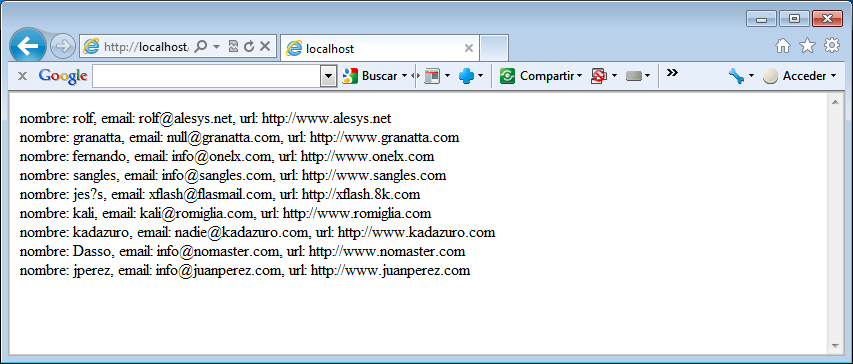
**Figura 4.2.**  
*Impresión de resultados usando* mysql\_fetch\_row.

#### 4.3.5.2. Impresión de resultados usando un *array* de lista

Para imprimir los resultados utilizando un *array* de lista, puede utilizar la función *mysql\_fetch\_array*, donde cada columna presente en la consulta es un índice del *array*. Esta circunstancia es de gran utilidad cuando hemos de seleccionar todas las columnas de la tabla, ya que no hemos de recordar los índices de las columnas sino que es suficiente con recordar los nombres de las mismas. El siguiente código (que puede encontrar en el documento *material/cap5/php/mysql\_fetch\_array.php* del CD) selecciona las columnas *nick, email* y *url* en la consulta SQL, siendo esos nombres los que se utilizan como índices del *array* a la hora de extraer la información (línea 13):

|  |
| --- |
| 1 <?php  2 include ("includes/config.php");  3 include ("includes/funciones.php");  4  5 //nos conectamos a mysql  6 $cnx = conectar ();  7 //consulta.  8 $sql = "SELECT nick,email,url FROM directorio ORDER BY id ASC";  9 $res= mysql\_query($sql) or die (mysql\_error());  10 if( mysql\_num\_rows($res) >0){  11 //impresión de los datos.  12 while ($fila = mysql\_fetch\_array($res)) {  13 echo "nombre: ". $fila['nick'] . ", email: " . $fila['email'] . ", url: " . $fila['url'] . "<br>";  14 }  15 }else{  16 echo "no se obtuvieron resultados";  17 }  18 mysql\_close($cnx);  19 ?> |

Si ahora crea un documento con este código, lo guarda en la carpeta *phpmysql* de su servidor con el nombre *mysql\_fetch\_array.php* y lo abre en su navegador tecleando [*http://localhost/phpmysql/mysql\_fetch\_array.php*](http://localhost/phpmysql/mysql_fetch_array.php), obtendrá la información deseada del mismo modo en que se muestra en la figura 4.3.



**Figura 4.3.**  
*Impresión de resultados usando* mysql\_fetch\_array.

#### 4.3.5.3. Impresión de resultados usando un objeto

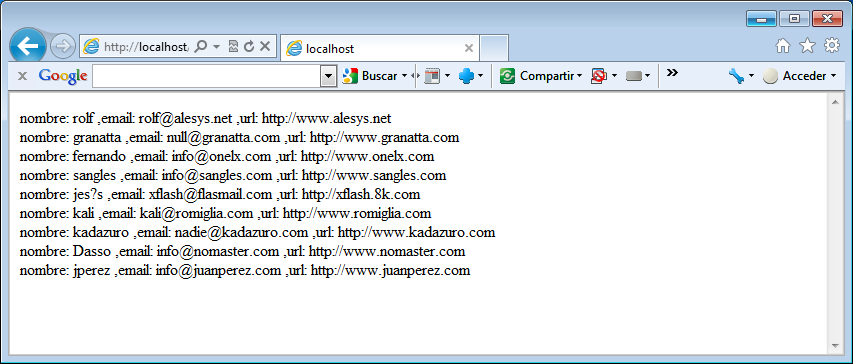
Utilizando la función *mysql\_fetch\_object*, cada columna de la fila es una propiedad de un objeto. En los ejemplos anteriores, las variables a las que se asignaba el resultado actual en el bucle *while* eran estructuras de tipo *array*, por lo que se accedía a la información utilizando índices (numerales o nominales) por medio de los caracteres "[" y "]". En este caso, la variable que recoge la información es un objeto cuyas propiedades (que llevarán el nombre de las columnas) son accedidas por medio del operador ->. Por ejemplo, la propiedad *nombre* del objeto *dato* es accedida mediante la expresión:

*$dato->nombre;*

El siguiente código (que puede encontrar en el documento *material/cap5/php/mysql\_fetch\_object.php* del CD) extrae la información de la base de datos y la imprime utilizando un objeto:

|  |
| --- |
| 1 <?php  2 include ("includes/config.php");  3 include ("includes/funciones.php");  4  5 //nos conectamos a mysql  6 $cnx = conectar ();  7 //consulta.  8 $sql = "SELECT nick,email,url FROM directorio ORDER BY id ASC";  9 $res= mysql\_query($sql) or die (mysql\_error());  10 if( mysql\_num\_rows($res) >0){  11 //impresión de los datos.  12 while ($fila = mysql\_fetch\_object($res)) {  13 echo "nombre: $fila->nick ,email: $fila->email ,url: $fila->url <br>";  14 }  15 }else{  16 echo "no se obtuvieron resultados";  17 }  18mysql\_close($cnx);  19 ?> |

Observe cómo en la línea 13 se accede a las distintas propiedades del objeto *$fila*. Al guardar este fichero con el nombre de *mysql\_fetch\_object.php* en la carpetea *phpmysql* de su servidor y abrirlo posteriormente en su navegador tecleando [*http://localhost/phpmysql/mysql\_fetch\_object.php*](http://localhost/phpmysql/mysql_fetch_object.php), obtendrá la información deseada del mismo modo en que se muestra en la figura 4.4.



**Figura 4.4.**  
*Impresión de resultados usando* mysql\_fetch\_object.

La última forma que veremos de procesar los resultados es mediante el uso de la función *list* (lista) junto con las funciones *mysql\_fetch\_array*  o *mysql\_fetch\_row*. De este modo, en la lista se pasan como parámetros los nombres de las variables que deseemos para cada columna, y a la hora de imprimir los resultados utilizaremos dichos nombres.

**Nota:** A la hora de utilizar la función *list*, recuerde siempre establecer tantos parámetros como columnas tenga la consulta SQL.

El siguiente código (que puede encontrar en el documento *material/cap5/php/mysql\_fetch\_list.php* del CD) extrae la información de la base de datos y la imprime utilizando conjuntamente las funciones *list* y *mysql\_fetch\_array*:

|  |
| --- |
| 1 <?php  2 include ("includes/config.php");  3 include ("includes/funciones.php");  4  5 //nos conectamos a mysql  6 $cnx = conectar ();  7 //consulta.  8 $sql = "SELECT nick,email,url FROM directorio ORDER BY id ASC";  9 $res= mysql\_query($sql) or die (mysql\_error());  10 if( mysql\_num\_rows($res) >0){  11 //impresión de los datos.  12 while (list($nick,$email,$url) = mysql\_fetch\_array($res)) {  13 echo "nombre: ". $nick . ", email: " . $email . ", url: " . $url . "<br>";  14 }  15 }else{  16 echo "no se obtuvieron resultados";  17 }  18 mysql\_close($cnx);  19 ?> |

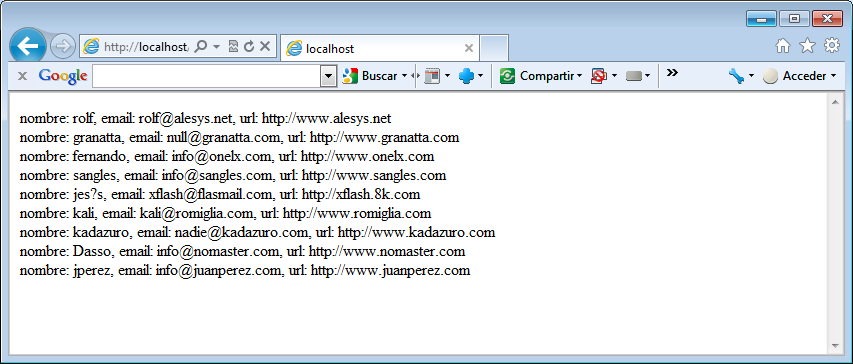
Observe en la línea 8 cómo la cantidad de variables es equivalente al número de columnas del resultado de la consulta. En la línea 12 puede observar cómo se asigna el resultado de la consulta a las variables utilizando la función *list*. Una de las ventajas de esta forma de impresión de datos es que puede mostrar los resultados en una sola línea sin necesidad de concatenar la cadena, de forma que:

*echo "nombre: $nick, email: $email, url: $url<br>";*

la expresión anterior retorna el mismo resultado que la expresión que veníamos utilizando hasta ahora:

*echo "nombre: ". $nick . ", email: ". $email . ", url: ". $url . "<br>";*

Guarde este fichero con el nombre de *mysql\_fetch\_list.php* en la carpeta *phpmysql* de su servidor y ábralo posteriormente en su navegador tecleando [*http://localhost/phpmysql/mysql\_fetch\_list.php*](http://localhost/phpmysql/mysql_fetch_list.php) para obtener la información deseada en la forma que se muestra en la figura 4.5.



**Figura 4.5.**  
*Impresión de resultados utilizando* list *y* mysql\_fetch\_array.

### 4.3.6. Liberación de memoria

Si en un determinado momento se está utilizando de forma intensiva su ordenador o cree estar utilizando más de lo debido la memoria del mismo y desea "aliviarla", puede utilizar la función *mysql\_free\_resulta* para liberar memoria. La sintaxis de uso de la función es la siguiente:

*mysql\_free\_result (identificador de resultado);*

El siguiente ejemplo asigna el resultado a la variable *$res* (línea 9) y libera la memoria utilizada mediante una llamada a la función *mysql\_free\_result* (línea 18):

|  |
| --- |
| 1 <?php  2 include ("includes/config.php");  3 include ("includes/funciones.php");  4  5 //nos conectamos a mysql  6 $cnx = conectar ();  7 //consulta.  8 $sql = "SELECT nick,email,url FROM directorio ORDER BY id ASC";  9 $res= mysql\_query($sql) or die (mysql\_error());  10 if( mysql\_num\_rows($res) >0){  11 //impresión de los datos.  12 while (list($nick,$email,$url) = mysql\_fetch\_array($res)) {  13 echo "nombre: ". $nick . ", email: " . $email . ", url: " . $url . "<br>";  14 }  15 }else{  16 echo "no se obtuvieron resultados";  17 }  18 mysql\_free\_result($res);  19 mysql\_close($cnx);  20 ?> |

### 4.3.7. Cierre de la conexión

PHP está configurado para cerrar las conexiones abiertas con las bases de datos y eliminar las variables utilizadas una vez que termina de ejecutar las páginas con extensión .php. Sin embargo, para asegurarnos de que realmente terminen las conexiones, puede indicarle a PHP que cierre la conexión con MySQL utilizando la función *mysql\_close*, con el identificador de conexión especificado como parámetro en la llamada a la función (como se puede observar, por ejemplo, en la línea 19 del código del aparatado anterior), siguiendo la sintaxis:

*mysql\_close ($cnx);*

## 4.4. Modificación de la información de la base de datos

En este apartado vamos a modificar la información contenida en la tabla *directorio* de la base de datos *pruebas*, lo que lograremos mediante el uso de formularios. También utilizaremos los mismos ficheros de configuración (*config.php* y *funciones.php*) que creamos anteriormente y aplicaremos todos los conceptos explicados hasta ahora de una forma más práctica, de forma que pueda comprobar cómo PHP imprime los resultados obtenidos de MySQL y cómo puede modificarlos.

