

Prácticas

Estilo: ▼ Tamaño letra: ▼ Interlineado: ▼

Desarrollo de Aplicaciones Web

Práctica 9: PHP 3 (MySQL y acceso a una base de datos)

1. Objetivos

- Aprender a administrar una base de datos con MySQL.
- Conocer algunas herramientas que ayudan a administrar una base de datos de MySQL.
- Aprender a acceder a una base de datos desde PHP con ext/mysql y ext/mysqli.
- Aprender a realizar una consulta SELECT y mostrar el resultado en una página web.
- Aprender a trabajar con ficheros de configuración .ini.

2. Recursos

¿Qué tipos de datos admite MySQL? ¿Qué funciones existen en MySQL?

- **MySQL Developer Zone** [\(1\)](#): sitio web oficial de MySQL con información para desarrolladores.
- **MySQL Cheat Sheet** [\(2\)](#): resumen de los tipos de datos y de las diversas funciones (de fecha y hora, matemáticas, etc.) disponibles en MySQL. También incluye una lista de las funciones disponibles en PHP para conectarse a MySQL y ejemplos de la consulta SELECT.
- **MySQL Cheat Sheet** [\(3\)](#): resumen de los principales comandos de MySQL.
- **MySQL Quick Reference** [\(4\)](#): guía de referencia rápida de lo más importante de MySQL (tipos de datos, funciones para usar en sentencias SELECT y WHERE, lenguaje de definición de datos, lenguaje de manipulación de datos, comandos de usuario y comandos de transacciones y bloqueos).
- **MySQL Reference Sheet** [\(5\)](#): resumen de los tipos de datos y funciones de MySQL.

¿Qué es el lenguaje SQL? ¿Qué sintaxis tiene el lenguaje SQL?

- **SQL** [\(6\)](#): definición en la Wikipedia del lenguaje SQL.
- **W3Schools** [\(7\)](#): cursos de aprendizaje y guías de referencia de diversas tecnologías empleadas en la programación web. Incluye un tutorial y temas avanzados sobre SQL [\(8\)](#).
- **SQLCourse** [\(9\)](#): sencillo tutorial sobre el lenguaje SQL.
- **SQL Cheat** [\(10\)](#): resumen de las sentencias principales del lenguaje SQL.

¿Qué herramientas existen para trabajar con MySQL?

- **phpMyAdmin** [\(11\)](#): permite crear y gestionar una base de datos en MySQL a través de una página web.
- **phpMinAdmin** [\(12\)](#): script escrito en PHP en un solo fichero que permite administrar una base de datos en MySQL, similar a phpMyAdmin, pero no es tan potente ni tan complejo.
- **MySQL GUI Tools** [\(13\)](#): incluye las herramientas MySQL Administrator 1.2, MySQL Query Browser 1.2 y MySQL Migration Toolkit 1.1.
- **MySQL Workbench** [\(14\)](#): herramienta visual que permite diseñar, gestionar y documentar una base de datos en MySQL.

¿Qué funciones se emplean en PHP para acceder a MySQL?

- **PHP Mysql** [\(15\)](#): ext/mysql, la API original para usar MySQL desde PHP.

- **PHP Mysqli** [\(16\)](#): ext/mysqli, la extensión mejorada para usar MySQL desde PHP.
- **PHP MySQL Introduction** [\(17\)](#): tutorial de W3Schools sobre el uso de una base de datos de MySQL desde PHP.
- **ext/mysqli: Part I - Overview and Prepared Statements** [\(18\)](#) y **Using ext/mysqli: Part II - Extending mysqli** [\(19\)](#): explica de forma detallada y clara el empleo de mysqli.

¿Existe algún peligro cuando se accede a una base de datos desde una página web?

- **Inyección SQL** [\(20\)](#): definición en la Wikipedia de este tipo de vulnerabilidad informática.
- **Inyección de SQL** [\(21\)](#): explica qué es la inyección de SQL y algunas técnicas de protección con PHP.
- **SQL Injection Cheat Sheet** [\(22\)](#): resumen de algunos métodos de inyección de SQL para distintos sistemas gestores de bases de datos.

3. ¿Qué tengo que hacer?

MUY IMPORTANTE

Aunque se haga referencia a MySQL, desde octubre de 2015 [\(23\)](#) XAMPP ya no incluye MySQL. En su lugar incluye MariaDB, que para la realización de las prácticas de esta asignatura es totalmente equivalente, es decir, todo lo que se dice sobre MySQL se puede aplicar a MariaDB. Los comandos y las herramientas son los mismos.