Los cambios que realizaremos en la tabla son los que se podrían denominar como "genéricos" a la hora de manipular información ya almacenada.

* Listado de los registros.
* Obtención de detalles de un registro.
* Modificación de un registro.
* Inserción de un nuevo registro.
* Borrado de un registro específico.

En la mayoría de casos, la forma en que se procesan formularios consiste en utilizar tres páginas distintas:

* Página con el formulario.
* Página que procesa los datos introducidos en el formulario.
* Página de información de los cambios realizados.

Sin embargo, en este apartado aprenderemos a crear documentos PHP "inteligentes", que consisten en que una sola página .php contendrá el formulario, lo procesará y posteriormente informará del éxito o fracaso de las tareas realizadas. Además, como ya habíamos indicado anteriormente utilizaremos el framework CSS Bootstrap para la presentación de las páginas. Asimismo, recuerde que en la tabla *directorio* utilizábamos un identificador único (la columna *id*), ya que pronto veremos la utilidad de esta columna; pero, previamente, conozcamos las formas en las que las variables pueden ser enviadas cuando se utilizan formularios mediante los métodos POST y GET.

### 4.4.1. Envío de variables usando POST y GET

Cuando envíe variables desde formularios puede utilizar los métodos POST y GET, siendo la diferencia entre ambos que las variables enviadas mediante GET son adjuntadas a la dirección Web que se teclea en la barra de dirección del navegador, al contrario que las variables enviadas mediante POST.

#### 4.4.1.1. Envío de variables mediante GET

El envío de variables mediante GET se realiza de la siguiente forma:

[*http://localhost/phpmysql/variablesGET.php?id=1&user=Juan*](http://localhost/phpmysql/variablesGET.php?id=1&user=Juan)

Observe el carácter ?, que se utiliza para indicar el fin de la dirección Web y el comienzo de las variables enviadas mediante GET y especificadas mediante el carácter &. Cada nombre de variable, junto con el valor de la misma, son lo que se denominan un "par", por lo que suele decirse que las variables de los formularios se obtienen en "pares".

Una forma segura para procesar los formularios consiste en procesar las variables sólo si han sido enviadas mediante el método esperado. Las variables enviadas mediante GET pueden ser escritas directamente en la barra de dirección de su navegador, por lo que no debe utilizarlas cuando desee realizar operaciones "delicadas" (esto es, que contengan información que no desee que un usuario pueda ver, como por ejemplo, el par *nombre\_de\_usuario* y *contraseña* para acceder a un documento concreto de un sitio Web).

Vamos a realizar un pequeño ejemplo, enviando primero las variables mediante GET y posteriormente mediante POST. Abra un nuevo documento de texto y escriba lo siguiente:

*<?php  
echo "Variable id con valor: " . \_GET['id'];  
?>*

Guarde el documento con el nombre *variablesGET.php* en la carpeta *phpmysql* de su servidor local en la que hemos estado trabajando y ábralo tecleando su dirección ([*http://localhost/phpmysql/variablesGET.php*](http://localhost/phpmysql/variablesGET.php)*).*

El mensaje que aparecerá en su pantalla será el siguiente:

*variable id con valor:*

Como puede observar en el código PHP, se está esperando una variable de nombre *id* enviada mediante GET que, cuando sea recibida, será impresa en su pantalla como parte del resultado. Puesto que no hemos especificado el valor de *id* en la llamada del documento .php, no se muestra ningún valor al generarse los resultados. Podemos configurar el documento PHP para que no se imprima nada hasta que la variable $\_GET['id'] exista (haciendo uso de la función *isset*), como puede verse en el documento *material/cap5/php/variablesGET.php* del CD:

|  |
| --- |
| <?php if(isset($\_GET['id']))  echo "variable id con valor: " . $\_GET['id']; ?> |

Modifique su documento *variablesGET.php* con el código anterior, guárdelo y abra la página nuevamente en su navegador, comprobando que PHP no imprime nada en pantalla.

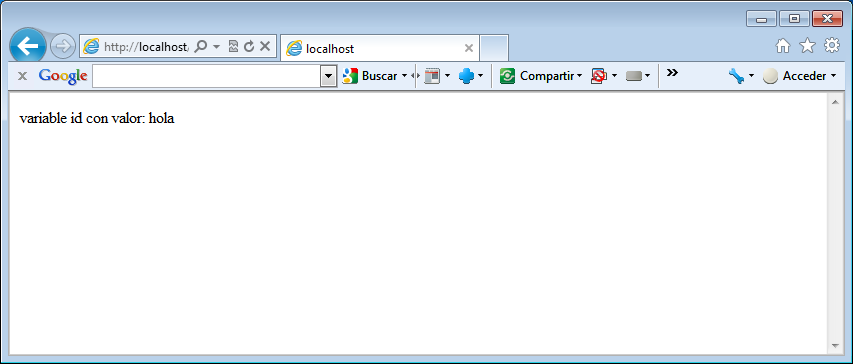
Ahora abra de nuevo el documento en su navegador, pero agregando lo siguiente a la dirección de la página:

"?id=hola"

Con lo que la barra de dirección de su navegador ha de contener la cadena: <http://localhost/phpmysql/variablesGET.php?id=hola>

Pulse la tecla **Intro** para procesar el documento y obtendrá en su pantalla el siguiente texto, tal y como se muestra en la figura 4.6:

*variable id con valor: hola*



**Figura 4.6.**  
*Impresión de variables enviadas mediante GET.*

En este caso, el documento PHP recibió un valor para la variable *id* mediante el método GET, puesto que lo escribimos en la barra de dirección del navegador.

A continuación realizaremos la misma operación pero utilizando un formulario para especificar el valor de la variable *id*. Cree un nuevo documento de texto de nombre *variablesGET\_b.php* (puede encontrar una copia del mismo en el directorio *material/cap5/php/* del CD) y añada este código:

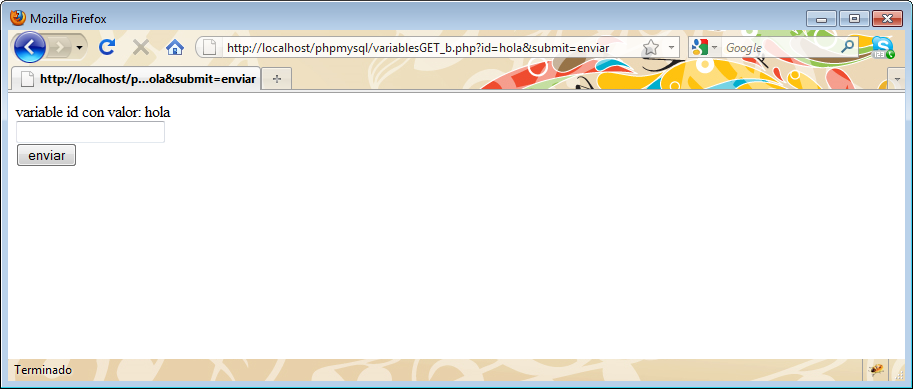
|  |
| --- |
| <?php if(isset($\_GET['id']))  echo "variable id con valor: " . $\_GET['id']; ?>  <form method="GET" action="<?php echo $\_SERVER['PHP\_SELF'];?>"> <input type="text" name="id"><br> <input type="submit" name="submit" value="enviar"> </form> |

Hemos añadido al documento un campo de texto de nombre *id* y un botón de tipo SUMBIT para enviar datos. La forma en que se envían las variables del formulario es GET, y como acción especificamos lo siguiente:

$\_SERVER['PHP\_SELF']

PHP\_SELF es una variable de servidor que retorna la ruta hasta el fichero, lo que nos indica que este documento, que es el que contiene el formulario, también será el que procese la información. Guarde el fichero y ábralo en su navegador tecleando <http://localhost/phpmysql/variablesGET_b.php>.

En este momento podrá ver el campo de texto y el botón, introduzca el texto que desee y presione el botón enviar, obteniendo un resultado que ha de ser similar al de la figura 4.7.



**Figura 4.7.**  
*Impresión de variables enviadas mediante GET vía formulario.*

Observe cómo las variables son visibles (ya que aparecen en la barra de dirección del navegador) y cómo se ha añadido una nueva variable de nombre *submit*, que se corresponde con el nombre del botón, y cuyo valor es en este caso "enviar". Posteriormente veremos la utilidad que podemos aplicar a esta variable cuando procesemos formularios.

Podemos concluir entonces que el envío de variables mediante GET puede obtenerse escribiendo el nombre de las variables en la barra de dirección o bien enviándolas desde un formulario. Más tarde las usaremos en forma de hipervínculo.

#### 4.4.1.2. Envío de variables mediante POST

Para probar el formulario usando POST como método de envío, modifique el código del fichero anterior para que contenga lo siguiente:

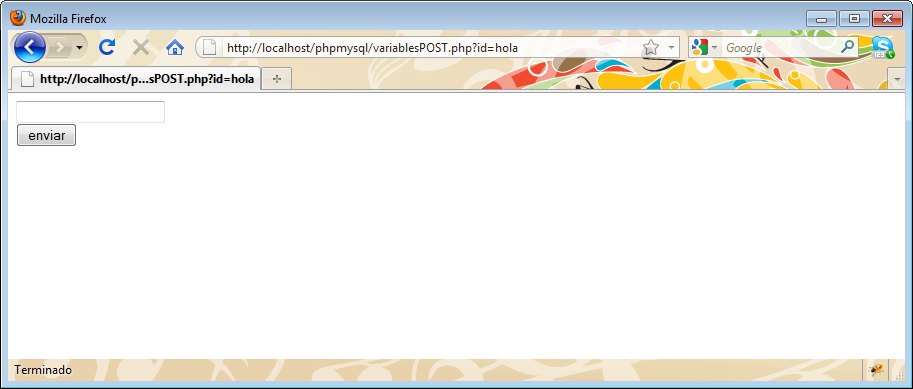
|  |
| --- |
| <?php if(isset($\_POST['id']))  echo "variable id con valor: " . $\_POST['id']; ?>  <form method="POST" action="<?php echo $\_SERVER['PHP\_SELF'];?>"> <input type="text" name="id"><br> <input type="submit" name="submit" value="enviar"> </form> |

Guarde el archive con un nuevo nombre (variablesPOST.php) en la carpeta phpmysql de su servidor local y ábralo en su navegador tecleando <http://localhost/phpmysql/variablesPOST.php>.

En esta ocasión, PHP está esperando a que la variable *id* sea enviada mediante POST, por lo que para comprobar que PHP reconoce el método con el que se envían las variables, teclee el valor para la variable *id* en la barra de dirección del navegador de la misma forma en que lo hicimos anteriormente con GET:

[*http://localhost/phpmysql/variablesPOST.php?id=hola*](http://localhost/phpmysql/variablesPOST.php?id=hola)

Al pulsar la tecla **Intro**, puede observar que PHP no imprime resultados debido a que la variable no es pasada mediante el método que se espera (figura 4.8)



**Figura 4.8.**  
*Variables enviadas mediante GET ignoradas cuando se esperan vía POST.*

Si en cambio introduce un texto en el campo de texto del formulario y pulsa el botón *"enviar"* para enviar los datos, podrá ver cómo el resultado sí es impreso al procesar el formulario.

El envío de variables mediante POST es más seguro, puesto que ni las variables ni los valores que almacenan son visibles en la barra de dirección del navegador, por lo que una vez enviado el formulario no pueden ser modificadas.

### 4.4.2. Bootstrap

Bootstrap es un framework (un conjunto de librerías que facilitan la codificación de cierto lenguaje o tecnología, en este caso CSS) que nos ayuda a estandarizar el diseño y desarrollo de sitios web.

A partir de la versión 3.x ha sido implementado pensando que se adapte tanto a las pantallas de equipos de escritorio como a móviles y tablets.

Bootstrap ha sido desarrollada y es mantenida por la empresa Twitter y la ha liberado como un producto Open Source.

Tiene una filosofía muy intuitiva para el maquetado de sitios web que puede ser rápidamente aprendida por desarrolladores que no vienen del mundo del diseño web.

El corazón de este framework es un archivo CSS que lo podemos descargar del sitio <http://getbootstrap.com>.

Luego debemos enlazar a nuestra página con el archivo bootstrap.min.css y ya podemos utilizar esta librería.

La primera clase que podemos probar es class="container" (esta muestra todo el contenido del div centrado), además el framework ya define estilos por defecto para una gran cantidad de elementos HTML de la página, por ejemplo define la tipografía de tipo Helvética.

El código mínimo de nuestra página es:

<!DOCTYPE html>

<html>

<head>

<title>Prueba de Bootstrap</title>

<link href="css/bootstrap.min.css" rel="stylesheet">

<meta name="viewport" content="width=device-width, initial-scale=1">

</head>

<body>

<div class="container">

<h1>Titulo</h1>

<p>Esto es una prueba de bootstrap.</p>

</div>

</body>

</html>

Además de la clase container, Bootstrap dispone de un sin número de pseudo clases para definir estilos adecuados para todos los elementos html como pueden ser tablas, enlaces y demás. No se indicarán todas estas clases ya que no es el propósito de este folleto. Se recomienda al lector la investigación correspondiente acerca de estas pseudo clases.

A continuación, vamos a formular un procedimiento para listar los datos de la tabla directorio de nuestra base de datos creada con anterioridad.

### 4.4.3. Listado de registros

La forma más fácil de conocer los registros de que disponemos en una tabla de la base de datos es listarlos para, a partir de entonces, saber qué registro queremos conocer en detalle, modificar o borrar. Este tipo de tarea suelen llevarla a cabo páginas PHP que disponemos como índice del proyecto: en este caso, como en los venideros, cada fichero dispondrá de un nombre acorde con la actividad que desarrolla.

Para crear la página que realizará los listados necesitamos realizar una selección de todos los registros que contengan las columnas *nick* e *id* de la tabla *directorio*. Estos datos los mostraremos dentro de una tabla generada con *tags* HTML y, además, estableceremos hipervínculos a las páginas que nos servirán para ver en detalle, editar y/o eliminar la información de cada una de las filas de la tabla, utilizando como parámetro el valor almacenado en el campo *id* del registro.

**Tabla 4.1.**  
*Estructura de la página de listado*

|  |  |  |
| --- | --- | --- |
| ***ID de registro*** | ***Nombre*** | ***Acción*** |
| Id de registro1 | Nick de persona1 | ver editar eliminar |
| … | … | … … … |
| Id de registroN | Nick de personaN | ver editar eliminar |

Para generar tanto las páginas que realicen el listado como las siguientes combinaremos texto HTML con PHP (incluyendo las consultas SQL de éste a la base de datos y la impresión de resultados); especificaremos "dentro de PHP" cuando el código que tecleemos se encuentre entre las etiquetas PHP (<?php y ?>) y "fuera de PHP" cuando hagamos referencia a texto normal, por lo general, escrito en lenguaje HTML.

Crearemos un nuevo documento de texto e incluiremos los archivos de configuración para comenzar:

*<?php  
include ("includes/config.php");  
include ("inlcudes/funciones.php");  
?>*

A continuación, fuera de PHP, comenzaremos a dibujar la tabla que almacenará la información, junto con lo que serán las cabeceras de la misma. Como puede notar observando la tabla 4.1, se requieren cinco columnas: *id, nombre, ver, editar* y *eliminar*, estando las tres últimas agrupadas bajo la misma cabecera de la tabla (la que lleva por título *acción*). El código HTML que genera esta estructura sería el siguiente:

|  |
| --- |
| *<table class="table table-bordered table-striped"> <tr>  <td>id</td>  <td>Nombre</td>  <td colspan="3" align="center">Acci&oacute;n</td> </tr>* |

Las líneas anteriores generan la declaración de la tabla (etiqueta *<table …>*) junto con su primera fila (*<tr> … </tr>*), en la que establecemos los textos de la cabecera. Cuando hayamos de imprimir los distintos registros tras la consulta mostraremos uno por fila, pero antes necesitaremos conectarnos a la base de datos mediante PHP utilizando la función que hemos declarado en el fichero *funciones.php* (*conectar*), guardando el identificador de conexión en una variable de nombre *$cnx*, para posteriormente realizar la consulta SQL a MySQL:

*<?php  
//nos conectamos a mysql  
$cnx = conectar( );*

Como comentamos anteriormente, sólo necesitaremos las columnas *nick* e *id* a la hora de realizar la consulta, obteniendo de ambas todos los registros en la tabla directorio. La consulta SQL se podría formular de la siguiente forma:

*$sql = "SELECT id, nick FROM directorio ORDER BY id ASC";*

**Nota:** Recuerde que no es lo mismo seleccionar todas las columnas de una table ("SELECT \* FROM …") que seleccionar todos los registros, ya que cuando sólo necesitamos algunos se utilizan los denominados límites ("… LIMIT 0, 10"), que ya vimos en la unidad anterior.

Observe cómo en la consulta SQL se ordenan de forma ascendente los resultados obtenidos de seleccionar las columnas *id* y *nick*. A continuación realizamos la consulta a MySQL almacenando el identificador de resultado en la variable *$res*, especificando que se impriman los errores en caso de que se produzca alguno:

*$res = mysql\_query($sql) or die(mysql\_error( ));*

Realizada la consulta, revisamos si existen resultados de la misma, para lo cual comprobamos si el número de filas almacenado en la variable *$res* es mayor que cero:

*if (mysql\_num\_rows($res) > 0) {*

Y si efectivamente comprobamos que existe más de una fila (lo que quiere decir que hay resultados devueltos por la consulta) procedemos a la impresión de los datos en la pantalla, utilizando para ello un bucle de tipo *while*; puesto que sólo tenemos dos columnas seleccionadas, la forma más sencilla de procesar los datos es asignar cada columna a una variable por medio de la función *list*:

*while (list($id, $nick) = mysql\_fetch\_array($res)) {*

A partir de ahora, el resultado de la fila en que se encuentre el bucle *while* que usamos para recorrer las columnas *id* y *nick* se encontrará en las variables *$id* y *$nick* que hemos asignado. Las dos primeras celdas de la tabla HTML son las que contienen los valores de las variables, así que las imprimimos dándole el formato de celda de tabla HTML (*<tr> <td> … </td> … <td> …</td> </tr>*, donde el *tag <tr>* sirve para especificar el comienzo de una fila y los *tags <td>* sirven para especificar el comienzo de una de las celdas de la fila):

*echo "<tr><td>$id</td>\n";  
echo "<td>$nick</td>\n";*

**Nota:** El carácter de salto de línea (\n) que se agrega al final de cada llamada a la función *echo* se utiliza para que el código HTML resulte legible una vez impresa la página.

Las tres siguiente celdas que se añaden a la fila han de contener hipervínculos a los documentos *ver.php, editar.php* y *eliminar.php*, por lo que para indicarles a éstas el registro que se quiere ver o editar utilizaremos el método GET, para pasar el valor de la columna *id* en la llamada al documento PHP. Por ejemplo, si la variable *id* contiene el valor 5, en el hipervínculo que apunta hacia el documento *ver.php* haremos referencia al valor *id* de la siguiente forma:

*ver.php?id=5*

lo que en HTML se genera así:

*<a href="ver.php?id=5">Ver</a>*

y si utilizamos PHP, como es nuestro caso, habremos de generarlo de este mfodo:

*echo "<a href='ver.php?id=$id'>Ver</a>";*

Si deseamos imprimir comillas dobles (" y ") dentro del código HTML, podemos conseguirlo anteponiendo el carácter "\" cada vez que aparecen las comillas en el texto:

*echo "<a href=\"ver.php?id=$id\">Ver</a>";*

O bien de esta otra forma:

*echo '<a href="ver.php?id=$id">Ver</a>';*

Sea cual sea la opción que elija, se imprimirá el mismo resultado. Recordemos también que hemos de imprimir los *tags* (<td> … </td>) que especifiquen que el texto con el vínculo se encuentran en una celda de la fila. Al final, la impresión de las tres celdas conteniendo los textos con hipervínculos se realizará así:

*echo "<td><a href='ver.php?id=$id'>Ver</a></td>\n";  
echo "<td><a href='editar.php?id=$id'>Editar</a></td>\n";  
echo "<td><a href='eliminar.php?id=$id'>Eliminar</a></td></tr>\n";*

Observe que en la última línea se agrega la etiqueta que cierra la fila HTML (</tr>). En caso de que no se hayan obtenido resultados, es conveniente imprimir un mensaje que informe de dicha incidencia:

*} else {  
 echo "<td colspan = '5' align = 'center'>No se obtuvieron resultados</td>";  
}*

**Nota:** La razón por la que se añade *colspan = '5'* en la declaración de la celda es la de que estamos imprimiendo texto dentro de una tabla HTML que dispone de cinco celdas por fila.