MariaDB es un *drop-in replacement* [\(24\)](#) de MySQL: una instalación de MySQL se puede sustituir por MariaDB sin tener que realizar tareas de reprogramación o reconfiguración en el software que usaba previamente MySQL y sin afectarle de forma negativa [\(25\)](#).

¿Por qué se cambió MySQL por MariaDB?

La compañía MySQL fue adquirida por Sun Microsystems en 2008, y esta a su vez fue adquirida por Oracle Corporation en 2010. Debido al miedo de que MySQL dejase de ser un proyecto de código abierto, en 2009 se inició el proyecto MariaDB como una “bifurcación” (*fork*) de MySQL.

MySQL es el sistema gestor de bases de datos de código abierto más popular en la actualidad. MySQL está disponible para distintos sistemas operativos, como Linux, Mac OS X, Solaris, Windows y otros más. MySQL es muy popular en el desarrollo de aplicaciones web, ya que forma parte como sistema gestor de bases de datos de las plataformas LAMP, BAMP, MAMP y WAMP.

En esta práctica tienes que crear una base de datos en MySQL para almacenar los datos que emplea el sitio web. La base de datos se tiene que llamar “pibd” y tiene que tener las siguientes tablas (se indica el tipo de dato más apropiado para cada columna, pero se puede cambiar según la necesidad de cada uno):

TABLA USUARIOS

IdUsuario

: entero, autoincremento y clave primaria.

NomUsuario

: texto (longitud máxima 15 caracteres) y valor único.

Clave

: texto (longitud máxima 15 caracteres).

Email

: texto.

Sexo

: entero pequeño.

FNacimiento

: fecha.

Ciudad

: texto; no se relaciona con una tabla porque los nombres de todas las ciudades del mundo es imposible de conocer.

Pais

: entero y clave ajena a Paises.

Foto

: texto; contiene el nombre y quizás la ruta del fichero que almacena la foto, el uso de este campo depende de la implementación que realice el alumno.

FRegistro

: marca de tiempo (fecha y hora); fecha de registro en el sistema del usuario.

Estilo

: entero y clave ajena a Estilos.

TABLA PAISES**IdPais**

: entero, autoincremento y clave primaria.

NomPais

: texto.

TABLA ESTILOS**IdEstilo**

: entero, autoincremento y clave primaria.

Nombre

: texto.

Descripcion

: texto.

Fichero

: texto; contiene el nombre y quizás la ruta del fichero que almacena la hoja de estilo, el uso de este campo depende de la implementación que realice el alumno.

TABLA ALBUMES**IdAlbum**

: entero, autoincremento y clave primaria.

Titulo

: texto.

Descripcion

: texto.

Usuario

: entero y clave ajena a Usuarios.

TABLA FOTOS**IdFoto**

: entero, autoincremento y clave primaria.

Titulo

: texto.

Descripcion

: texto.

Fecha

: fecha; fecha en la que fue tomada la foto, se puede dejar en blanco.

Pais

: entero y clave ajena a Paises; país en el que se tomo la foto, se puede dejar en blanco.

Album

: entero y clave ajena a Albumes; una foto siempre tiene que estar asociada a un álbum.

Fichero

: texto; contiene el nombre y quizás la ruta del fichero que almacena la foto, el uso de este campo depende de la implementación que realice el alumno.

Alternativo

: texto (longitud mínima 10 caracteres); texto alternativo para ser empleado con el atributo alt de HTML.

FRegistro

: marca de tiempo (fecha y hora); fecha de registro en el sistema de la foto, se emplea para mostrar las últimas cinco fotos subidas en la página principal.

TABLA SOLICITUDES**IdSolicitud**

: entero, autoincremento y clave primaria.

Album

: entero y clave ajena a Albumes.

Nombre

: texto (longitud máxima 200 caracteres); también puede ser dos campos separados (nombre, apellidos).

Titulo

: texto (longitud máxima 200 caracteres).

Descripcion

: texto (longitud máxima 4000 caracteres).

Email

: texto (longitud máxima 200 caracteres).

Direccion

: texto; también puede ser campos separados con la estructura típica (calle, número, piso, puerta, código postal, localidad, provincia y país).

Color

: texto.

Copias

: entero.

Resolucion

: entero.

Fecha

: fecha.

IColor

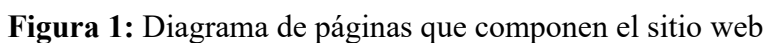
: booleano.

FRegistro

: marca de tiempo (fecha y hora); fecha de registro en el sistema de la solicitud.