Cerramos la conexión a MySQL y salimos de PHP:

*mysql\_close ($cnx);  
?>*

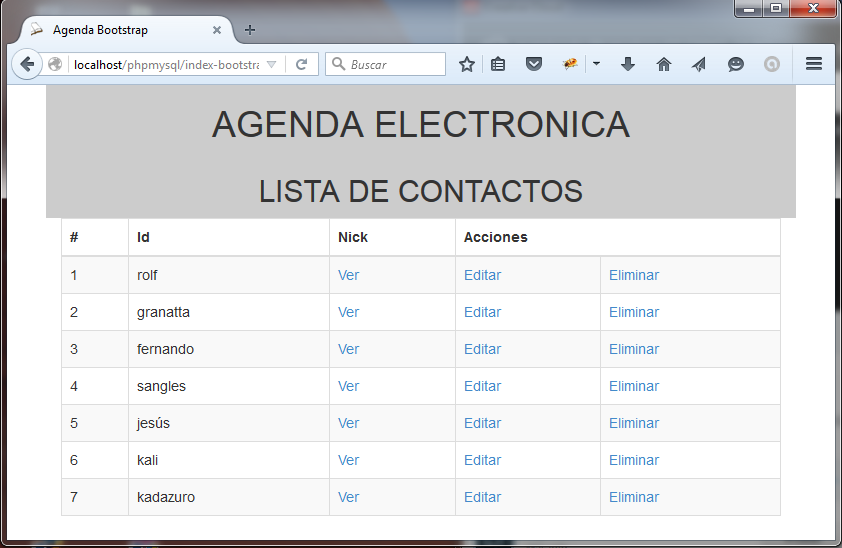
Y por último, cerramos la tabla HTML:

*</table>*

El código completo del documento (incluido el uso del framework CSS Bootstrap) es el siguiente:

|  |
| --- |
| <?php  include\_once("includes/config.php");  include\_once("includes/funciones.php");  //nos conectamos a mysql  $cnx = conectar ();  //consulta.  $sql = "SELECT id,nick FROM directorio ORDER BY id ASC";  $res= mysql\_query($sql) or die (mysql\_error());  ?>  <!DOCTYPE html>  <html lang="es">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Agenda Bootstrap</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <link rel="stylesheet" href="css/styles.css">  <link rel="shortcut icon" href="favicon.ico" />  </head>  <body>  <div class="container">  <div class="row text-center color1">  <h1>AGENDA ELECTRONICA</h1>  </div>  <div class="row text-center color1">  <h2>LISTA DE CONTACTOS</h2>  </div>  <div class="table-responsive">  <table class="table table-bordered table-striped">  <thead>  <tr>  <th>#</th>  <th>Id</th>  <th>Nick</th>  <th colspan="3">Acciones</th>  </tr>  </thead>  <tbody>  <!-- Aqui voy a generar la tabla con una consulta -->  <?php  if( mysql\_num\_rows($res) > 0){  //impresión de los datos.  while (list($id,$nick) = mysql\_fetch\_array($res)) {  echo "<tr><td>$id</td>\n";  echo "<td>".utf8\_encode($nick)."</td>\n";  echo "<td><a href='ver.php?id=$id'>Ver</a></td>\n";  echo "<td><a href='editar.php?id=$id'>Editar</a></td>\n";  echo "<td><a href='eliminar.php?id=$id'>Eliminar</a></td></tr>\n";  }  }else{  echo "<td colspan='5' align='center'>No se obtuvieron resultados</td>";  }  mysql\_close($cnx);  ?>  </tbody>  </table>  </div>  </div>  </body>  </html> |

Guarde este documento con el nombre *index.php* en la carpeta *phpmysql* de su servidor local y ábralo en su navegador tecleando [http://localhost/phpmysql/index.php](http://localhost/phpmysql/index-bootstrap.php). Los resultados que debe ver en su pantalla han de ser como los de la figura 4.9.



**Figura 4.9.**  
*Fichero* index.php *mostrando resultados en pantalla.*

### 4.4.4. Obtención de detalles de un registro

El documento que nos mostrará los detalles de un registro es una selección de todas las columnas de la tabla *directorio*. La variable *id*, enviada mediante GET, nos ayudará a saber cuál es el registro que hay que mostrar y, una vez realizada la consulta a la base de datos, mostraremos la información en una tabla HTML con una estructura como la mostrada en la tabla 4.2.

**Tabla 4.2.**  
*Estructura de la página de detalle*

|  |  |
| --- | --- |
| ***ID de registro*** | ***Id*** |
| Nombre | nombre apellido |
| Nick | nick |
| Email | email |
| Web site | url |

Este tipo de documentos podemos dividirlos en dos secciones:

* Si la variable *id* enviada mediante GET no existe, no se puede continuar.
* Si existen resultados, los muestra. En caso contrario, se muestra un mensaje de ello.

Esta última sección ya sabemos cómo implementarla; la primera la desarrollaremos a continuación. Como vimos en uno de los apartados anteriores, la función *isset* se utiliza para saber si una variable existe (está definida) o no, y su sintaxis es:

*isset ($miVariable)*

Si *$miVariable* está definida, la función *isset* retorna el valor *true* (verdadero), pero lo que deseamos consultar es si no está definida, para lo que utilizamos el operador ! (NOT):

*!isset ($miVariable)*

En caso de que la variable no exista, lo que haremos será retornar a la página de listado:

*if (!isset($miVariable)) {  
 //redireccionar  
}*

Pero, ¿qué ocurriría si la variable está definida pero no tiene un valor asignado? La página no accederá al redireccionamiento y seguirá ejecutándose. Para evitar este hecho utilizaremos la función *empty*, que sirve para comprobar si la variable está definida y además contiene algún valor (no está vacía). Su sintaxis es la siguiente:

*empty ($miVariable)*

Por tanto, el uso que daremos a la función será:

*if (empty($miVariable)) {  
 //redireccionar  
}*

*Para redireccionar y retornar a la página de listado, usaremos las siguientes dos líneas de código:*

*header("Location: index.php");  
exit;*

El documento *ver.php* comenzará verificando la existencia y valor de la variable *id* y comprobará que ésta haya sido enviada mediante el método GET. Si todo es correcto, se incluyen nuestros archivos de configuración y funciones, y en caso contrario se redireccionará al usuario a la página de listado (todas estas acciones se declaran dentro de PHP).

*<?php  
//si no hay id, no puede seguir.  
if (empty($\_GET['id'])) {  
 header ("Location: index.php");  
 exit;  
}  
include ("includes/config.php");  
include ("includes/funciones.php");  
?>*

A continuación, construimos en HTML la tabla que contendrá la información:

*<table class="table table-bordered table-striped">*

Establecemos una conexión con la base de datos, entrando de nuevo en PHP:

*<?php  
//nos conectamos a mysql  
$cnx=conectar( );*

Recuerde, a la hora de realizar la consulta SQL, que deseamos seleccionar todas las columnas donde el identificador único tenga el mismo valor que el de la variable *id* enviada mediante GET:

*$sql = "SELECT \* FROM directorio WHERE id =" . $\_GET['id'];*

Posteriormente, efectuamos la consulta SQL almacenando el identificador de resultado en una variable de nombre *$res* y especificando que imprima los errores si se producen:

*$res = mysql\_query ($sql) or die (mysql\_error( ));*

Comprobamos el número de filas de la variable *$res* para saber si hay resultados o no:

*if ( mysql\_num\_rows($res) > 0) {*

Si encontramos resultados, comenzamos a procesarlos asignando el resultado que retorna la función *mysql\_fetch\_array* (un *array* de lista con los nombres de las columnas) a una variable de nombre *$fila:*

*while ($fila = mysql\_fetch\_array ($res)) {*

En esta ocasión no saldremos de PHP, y la siguiente sección la generaremos mediante HTML, insertando pequeños fragmentos de código PHP cuando deseemos imprimir los datos de la columna. Por ejemplo, si vamos a imprimir la columna *id*, utilizaríamos:

*<tr>  
<td>Id</td>  
<td><?php echo $fila['id']; ?></td>  
</tr>*

En PHP se realizaría de la misma forma que vimos con anterioridad:

*echo "<tr><td>Id</td><td>".$fila['id']."</td></tr>";*

Insertar pequeños fragmentos de código PHP dentro de un documento escrito en HTML nos ayuda visualmente, ya que estemos utilizando un editor de texto o un editor de tipo WYSIWYG para generar los documentos .html. Cuando anteriormente creamos el documento *listado.php*, el código HTML fue impreso por medio de PHP. En esta ocasión el código PHP será embebido o incrustado en el código HTML, de modo que dispone de varias opciones para trabajar, todas ellas perfectamente válidas. Elija aquella que más cómoda le resulte.

La impresión de los datos consiste en una columna de la consulta por cada fila de la tabla HTML, donde una de las celdas se utiliza para imprimir el nombre y la otra para imprimir el valor de la columna (en el resultado de la consulta), teniendo en cuenta que tanto el nombre como el apellido se mostrarán en una sola celda:

*<td><?php echo $fila['nombre']." ".$fila['apellido']; ?></td>*

Ya que estamos siguiendo la estructura de tabla que especificamos en la tabla 4.2. El código para imprimir los datos será el siguiente:

|  |
| --- |
| <tr>  <td>Id</td>  <td><?php echo $fila['id']; ?></td>  </tr>  <tr>  <td>nombre</td>  <td><?php echo $fila['nombre']." ". $fila['apellido']; ?></td>  </tr>  <tr>  <td>nick</td>  <td><?php echo $fila['nick'];?></td>  </tr>  <tr>  <td>email</td>  <td><?php echo $fila['email'];?></td>  </tr>  <tr>  <td>web site</td>  <td><?php echo $fila['url'];?></td>  </tr> |

Cuando hayamos terminado de imprimir los datos, cerraremos el bucle *while* y especificaremos el mensaje alternativo que ha de mostrarse en caso de que no se hayan encontrado resultados, utilizando para ello código PHP:

|  |
| --- |
| <?php  }  } else {  echo "<tr><td colspan='2' align='center'>no se obtuvieron resultados</td></tr>";  } |

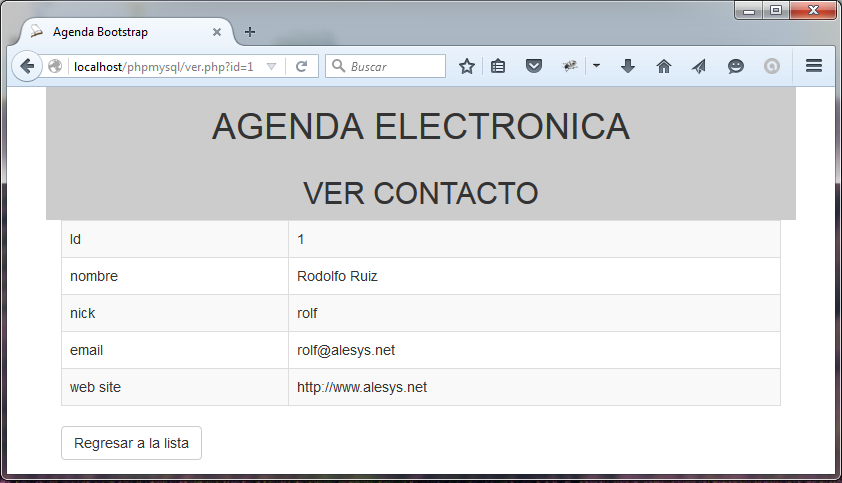
Para finalizar la construcción de la página, cerramos tanto la conexión con MySQL como la tabla HTML:

*mysql\_close ($cnx);  
?>  
</table>*

El código completo de la página *ver.php* es el siguiente:

|  |
| --- |
| <?php  //si no hay id, no puede seguir.  if(empty($\_GET['id'])){  header("Location: index.php");  exit;  }  include\_once ("includes/config.php");  include\_once ("includes/funciones.php");  ?>  <!DOCTYPE html>  <html lang="es">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Agenda Bootstrap</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <link rel="stylesheet" href="css/styles.css">  <link rel="shortcut icon" href="favicon.ico" />  </head>  <body>  <div class="container">  <div class="row text-center color1">  <h1>AGENDA ELECTRONICA</h1>  </div>  <div class="row text-center color1">  <h2>VER CONTACTO</h2>  </div>  <div class="table-responsive">  <table class="table table-bordered table-striped">  <?php  //nos conectamos a mysql  $cnx = conectar ();  //consulta.  $sql = "SELECT \* FROM directorio WHERE id=".$\_GET['id'];  $res= mysql\_query($sql) or die (mysql\_error());    if( mysql\_num\_rows($res) >0){  //impresión de los datos.  while ($fila = mysql\_fetch\_array($res)) {  ?>  <tr>  <td>Id</td>  <td><?php echo $fila['id'];?></td>  </tr>  <tr>  <td>nombre</td>  <td><?php echo utf8\_encode($fila['nombre']." ". $fila['apellido']);?></td>  </tr>  <tr>  <td>nick</td>  <td><?php echo utf8\_encode($fila['nick']);?></td>  </tr>  <tr>  <td>email</td>  <td><?php echo $fila['email'];?></td>  </tr>  <tr>  <td>web site</td>  <td><?php echo $fila['url'];?></td>  </tr>  <?php  }  }else{  echo "<tr><td colspan='2' align='center'>no se obtuvieron resultados</td></tr>";  }  mysql\_close($cnx);  ?>  </table>  </div>  <a class="btn btn-default" href="index.php" role="button">Regresar a la lista</a>  </div>  </body>  </html> |

Guarde este documento con el nombre *ver.php* en la carpeta *phpmysql* de su servidor local y abra de nuevo en su navegador el documento *index.php* tecleando <http://localhost/phpmysql/index.php>. Cuando presione cada uno de los vínculos de nombre *ver* (junto a cada uno de los nombres listados) en la tabla HTML, será enviado al documento *ver.php* que acabamos de realizar, mostrando los datos asociados al registro elegido, como puede ver en la figura 4.10, que corresponden a los datos obtenidos tras pulsar el enlace *ver* para obtener más información sobre *Rolf Ruiz.*

  
**Figura 4.10.**  
*Visualización de los detalles del registro cuyo* id *es* 1.

### 4.4.5. Modificación de un registro

Los pasos que realizaremos para modificar los contenidos de un registro de la tabla *directorio* son los siguientes:

* Enviaremos mediante GET el identificador del registro.
* Mostraremos la información actual que contiene el registro.
* Construiremos un formulario para actualizar la información.
* Una vez enviado el formulario, actualizaremos la información.

El identificador de registro que pasamos mediante GET a las páginas *ver.php*, *editar.php* y *eliminar.php* es el identificador que utilizamos para referenciar el registro del que deseamos obtener información tras realizar la consulta SQL.

Para poder editar la información crearemos un formulario cuyos campos contengan los datos del registro. Estos campos serán editables a excepción del campo único, cuyo valor no ha de ser modificado pero sí mostrado para que sepamos sobre qué registro estamos aplicando los cambios, por lo que utilizaremos un campo oculto en lugar de un campo de texto.

El formulario, al ser enviado, actualizará los datos. Puesto que vamos a realizar todas las acciones en un solo documento, hemos de determinar cuándo se efectuarán cada una de ellas:

* Si no existe la variable enviada mediante GET, redireccionaremos al usuario al documento *index.php*.
* Si existe dicha variable, realizamos la consulta.
* Si el formulario es enviado, dispondremos de una variable de nombre *submit* que nos servirá para saber que podemos actualizar la información.

El orden en que el código estará dispuesto en el documento PHP será el siguiente:

* Actualización de los datos.
* Verificación del valor de *id*.
* Despliegue de información en el formulario.

**Nota:** El orden en que el código está dispuesto en el documento PHP no es el mismo orden en el que el código se ejecutará, ya que la actualización de datos no se llevará a cabo hasta que el formulario haya sido enviado, y éste no puede contener datos hasta que se haya verificado la existencia y valor de *id*.

Abra un nuevo documento de texto e incluya primeramente los archivos de configuración que hemos venido utilizando en los distintos ejemplos:

*<?php  
include ("includes/config.php");  
include ("includes/funciones.php");*

Ahora crearemos la sección de actualización de datos; nos centraremos por el momento en la consulta de los datos y en mostrar estos últimos en el formulario.

**Nota:** Recordemos que falta por realizar la sección en la que se van a actualizar los datos y que debe colocarse justamente después de la inclusión de los archivos de configuración.

Comprobamos si existe la variable *id* enviada mediante GET. De no ser así, redireccionamos al usuario al documento *index.php:*

*//si no hay id, no puede seguir.  
if (empty($\_GET['id'])) {  
 header ("Location: index.php");  
 exit;  
}*

Posteriormente, establecemos la conexión con MySQL:

*$cnx = conectar( );*

Especificamos la consulta SQL teniendo en cuenta que para editar cualquier campo del registro hemos de seleccionar todas las columnas de la tabla donde el valor del identificador único coincida con el de la variable *id* enviada mediante GET:

*$sql = "SELECT \* FROM directorio WHERE id = ". $\_GET['id'];*

Y realizamos la consulta configurando que se muestren los errores en caso de que se produzca alguno:

*$res = mysql\_query($sql) or die(mysql\_error( ));*

Si existen resultados (el número de filas almacenado en la variable *$res* es mayor que cero), extraeremos los datos para procesarlos:

*if ( mysql\_num\_rows($res) > 0) {  
//si hay resultados hacemos la forma.  
?>*

En este momento hemos cerrado la etiqueta de código PHP, con lo que el código que dispondremos a continuación es HTML, mediante el que imprimiremos la etiqueta del formulario y la cabecera de la tabla que contendrá los campos de texto con la información de la base de datos:

|  |
| --- |
| <form class="form-horizontal form-margin" role="form" method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>"> |

Como dijimos anteriormente, esta página se encargará de procesar la información enviada por el formulario, por lo que establecemos como acción para el mismo la variable *$\_SERVER['PHP\_SELF´]*, que se encargará de referenciar la ruta de este archivo para que el formulario sea enviado a la misma página en que se declara.

Abrimos de nuevo las etiquetas PHP tan sólo para crear un bucle de tipo *while*, puesto que las celdas de la tabla y los campos de texto los realizaremos mediante HTML, embebiendo el código PHP con los datos de las columnas cuando sea necesario:

|  |
| --- |
| <?php  //impresión de los datos.  while ($fila = mysql\_fetch\_array($res)) {  ?> |

Utilizaremos estilos Bootstrap para mostrar la información del contacto. El “id” del registro que estamos editando lo colocaremos en un input de tipo “hidden” para poderlo enviarlo como parámetro para la consulta de actualización.

**Tabla 4.3.**  
*Estructura de la página editar.*

|  |  |
| --- | --- |
| ***ID de registro*** | ***Id*** |
| Nombre | [nombre] |
| Apellido | [apellido] |
| Nick | [nick] |
| Email | [email] |
| Web site | [url] |

Para escribir los datos en la forma indicada, debemos incluir la etiqueta del campo de texto de la siguiente forma:

|  |
| --- |
| <input name="{nombre}" type="text" id="{nombre}"> |

Y, posteriormente, embeber el código PHP cuando queramos imprimir el valor de la columna:

|  |
| --- |
| <input class="form-control" name="{nombre}" type="text" id="{nombre}" value="<?php echo $fila['{nombre}'];?>"> |

Finalmente, el formato de cada una de las columnas se mostrará como sigue:

|  |
| --- |
| <div class="form-group">  <label for="{nombre}" class="col-sm-2 control-label">{nombre:}</label>  <div class="col-sm-10">  <input class="form-control" name="{nombre}" type="text" id="{nombre}" value="<?php echo $fila['{nombre}'];?>">  </div>  </div> |

Además necesitamos un campo de texto oculto con el valor del campo *id*, pudiendo añadir una impresión en texto del valor si se desea:

|  |
| --- |
| <input class="form-control" name="id" type="hidden" id="id" value="<?php echo $fila['id'];?>"> |

El resto de las columnas se muestran por pantalla de forma uniforme:

|  |
| --- |
| <div class="form-group">  <label for="nombre" class="col-sm-2 control-label">Nombre:</label>  <div class="col-sm-10">  <input class="form-control" name="nombre" type="text" id="nombre" value="<?php echo $fila['nombre'];?>">  </div>  </div>  <div class="form-group">  <label for="apellido" class="col-sm-2 control-label">Apellido:</label>  <div class="col-sm-10">  <input class="form-control" name="apellido" type="text" id="apellido" value="<?php echo $fila['apellido'];?>">  </div>  </div>  <div class="form-group">  <label for="nick" class="col-sm-2 control-label">Nick:</label>  <div class="col-sm-10">  <input class="form-control" name="nick" type="text" id="nick" value="<?php echo $fila['nick'];?>">  </div>  </div>  <div class="form-group">  <label for="email" class="col-sm-2 control-label">Email:</label>  <div class="col-sm-10">  <input class="form-control" name="email" type="text" id="email" value="<?php echo $fila['email'];?>">  </div>  </div>  <div class="form-group">  <label for="url" class="col-sm-2 control-label">Web site:</label>  <div class="col-sm-10">  <input class="form-control" name="url" type="text" id="url" value="<?php echo $fila['url'];?>">  </div>  </div> |

Para añadir el botón SUBMIT al formulario declararíamos la celda de la siguiente forma:

|  |
| --- |
| <div class="form-group">  <div class="col-sm-10 col-sm-offset-2">  <input type="submit" name="submit" id="submit" value="Enviar" class="btn btn-primary">  </div>  </div> |

Es importante que especifiquemos *sumbit* como nombre para el botón, ya que es el dato que vamos a verificar cuando estemos realizando la actualización de los datos. Posteriormente, cerramos el formulario HTML:

</form>

De nuevo dentro de PHP, cerramos el bucle de tipo *while* y especificamos un mensaje alternativo en caso de que no haya resultados tras la consulta. En esta ocasión, el mensaje no necesita mostrarse conjuntamente con código HTML, ya que la tabla sólo se imprime a la hora de construirse el formulario:

|  |
| --- |
| <?php  } }else{  echo "No se obtuvieron resultados"; } |

Cerramos la conexión a MySQL y la etiqueta PHP:

*mysql\_close($cnx);  
?>*

Ahora vamos a construir la parte que tenemos pendiente, la actualización de los datos enviados desde el formulario. Cuando éste es enviado sabemos que entre las variables que contiene existe una de nombre *submit* (la que envía el botón ***SUBMIT***) y otra de nombre *id* (el campo oculto). Para comenzar, comprobamos la existencia de *submit*:

*if (isset($\_POST['submit'])){*

Si es así, establecemos una conexión con MySQL:

*$cnx=conectar( );*

El formato de actualización de registros en la base de datos es, como vimos en la unidad anterior:

|  |
| --- |
| UPDATE {tabla} SET campo1='valor1', … campoN='valorN' WHERE campo = valor; |

En nuestro caso, agregaremos los campos de uno en uno y en líneas separadas para que pueda "visualizar" el proceso de recolección de datos. Declaramos una variable de nombre *$sql* para realizar la consulta:

$sql = "UPDATE directorio SET ";

Y concatenamos el contenido de la variable con el nombre de cada columna junto con la variable asociada a ésta:

|  |
| --- |
| $sql .= "nombre = ' ". $\_POST['nombre']. " ', ";  $sql .= "apellido = ' ". $\_POST['apellido']. " ', ";  $sql .= "nick = ' ". $\_POST['nick']. " ', ";  $sql .= "email = ' ". $\_POST['email']. " ', ";  $sql .= "url = ' ". $\_POST['url']. " ' "; |