Coste

: real.



El mantenimiento de las tablas **Países** y **Estilos** se realiza directamente a través de la base de datos: los datos se introducen directamente en la tabla. Respecto a las otras tablas, el mantenimiento se realizará en una próxima práctica, pero por ahora **introduce los datos directamente en las tablas para poder hacer pruebas.**

Página principal

5/26

Página con el formulario de registro como nuevo usuario

En la lista desplegable para seleccionar el país, mostrar los países a partir de la tabla **Países** de la base de datos.

Página con el formulario de búsqueda

En la lista desplegable para seleccionar el país, mostrar los países a partir de la tabla **Países** de la base de datos.

Página con el listado resultado de una búsqueda

A partir de los datos recibidos desde la [Página con el formulario de búsqueda] (título, fecha y/o país) debe realizar una búsqueda en la tabla **Fotos** y mostrar el resultado obtenido.

Página detalle foto

Muestra toda la información sobre una foto seleccionada en la página anterior (foto, título, descripción, fecha, país, álbum de fotos y usuario al que pertenece).

Página control de acceso

Controla el acceso a la parte privada para los usuarios registrados (los que figuran en la tabla **Usuarios**). Si el usuario está registrado, mediante una redirección en la parte del servidor se debe mostrar la página con el menú de usuario registrado; si el usuario no está registrado, mediante una redirección en la parte del servidor se debe mostrar la página principal del sitio web o la página con el formulario de acceso con un mensaje de error que explique al usuario lo que ha pasado.

Página “Solicitar álbum”

Muestra un listado con todos los álbumes del usuario.

Página “Crear álbum”

Contiene un formulario con los datos necesarios para crear un álbum (título y descripción).

Y debes crear las siguientes páginas nuevas:

Página “Perfil usuario”

Muestra el perfil público (nombre de usuario, foto, fecha de incorporación) del usuario seleccionado. Se debe mostrar un listado con todos los álbumes del usuario.

Página “Ver álbum” (pública)

Muestra todas las fotos que contiene un álbum. Se deben mostrar la descripción del álbum, los países en los que se han tomado las fotos que contiene el álbum y el intervalo de tiempo de las fotos (la fecha más antigua y la fecha más actual).

Página “Configurar”

Muestra un listado con todos los estilos alternativos que puede seleccionar el usuario.

Página “Mis datos”

Contiene un formulario similar a [Página con el formulario de registro como nuevo usuario]. Permite al usuario visualizar sus datos de registro y modificarlos (**en una próxima práctica**). En el formulario de modificación se tienen que mostrar los datos actuales del usuario. El formulario de modificación y el formulario de registro como nuevo usuario deben realizar las mismas validaciones, por lo que se recomienda que se traten como un único formulario (aislar la definición del formulario en un único fichero).

Página “Mis álbumes”

Muestra un listado con todos los álbumes del usuario. Se debe mostrar la descripción de cada álbum.

Página “Mis fotos”

Muestra un listado con todas las fotos del usuario. Se debe indicar el álbum al que pertenece cada foto.

Página “Ver álbum” (privada)

Desde la [Página “Mis álbumes”] y [Página “Mis fotos”], muestra todas las fotos que contiene un álbum. Se deben mostrar la descripción del álbum, una lista de todos los países en los que se han tomado las fotos que contiene el álbum y el intervalo de tiempo de las fotos (desde la fecha más antigua y hasta la fecha más reciente).

Página “Añadir foto a álbum”

Contiene un formulario con los datos necesarios para añadir una foto (título, descripción, fecha, país, foto, texto alternativo y álbum al que se añade la foto). En esta práctica no se tiene que realizar la subida de la foto ni la página de respuesta, eso se hará en otra práctica.

Por último, en la práctica anterior empleabas las sesiones de PHP para conservar el estilo seleccionado por el usuario. Por ahora, la selección del estilo no se puede cambiar, la tienes que almacenar con los datos de los usuarios, en el campo **Estilo** de la tabla **Usuarios**. En **una próxima práctica** tendrás que permitir que el usuario seleccione el estilo alternativo que desea.

4. ¿Cómo lo hago?

4.1. Creación de la base de datos desde la línea de comandos

Para crear la base de datos en MySQL tienes diferentes alternativas. Por un lado, puedes acceder a MySQL a través de MySQL monitor que se encuentra en el directorio `\xampp\mysql\bin`. En la Figura 2 podemos ver una sesión de ejecución con los siguientes comandos:

- `mysql -u root`: inicia la conexión a la base de datos con el usuario root.
- `show databases;`: muestra las bases de datos que existen.
- `use library;`: selecciona una base de datos.
- `show tables;`: muestra las tablas que existen en la base de datos.
- `describe books;`: muestra el esquema de la tabla.

```

C:\xampp\mysql\bin>mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.67-community MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| cdcol |
| library |
| mysql |
| phpmyadmin |
| test |
| webauth |
+-----+
7 rows in set (0.00 sec)

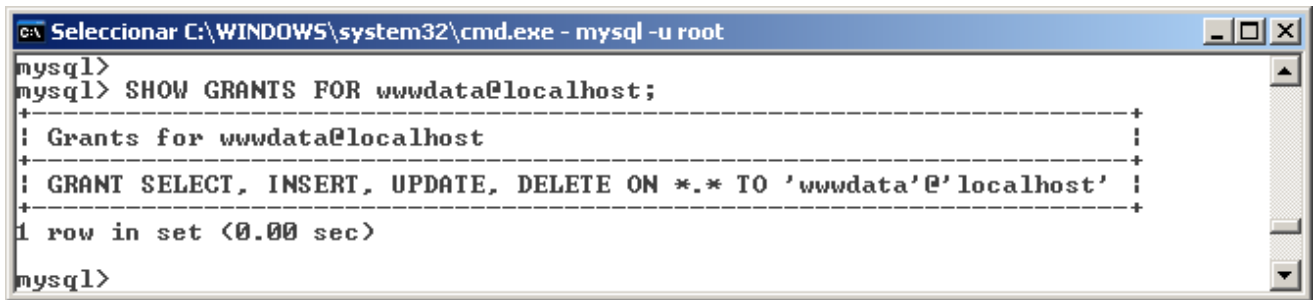
mysql> use library;
Database changed
mysql> show tables;
+-----+
| Tables_in_library |
+-----+
| authors |
| books |
| categories |
+-----+
3 rows in set (0.00 sec)

mysql> describe books;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id | int(11) | NO | PRI | NULL | auto_increment |
| title | varchar(250) | NO | | NULL | |
| abstract | varchar(2000) | YES | | NULL | |
| author | int(11) | NO | | NULL | |
| category | int(11) | NO | | NULL | |
| publisher | varchar(250) | YES | | NULL | |
| year | int(11) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.06 sec)

mysql>

```

Figura 2: Acceso a MySQL desde la línea de comandos



```

C:\WINDOWS\system32\cmd.exe - mysql -u root
mysql>
mysql> SHOW GRANTS FOR wwwdata@localhost;
+-----+
| Grants for wwwdata@localhost |
+-----+
| GRANT SELECT, INSERT, UPDATE, DELETE ON *.* TO 'wwwdata'@'localhost' |
+-----+
1 row in set (0.00 sec)

mysql>

```

Figura 3: Privilegios de un usuario en MySQL

Para crear la base de datos debemos emplear el lenguaje de definición de datos (*Data Definition Language*, DDL) de SQL que permite definir las estructuras de la base de datos que almacenarán los datos. En concreto, los comandos SQL más importantes que se utilizan para crear y mantener una base de datos son:

- CREATE DATABASE: crea una base de datos con el nombre dado.
- DROP DATABASE: borra todas las tablas en la base de datos y borra la base de datos.
- CREATE TABLE: crea una tabla con el nombre dado.
- ALTER TABLE: permite cambiar la estructura de una tabla existente.
- DROP TABLE: borra una o más tablas.

Además, MySQL es un sistema gestor de bases de datos que funciona con usuarios y permisos. Cuando se realiza una conexión a una base de datos desde una página web se debe emplear un usuario especial para reducir los riesgos de seguridad y evitar que un usuario malintencionado pueda modificar o incluso eliminar toda una base de datos. El usuario para conectarse desde una página web debe tener otorgados únicamente los permisos para manipular los datos (SELECT, INSERT, UPDATE y DELETE) y NO los permisos para cambiar la estructura (CREATE, ALTER, etc.) o administrar (GRANT, SHUTDOWN, etc.) la base de datos.

En MySQL se puede crear una cuenta de usuario de tres formas:

- Usando el comando GRANT.
- Manipulando las tablas de permisos de MySQL directamente.
- Usar uno de los diversos programas proporcionados por terceras partes que ofrecen capacidades para administradores de MySQL, como phpMyAdmin.

Desde la línea de comandos el método preferido es usar el comando GRANT, ya que es más conciso y menos propenso a errores que manipular directamente las tablas de permisos de MySQL.