Posteriormente, añadimos una cláusula *where* para indicar qué registro ha de ser modificado (aquel en el que el valor del identificador único coincida con el de la variable *id*):

$sql .= " WHERE id = ". $\_POST['id'];

Para finalizar, redireccionamos a la página mensaje1.php que despliega un mensaje confirmando que la actualización de datos se ha efectuado con éxito y un vínculo al documento *index.php*.

|  |
| --- |
| header("Location: mensaje1.php"); |

Cerramos la conexión establecida con MySQL:

*mysql\_close ($cnx);*

Y hacemos uso de la función *exit* para especificarle a PHP que deje de interpretar el documento y así no aparezca de nuevo el formulario:

*exit;  
}*

El código de la página *mensaje1.php* es el siguiente:

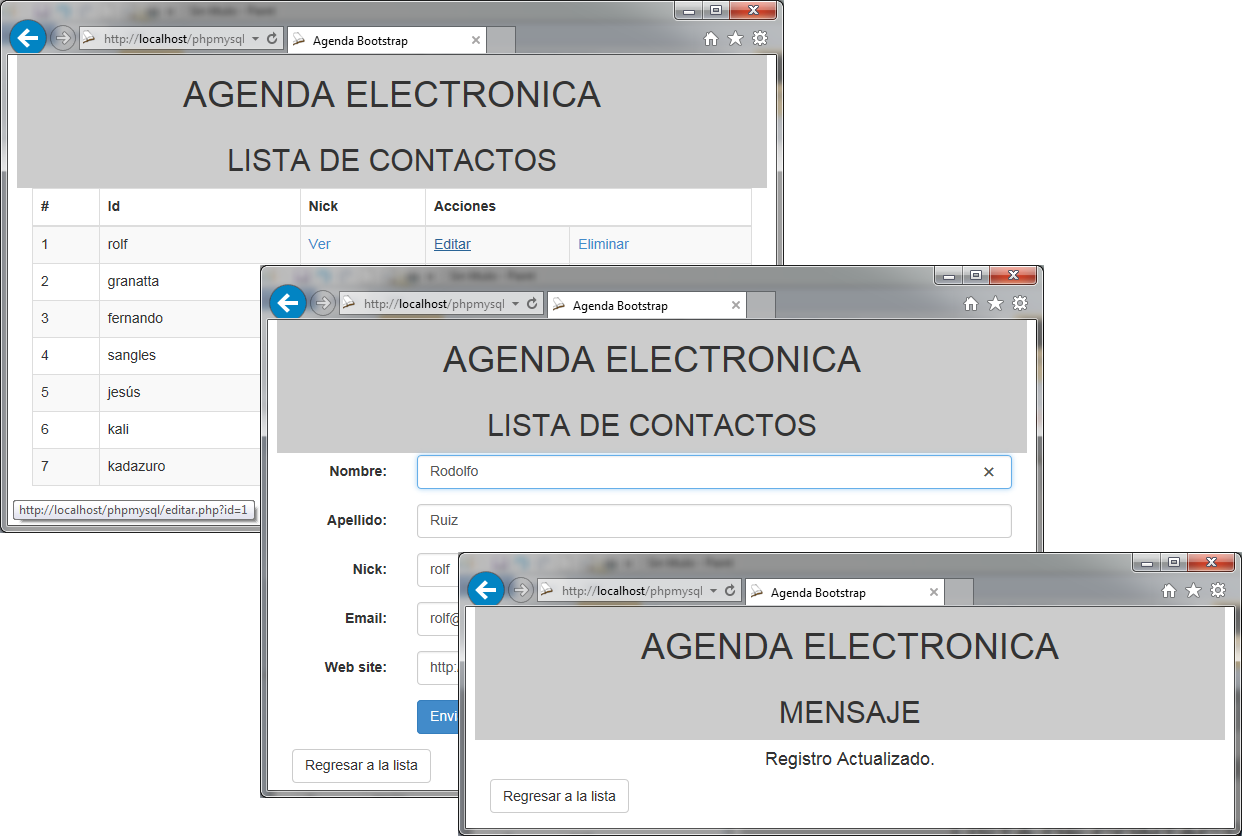
|  |
| --- |
| <!DOCTYPE html>  <html lang="es">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Agenda Bootstrap</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <link rel="stylesheet" href="css/styles.css">  <link rel="shortcut icon" href="favicon.ico" />  </head>  <body>  <div class="container">  <div class="row text-center color1">  <h1>AGENDA ELECTRONICA</h1>  </div>  <div class="row text-center color1">  <h2>MENSAJE</h2>  </div>  <div class="row text-center">  <h4>Registro Actualizado.</h4>  </div>  <a class="btn btn-default" href="index.php" role="button">Regresar a la lista</a>  </div>  </body>  </html> |

El código completo de la página es el siguiente:

|  |
| --- |
| <?php  include ("includes/config.php");  include ("includes/funciones.php");  //si la forma ha sido enviada editamos el registro.  if(isset($\_POST['submit'])){  //nos conectamos a mysql  $cnx = conectar ();  $sql = "UPDATE directorio SET ";  $sql .= "nombre ='".$\_POST['nombre']."',";  $sql .= "apellido ='".$\_POST['apellido']."',";  $sql .= "nick ='".$\_POST['nick']."',";  $sql .= "email ='".$\_POST['email']."',";  $sql .= "url ='".$\_POST['url']."'";  $sql .= " WHERE id =".$\_POST['id'];  $res = mysql\_query($sql) or die(mysql\_error());  //echo "Registro actualizado.<br><a href='index.php'>regresar</a>";  mysql\_close($cnx);  header("Location: mensaje1.php");  exit;  }  //si no hay id, no puede seguir.  if(empty($\_GET['id'])){  header("Location: index.php");  exit;  }  //nos conectamos a mysql  $cnx = conectar ();  //consulta para mostrar los datos.  $sql = "SELECT \* FROM directorio WHERE id=".$\_GET['id'];  $res= mysql\_query($sql) or die (mysql\_error());  ?>  <!DOCTYPE html>  <html lang="es">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Agenda Bootstrap</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <link rel="stylesheet" href="css/styles.css">  <link rel="shortcut icon" href="favicon.ico" />  <style type="text/css">  .form-margin {  margin-top: 2px;  }  </style>  </head>  <body>  <div class="container">  <div class="row text-center color1">  <h1>AGENDA ELECTRONICA</h1>  </div>  <div class="row text-center color1">  <h2>LISTA DE CONTACTOS</h2>  </div>  <?php  if( mysql\_num\_rows($res) >0){  //si hay resultados hacemos la forma.  ?>  <form class="form-horizontal form-margin" role="form" method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">  <?php  //impresión de los datos.  while ($fila = mysql\_fetch\_array($res)) {  ?>  <input class="form-control" name="id" type="hidden" id="id" value="<?php echo $fila['id'];?>">  <div class="form-group">  <label for="nombre" class="col-sm-2 control-label">Nombre:</label>  <div class="col-sm-10">  <input class="form-control" name="nombre" type="text" id="nombre" value="<?php echo $fila['nombre'];?>">  </div>  </div>  <div class="form-group">  <label for="apellido" class="col-sm-2 control-label">Apellido:</label>  <div class="col-sm-10">  <input class="form-control" name="apellido" type="text" id="apellido" value="<?php echo $fila['apellido'];?>">  </div>  </div>  <div class="form-group">  <label for="nick" class="col-sm-2 control-label">Nick:</label>  <div class="col-sm-10">  <input class="form-control" name="nick" type="text" id="nick" value="<?php echo $fila['nick'];?>">  </div>  </div>  <div class="form-group">  <label for="email" class="col-sm-2 control-label">Email:</label>  <div class="col-sm-10">  <input class="form-control" name="email" type="text" id="email" value="<?php echo $fila['email'];?>">  </div>  </div>  <div class="form-group">  <label for="url" class="col-sm-2 control-label">Web site:</label>  <div class="col-sm-10">  <input class="form-control" name="url" type="text" id="url" value="<?php echo $fila['url'];?>">  </div>  </div>  <div class="form-group">  <div class="col-sm-10 col-sm-offset-2">  <input type="submit" name="submit" id="submit" value="Enviar" class="btn btn-primary">  </div>  </div>  </form>  <?php  }  }else{  //no hay resultados, id malo o no existe.  echo "No se obtuvieron resultados";  }  mysql\_close($cnx);  ?>  <a class="btn btn-default" href="index.php" role="button">Regresar a la lista</a>  </div>  </body>  </html> |

Guarde el documento con el nombre *editar.php* en la carpeta *phpmysql* de su servidor local y abra de nuevo en su navegador el documento *index.php* tecleando [*http://localhost/phpmysql/index.php*](http://localhost/phpmysql/index.php)*.*

Seleccione el registro que desee editar y realice los cambios que desee, en un proceso que debe ser como el que se muestra en la figura 4.11.

  
**Figura 4.11.**  
*Documento* editar.php *como formulario y resultado de edición.*

4.4.6. Inserción de un nuevo registro

El proceso de insertar un nuevo registro es más simple que el de editarlo, ya que, aunque también necesitamos un formulario para introducir los datos, no hemos de realizar una consulta previa para llenar sus campos con los valores obtenidos de la misma. Por otra parte, tampoco necesitaremos utilizar la variable *id*, ya que al ser un registro nuevo la base de datos se encargará de asignarle el número siguiente a los ya creados (al ser ese campo de la tabla de tipo autoincremental), siendo el número asignado el que lo identificará como único en la tabla. La página PHP para realizar las inserciones sólo consta de dos partes:

* Formulario.
* Ingreso de información.

Tal y como hicimos al crear la página para editar registros, nos basaremos en la existencia de una variable llamada *submit* para saber que estamos realizando un envío de nuevos datos. Primero realizaremos el código HTML para el formulario, y posteriormente el código PHP para realizar la inserción de información. Además, puesto que no necesitamos hacer ninguna consulta para mostrar los datos de la tabla *directorio*, no será preciso incluir los archivos de configuración de que disponemos en nuestro directorio *phpmysql/includes* del servidor local hasta que sea enviado el formulario.

Abra un nuevo documento de texto para crear la página que realizará las inserciones. La parte HTML de la misma es un formulario que contiene una tabla con los campos de texto, similar al usado en el documento *editar.php*, pero con la diferencia de que en este caso no necesitamos un campo oculto para el valor de *id:*

|  |
| --- |
| <form class="form-horizontal form-margin" role="form" method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">  <div class="form-group">  <label for="nombre" class="col-sm-2 control-label">Nombre:</label>  <div class="col-sm-10">  <input class="form-control" name="nombre" type="text" id="nombre" placeholder="Nombre" value="">  </div>  </div>  <div class="form-group">  <label for="apellido" class="col-sm-2 control-label">Apellido:</label>  <div class="col-sm-10">  <input class="form-control" name="apellido" type="text" id="apellido" placeholder="Apellido" value="">  </div>  </div>  <div class="form-group">  <label for="nick" class="col-sm-2 control-label">Nick:</label>  <div class="col-sm-10">  <input class="form-control" name="nick" type="text" id="nick" placeholder="Nick" value="">  </div>  </div>  <div class="form-group">  <label for="email" class="col-sm-2 control-label">Email:</label>  <div class="col-sm-10">  <input class="form-control" name="email" type="text" id="email" placeholder="Email" value="">  </div>  </div>  <div class="form-group">  <label for="url" class="col-sm-2 control-label">Web site:</label>  <div class="col-sm-10">  <input class="form-control" name="url" type="text" id="url" placeholder="Web site" value="">  </div>  </div>  <div class="form-group">  <div class="col-sm-10 col-sm-offset-2">  <input type="submit" name="submit" id="submit" value="Enviar" class="btn btn-primary">  </div>  </div>  </form> |

Puede observar que hemos establecido como acción para el formulario la variable $\_SERVER['PHP\_SELF'], con lo que el formulario será enviado a esta misma página en que se declara. El botón SUMBIT también se declara exactamente igual que en el documento *editar.php*.