Por ejemplo, las siguientes instrucciones crean un nuevo usuario llamado `wwwdata` con contraseña `abc`, que sólo se puede usar cuando se conecte desde el equipo local (`localhost`) y le otorga únicamente los permisos SELECT, INSERT, UPDATE y DELETE sobre todas las bases de datos alojadas en el servidor:

```

# Crea un nuevo usuario
CREATE USER 'wwwdata'@'localhost' IDENTIFIED BY 'abc';

# Otorga los permisos para poder manipular los datos
# sobre todas las bases de datos (*.*)
GRANT SELECT, INSERT, UPDATE, DELETE ON *.* TO 'wwwdata'@'localhost'
    IDENTIFIED BY 'abc' WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
    MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0 ;

# Recarga los permisos de las tablas (en principio, no es necesario porque

```



```
# GRANT debe hacerlo de forma automática)
FLUSH PRIVILEGES;
```

Una vez creado un usuario, podemos consultar sus permisos con el comando SHOW GRANTS, tal como podemos ver en la Figura 3.

Desde la línea de comandos también se pueden ejecutar otros programas, como mysqladmin, mysqlcheck, mysqldump o mysqlshow.

4.2. Creación de la base de datos desde phpMyAdmin

phpMyAdmin es una herramienta escrita en PHP que permite la administración de una base de datos de MySQL a través de páginas web, ya sea en local o de forma remota a través de Internet. Es un desarrollo de código abierto y está disponible bajo la licencia GPL.

En la Figura 4 podemos ver la pantalla principal de la aplicación. En el panel de la izquierda aparecen las bases de datos que existen y entre paréntesis se indica el número de tablas que posee cada base de datos. En la parte principal de la pantalla se indica la versión del servidor de MySQL y el usuario que se está empleando para conectarse. En XAMPP, por defecto se emplea el usuario “root” sin contraseña, lo que supone una vulnerabilidad del sistema ya que facilita un posible ataque. Para evitarlo, es conveniente asignar una contraseña al usuario “root” en MySQL y configurar la contraseña para phpMyAdmin en el fichero config.inc.php.

Además, en la página principal existen varias funciones, como crear una nueva base de datos, modificar los privilegios o importar y exportar el esquema y los datos de una base de datos.

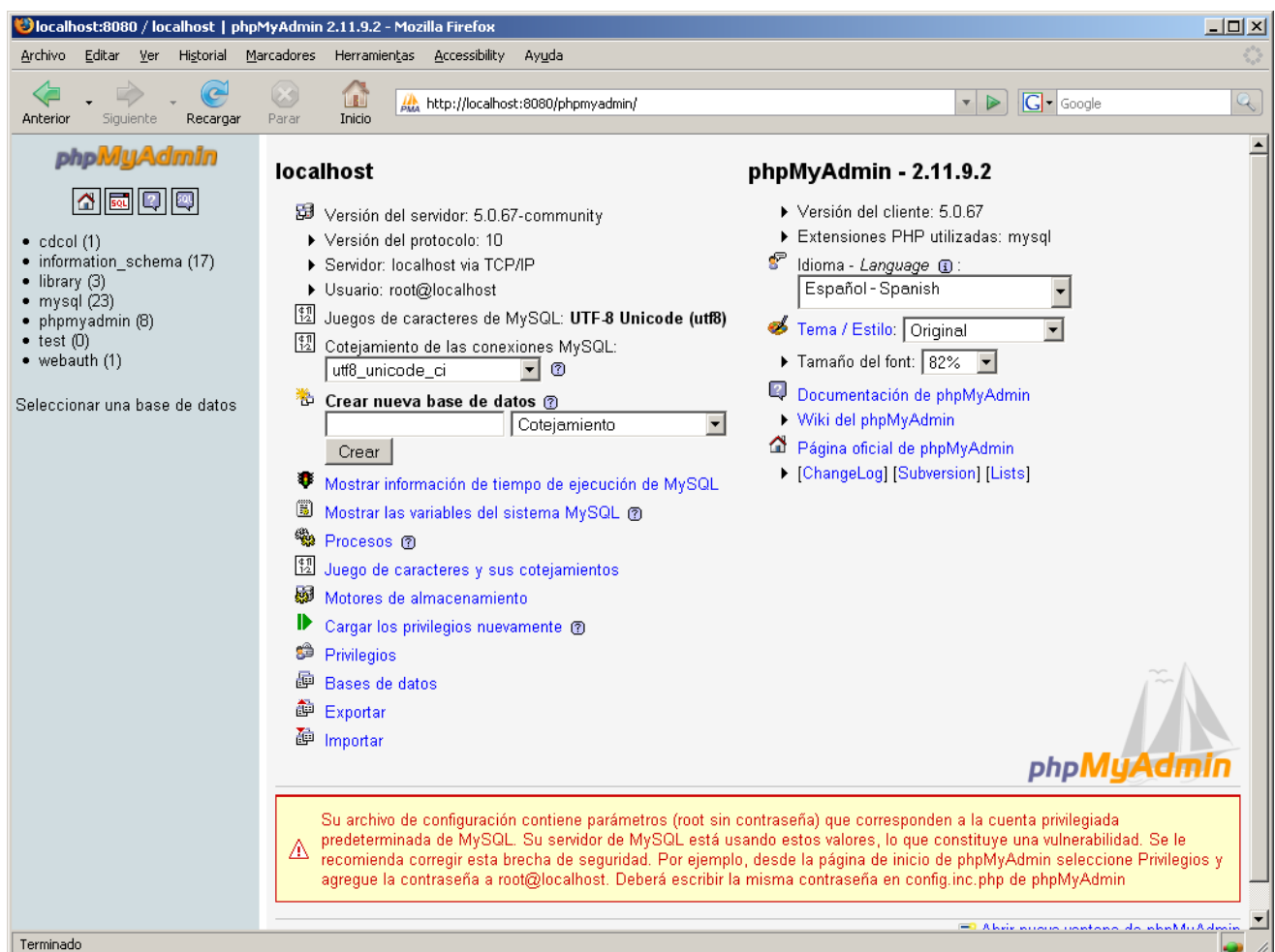
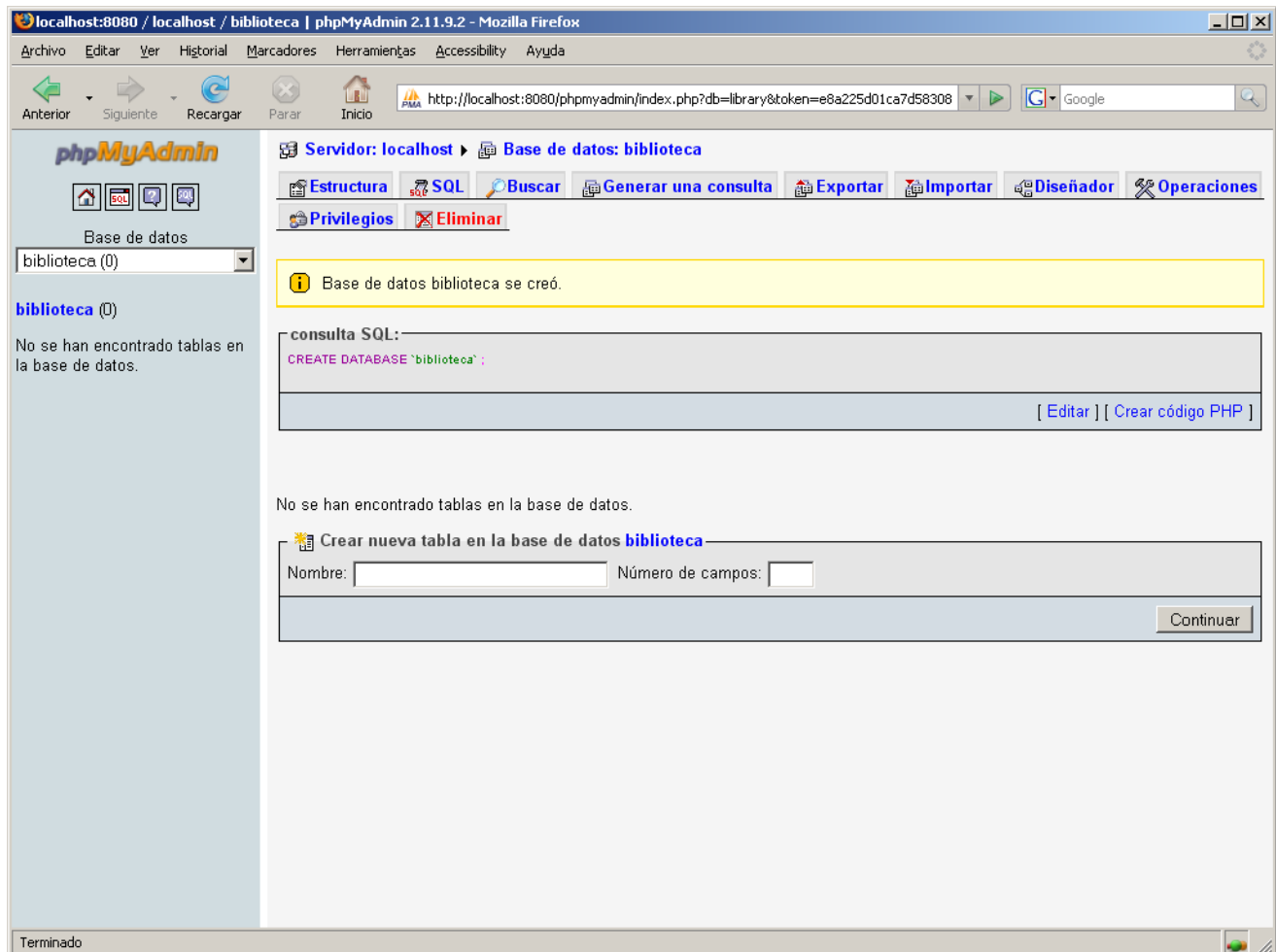