Cuando creamos la parte PHP, comprobamos primeramente la existencia o no de la variable *submit* enviada mediante POST:

<?php  
if (isset($\_POST['submit'])) {

Si existe, incluimos los ficheros de configuración:

include ("includes/config.php");  
include ("includes/funciones.php");

Establecemos la conexión con la base de datos:

$cnx = conectar ( );

Ahora realizaremos la consulta SQL para la inserción. El formato, como se explicó en la unidad 3, es el siguiente:

INSERT INTO {tabla}  
(campo1, …campoN)  
VALUES (valor1, …valorN);

Como hicimos al utilizar UPDATE para la actualización de datos, realizaremos la consulta paso a paso con el fin de familiarizarnos con la concatenación de cadenas en variables. Separaremos la consulta en tres partes, una con el nombre de las columnas que serán afectadas en la consulta (todas las de la tabla excepto la de nombre *id*), otra con la concatenación de las variables enviadas mediante POST, y la última, que incluirá las dos anteriores junto con la estructura para realizar la inserción.

Creamos una variable llamada *$campos* que almacena los nombres de las columnas:

$campos = "nombre,apellido,nick,email,url";

**Nota:** Observe cómo los nombres se separan entre sí utilizando una coma.

Los valores de los campos han de colocarse entre comillas simples; creamos una nueva variable llamada *$valores* y concatenaremos en ella cada una de las variables a enviar (las disponemos de una en una para que el código resulte más legible:

|  |
| --- |
| $valores = "'".$\_POST['nombre']."',"; $valores = "'".$\_POST['apellido']."',"; $valores = "'".$\_POST['nick']."',"; $valores = "'".$\_POST['email']."',"; $valores = "'".$\_POST['url']."'"; |

El formato que hemos utilizado, si se fija, es el siguiente:

|  |
| --- |
| "comilla simple" + valor de variable + "comilla simple + coma" |

Con este formato conseguiremos que el valor de las variables quede declarado entre las comillas simples y separadas por una coma. El orden seguido es idéntico al especificado en la variable *$campos*, lo que es de suma importancia, ya que al ejecutar la consulta SQL se tomará el primer nombre de columna (en la variable *$campos*) junto con el primer valor a insertar en ésta (en la variable *$valores*).

Debe asegurarse además de no utilizar comas ni al final del último nombre de columna ni tras el último valor de las variables.

Formamos la consulta SQL con las dos variables que hemos creado (*$campos* y *$valores*):

*$sql = "INSERT INTO directorio ($campos) VALUES ($valores)";*

Después ejecutamos la consulta, especificando el error en caso de que la consulta no se realice correctamente:

*$res = mysql\_query ($sql) or die (mysql\_error( ));*

Mostramos en pantalla el mensaje de que el registro fue añadido con éxito y un enlace al documento *mensaje2.php*:

*header("Location: mensaje2.php");*

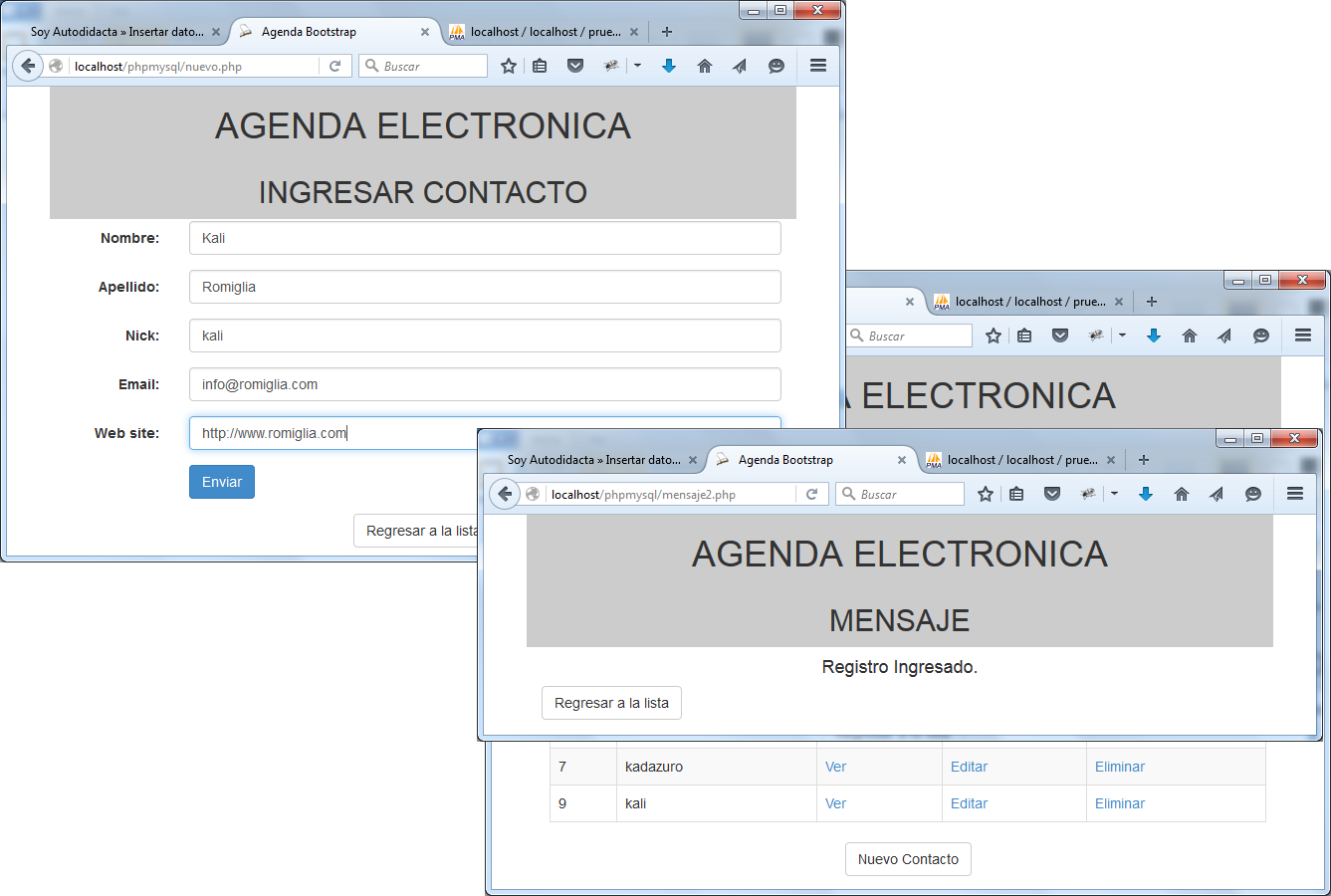
Cerramos la conexión establecida con MySQL y le ordenamos a PHP que deje de interpretar el documento utilizando la función *exit*:

*mysql\_close ($cnx);  
 exit;  
?>*

El código completo es el siguiente:

|  |
| --- |
| <?php  //si la forma ha sido enviada editamos el registro.  if(isset($\_POST['submit'])){    include ("includes/config.php");  include ("includes/funciones.php");  //nos conectamos a mysql  $cnx = conectar ();    $campos = "nombre,apellido,nick,email,url";  $valores = "'".$\_POST['nombre']."',";  $valores .= "'".$\_POST['apellido']."',";  $valores .= "'".$\_POST['nick']."',";  $valores .= "'".$\_POST['email']."',";  $valores .= "'".$\_POST['url']."'";  $sql = "INSERT INTO directorio ($campos) VALUES($valores)";  $res = mysql\_query($sql) or die(mysql\_error());  mysql\_close($cnx);  header("Location: mensaje2.php");  exit;  }  ?>  <!DOCTYPE html>  <html lang="es">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Agenda Bootstrap</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <link rel="stylesheet" href="css/styles.css">  <link rel="shortcut icon" href="favicon.ico" />  <style type="text/css">  .form-margin {  margin-top: 2px;  }  </style>  </head>  <body>  <div class="container">  <div class="row text-center color1">  <h1>AGENDA ELECTRONICA</h1>  </div>  <div class="row text-center color1">  <h2>INGRESAR CONTACTO</h2>  </div>    <form class="form-horizontal form-margin" role="form" method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">  <div class="form-group">  <label for="nombre" class="col-sm-2 control-label">Nombre:</label>  <div class="col-sm-10">  <input class="form-control" name="nombre" type="text" id="nombre" placeholder="Nombre" value="">  </div>  </div>  <div class="form-group">  <label for="apellido" class="col-sm-2 control-label">Apellido:</label>  <div class="col-sm-10">  <input class="form-control" name="apellido" type="text" id="apellido" placeholder="Apellido" value="">  </div>  </div>  <div class="form-group">  <label for="nick" class="col-sm-2 control-label">Nick:</label>  <div class="col-sm-10">  <input class="form-control" name="nick" type="text" id="nick" placeholder="Nick" value="">  </div>  </div>  <div class="form-group">  <label for="email" class="col-sm-2 control-label">Email:</label>  <div class="col-sm-10">  <input class="form-control" name="email" type="text" id="email" placeholder="Email" value="">  </div>  </div>  <div class="form-group">  <label for="url" class="col-sm-2 control-label">Web site:</label>  <div class="col-sm-10">  <input class="form-control" name="url" type="text" id="url" placeholder="Web site" value="">  </div>  </div>  <div class="form-group">  <div class="col-sm-10 col-sm-offset-2">  <input type="submit" name="submit" id="submit" value="Enviar" class="btn btn-primary">  </div>  </div>  </form>  <div class="text-center">  <a class="btn btn-default" href="index.php" role="button">Regresar a la lista</a>  </div>  </div>  </body>  </html> |

Guarde el documento con este código con el nombre *nuevo.php* en la carpeta *phpmysql* de su servidor local y ábralo en su navegador tecleando [*http://localhost/phpmysql/nuevo.php*](http://localhost/phpmysql/nuevo.php), ya que para añadir un nuevo registro no necesita pasar previamente por el documento de listado. Teclee los datos para el nuevo registro y añádalo a la base de datos. Al regresar al documento *index.php*, podrá observar cómo el registro que acaba de añadir se muestra junto con los que ya tenía almacenados, en un proceso que debe ser similar al de la figura 4.12.



### 

**Figura 4.12.**  
*Ingreso de registro, registro ingresado y listado al final.*

### 4.4.**7**. Borrado de un registro específico

Un gran inconveniente a la hora de eliminar registros en una página web, es tener **volver a cargar la página completa** para ver los resultados, algo que **podemos evitar usando Ajax.**

Para conseguirlo, Ajax recogerá el id de nuestro formulario de listado y lo enviará a otra página PHP que se encarga de realizar la eliminación. En este apartado veremos **cómo eliminar datos en MySQL con PHP y AJAX.**

De esta forma, la página web **no necesita recargarse** para mostrar los datos insertados, ni siquiera para eliminarlos simplemente.