Figura 4: Página principal de phpMyAdmin

Desde la pantalla principal se puede crear una nueva base de datos. Una vez creada, aparece la pantalla que podemos ver en la Figura 5; en esta pantalla se visualiza la sentencia SQL que ha creado la base de datos y se puede indicar el nombre para una nueva tabla en la base de datos recién creada. En este último caso, también hay que indicar el número de campos (columnas) que se quiere que tenga la tabla; más adelante se pueden añadir más campos en cualquier momento.

**Figura 5:** Creación de una nueva base de datos en phpMyAdmin

En la Figura 6 podemos ver la pantalla de creación de una nueva tabla con dos campos. En esta pantalla se tiene que indicar la definición de cada campo (columna) de la tabla, como el nombre del campo, el tipo de dato, si admite valor nulo, si es clave primaria, etc. Esta pantalla cambia de aspecto según el número de campos que tenga la tabla; por ejemplo, en la Figura 7 podemos ver la misma pantalla pero cuando una tabla posee siete campos, en vez de una disposición vertical la definición de los campos adquiere una disposición horizontal.

Además, se tiene que seleccionar el motor de almacenamiento para la tabla. MySQL permite seleccionar diferentes motores de almacenamiento. La principal diferencia entre los distintos motores reside en el soporte de las transacciones, el manejo de las claves ajenas y el particionamiento de las tablas.

Figura 6: Creación de una nueva tabla en phpMyAdmin

localhost:8080 / localhost / biblioteca / libros | phpMyAdmin 2.11.9.2 - Mozilla Firefox

Archivo Editar Ver Historial Marcadores Herramientas Accessibility Ayuda

Anterior Siguiente Recargar Parar Inicio

http://localhost:8080/phpmyadmin/index.php?db=library&token=e8a225d01ca7d58308

Google

phpMyAdmin

Base de datos
biblioteca (0)

biblioteca (0)

No se han encontrado tablas en la base de datos.

Servidor: localhost ▶ **Base de datos: biblioteca** ▶ **Tabla: libros**

Campo	Tipo	Longitud/Valores ¹	Cotejamiento	Atributos
	VARCHAR			
	VARCHAR			
	VARCHAR			
	VARCHAR			
	VARCHAR			
	VARCHAR			
	VARCHAR			

Comentarios de la tabla:

Motor de almacenamiento: MyISAM

Cotejamiento:

Grabar O Añadir 1 campo(s) Continuar

¹ Si el tipo de campo es "enum" o "set", por favor ingrese los valores usando este formato: 'a','b','c'...
Si alguna vez necesita poner una barra invertida ("\") o una comilla simple (") entre esos valores, siempre ponga una barra invertida. (Por ejemplo '\xyz' o 'a\b').

² Para valores predeterminados, por favor ingrese solamente un valor, sin caracteres de escape "\\\\" ni comillas, usando este formato: a

³ Por favor ingrese los valores para las opciones de transformación usando este formato: 'a','b','c'...
Si alguna vez requiere insertar un Backslash ("\") o comilla sencilla (") entre esos valores, use backslash (por ejemplo '\xyz' o 'a\b').

Para una lista de opciones de transformación disponibles y sus transformaciones MIME-type transformations, dé clic en [transformation descriptions](#)

Terminado

Figura 7: Creación de una nueva tabla en phpMyAdmin

En la Figura 8 podemos ver la pantalla de respuesta que aparece al crear una nueva tabla. En esta pantalla figura la sentencia SQL de creación de la tabla y también se puede modificar la estructura de la tabla recién creada.