Tendremos los siguientes archivos:

* **eliminar.php**: Elimina el registro de la base de datos.
* **index.php**: Despliega los registros insertados en la base de datos.
* **consulta.php**: Realiza una consulta de selección de los datos insertados en la base de datos.

Cabe recalcar que utilizaremos JQuery para controlar los eventos y las llamadas a AJAX. Se recomienda al lector investigar acerca de JQuery y Ajax.

Si bien eliminar registros es la tarea más sencilla, es también la más delicada, ya que un solo error puede conducir a la pérdida de todos los datos almacenados. En nuestro caso no vamos a borrar el registro al invocar la página, sino que crearemos una función en Javascript que invocará un alert con las opciones de “aceptar” la eliminación o “cancelar” la acción.

El borrado se realizará utilizando como parámetro el identificador (enviándolo mediante *POST* al fichero *eliminar.php*) del registro que queremos borrar. Este valor se recogerá en la variable *$\_POST["id"]* y se utilizará para borrar el registro especificado.

Abra un nuevo documento de texto para crear el fichero que realizará la eliminación con el siguiente contenido:

|  |
| --- |
| <?php  include\_once("../includes/config.php");  include\_once("../includes/funciones.php");    //nos conectamos a mysql  $cnx = conectar ();  $sql = "DELETE FROM directorio WHERE id =".$\_POST['id'];  $res = mysql\_query($sql) or die(mysql\_error());    echo "Registro eliminado exitosamente.";    mysql\_close($cnx);  ?> |

Incluimos los ficheros de configuración:

*include ("../includes/config.php");  
include ("../includes/funciones.php");*

Establecemos la conexión con MySQL utilizando la función *conectar* declarada en el fichero *funciones.php* de la carpeta *phpmysql/includes* de nuestro servidor local:

*$cnx = conectar( );*

La consulta SQL para realizar el borrado de un registro sigue el siguiente formato:

*DELETE FROM {tabla} WHERE campo = valor;*

En esta consulta, la cláusula *where* es la de mayor importancia, ya que es la que especifica qué registro eliminar. Éste es el lugar en el que adjuntamos el valor de la variable *id* enviada mediante *POST* cuando creamos la variable *$sql* que utilizamos para la consulta:

*$sql = "DELETE FROM directorio WHERE id = ". $\_POST['id'];*

Efectuamos la consulta:

*$res = mysql\_query ($sql) or die(mysql\_error( ));*

Posteriormente, informamos del éxito en la eliminación del registro:

*echo "Registro eliminado exitosamente.";*

Cerramos la conexión con MySQL y utilizamos la función *exit* para que PHP termine de interpretar el documento .php:

*mysql\_close($cnx);  
 exit;  
}*

Guarde el documento con el nombre *eliminar.php* en la carpeta *phpmysql/scripts* de su servidor local y realice los siguientes cambios en el archivo *index.php* (los cambios están indicados en negrita y cursiva):

|  |
| --- |
| <!DOCTYPE html>  <html lang="es">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>Agenda Bootstrap</title>  <link rel="stylesheet" href="css/bootstrap.min.css">  <link rel="stylesheet" href="css/styles.css">  <link rel="shortcut icon" href="favicon.ico" />  ***<script src="js/jquery-1.9.1.js"></script>***  ***<script type="text/javascript">***  ***$(document).ready(function()***  ***{***  ***cargarContactos();***  ***});***    ***function cargarContactos()***  ***{***  ***$.get("scripts/consulta.php",***  ***function(resultado){***  ***$("#listado").html(resultado);***  ***});***  ***}***    ***function eliminarContacto(id,nick)***  ***{***  ***var eliminar = confirm("¿Seguro que desea eliminar el Contacto [" + nick + "]?")***  ***if (eliminar) {***  ***$("#mensaje").html("<img src='img/ajax-loader.gif' alt='procesando...' />");***  ***$.ajax({***  ***type: "POST",***  ***url: "scripts/eliminar.php",***  ***data: "id="+id,***  ***success: function(resultado){***  ***$("#mensaje").html(resultado);***  ***cargarContactos();***  ***}***  ***});***  ***}***  ***}***  ***</script>***  </head>  <body>  <div class="container">  <div class="row text-center color1">  <h1>AGENDA ELECTRONICA</h1>  </div>  <div class="row text-center color1">  <h2>LISTA DE CONTACTOS</h2>  </div>  <div id="mensaje" class="row text-center">  <!-- Aqui va el mensaje de eliminacion -->  </div>  ***<div id="listado" class="table-responsive">***  ***<!-- Aqui va la consulta realizada con AJAX -->***  ***</div>***  <div class="text-center">  <a class="btn btn-default" href="nuevo.php" role="button">Nuevo Contacto</a>  </div>  </div>  </body>  </html> |

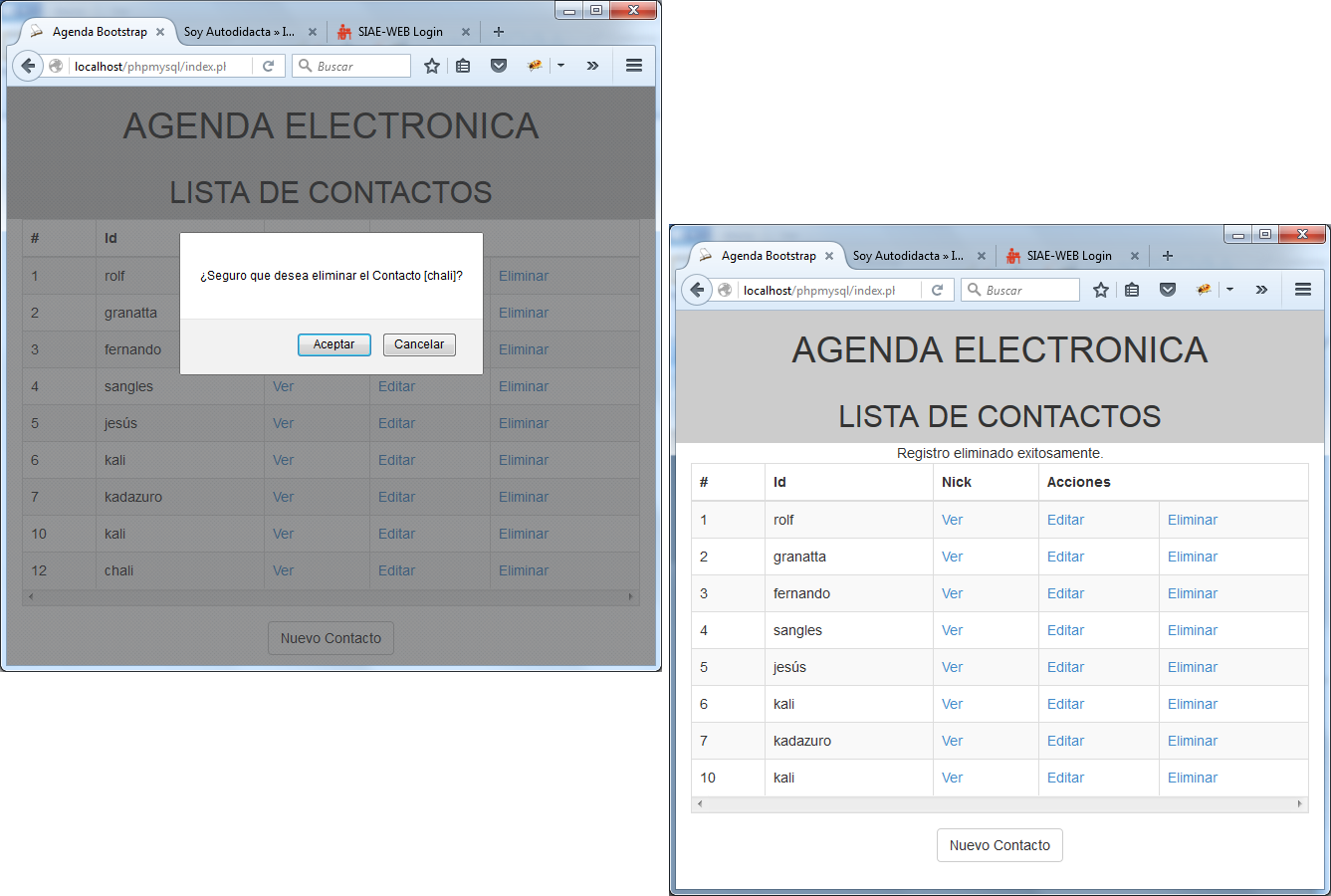
Finalmente abra su editor de texto e ingrese el siguiente código para el archivo *consulta.php*:

|  |
| --- |
| <?php  include\_once("../includes/config.php");  include\_once("../includes/funciones.php");  //nos conectamos a mysql  $cnx = conectar ();  //consulta.  $sql = "SELECT id,nick FROM directorio ORDER BY id ASC";  $res= mysql\_query($sql) or die (mysql\_error());    echo "<table class=\"table table-bordered table-striped\">\n";  echo "<thead>\n";  echo "<tr>\n";  echo "<th>#</th>\n";  echo "<th>Id</th>\n";  echo "<th>Nick</th>\n";  echo "<th colspan=\"3\">Acciones</th>\n";  echo "</tr>\n";  echo "</thead>\n";  echo "<tbody>\n";  if( mysql\_num\_rows($res) > 0){  //impresión de los datos.  while (list($id,$nick) = mysql\_fetch\_array($res)) {  echo "<tr><td>$id</td>\n";  echo "<td>".utf8\_encode($nick)."</td>\n";  echo "<td><a href='ver.php?id=$id'>Ver</a></td>\n";  echo "<td><a href='editar.php?id=$id'>Editar</a></td>\n";  echo "<td><a href='#' onclick=\"eliminarContacto($id,'$nick')\">Eliminar</a></td></tr>\n";  }  }else{  echo "<td colspan='5' align='center'>No se obtuvieron resultados</td>";  }  echo "</tbody>\n";  echo "</table>\n";  mysql\_close($cnx);  ?> |

Abra en su navegador el documento de listado tecleando [*http://localhost/phpmysql/index.php*](http://localhost/phpmysql/index.php). Cuando seleccione un registro para eliminar, será enviado a la función *eliminarContacto(),* la misma que llama al documento *eliminar.php* que acaba de realizar. La llamada se realiza a través de AJAX.

La llamada devolverá el texto "Registro eliminado exitosamente.", si no existió algún error al ejecutar la consulta, en caso contrario, devolverá el mensaje de error.

La Figura 4.13. muestra la ejecución del proceso indicado anteriormente.



**Figura 4.13.**  
*Listado, eliminación de registro y mensaje de confirmación de borrado.*

# Bibliografía:

Pixelar.me, Marcelo. (10 de febrero de 2008). *Pixelar.me*. Obtenido de http://pixelar.me/wampserver-guia-de-descarga-e-instalacion/

1. Expresar un número de la forma 3.402823466E+38 es equivalente a expresarlo de la forma 3.402823466x1038. [↑](#footnote-ref-1)
2. **Nota:** Recuerde que para acceder a contenidos alojados en <http://localhost> ha de haber inicializado previamente el servidor Apache. [↑](#footnote-ref-2)