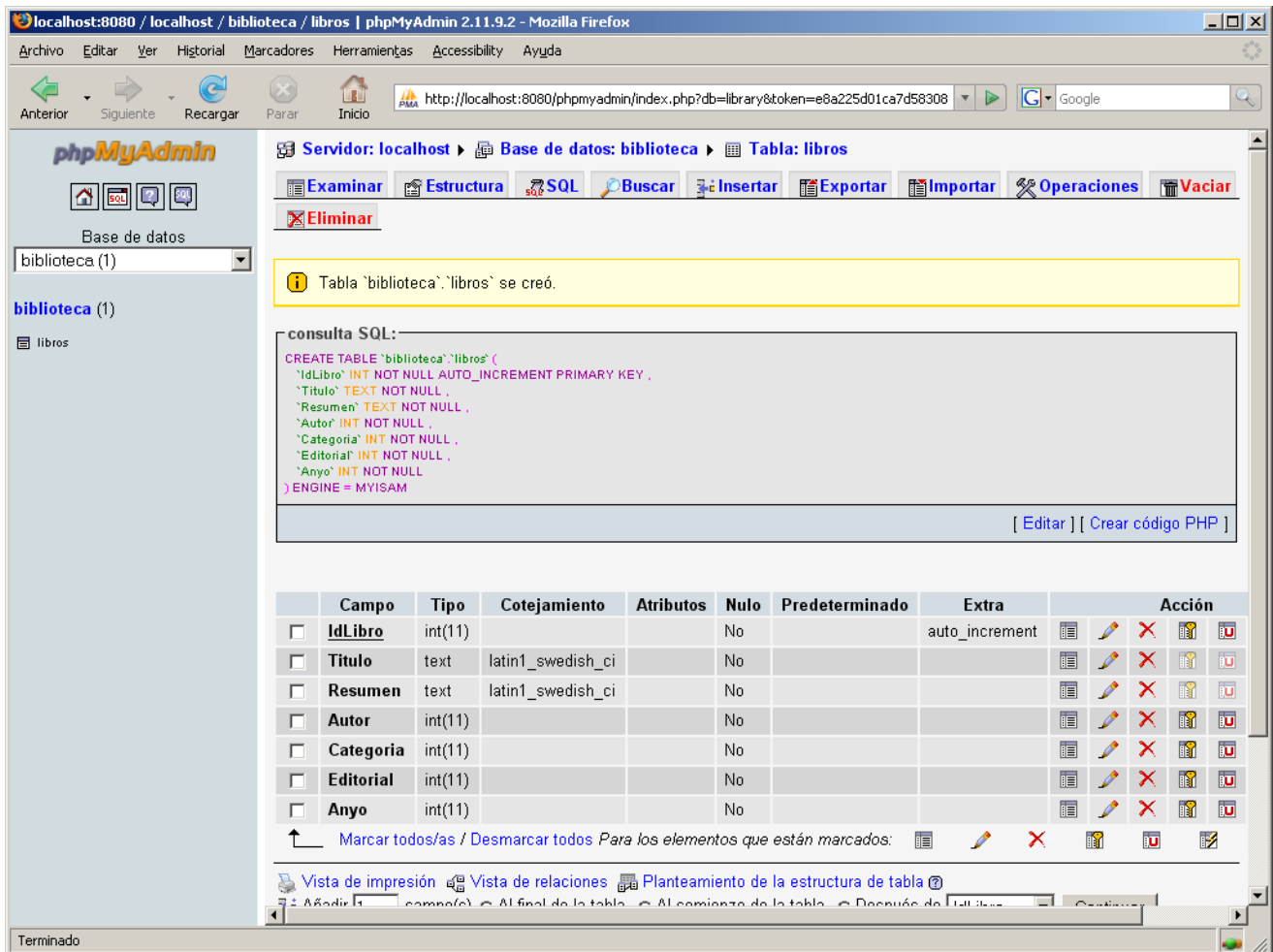


Figura 8: Creación de una nueva tabla en phpMyAdmin

Una vez creada una tabla se pueden insertar datos en la misma. Para ello se emplea la opción **Insertar** que muestra un formulario como el de la Figura 9. En este formulario aparecen todos los campos que componen una tabla y para cada campo se indica su tipo de dato. Cuando un campo es de tipo autoincremento la base de datos le asignará un valor de forma automática, pero de todas formas aparecerá en el formulario de inserción, por lo que se debe dejar vacío.

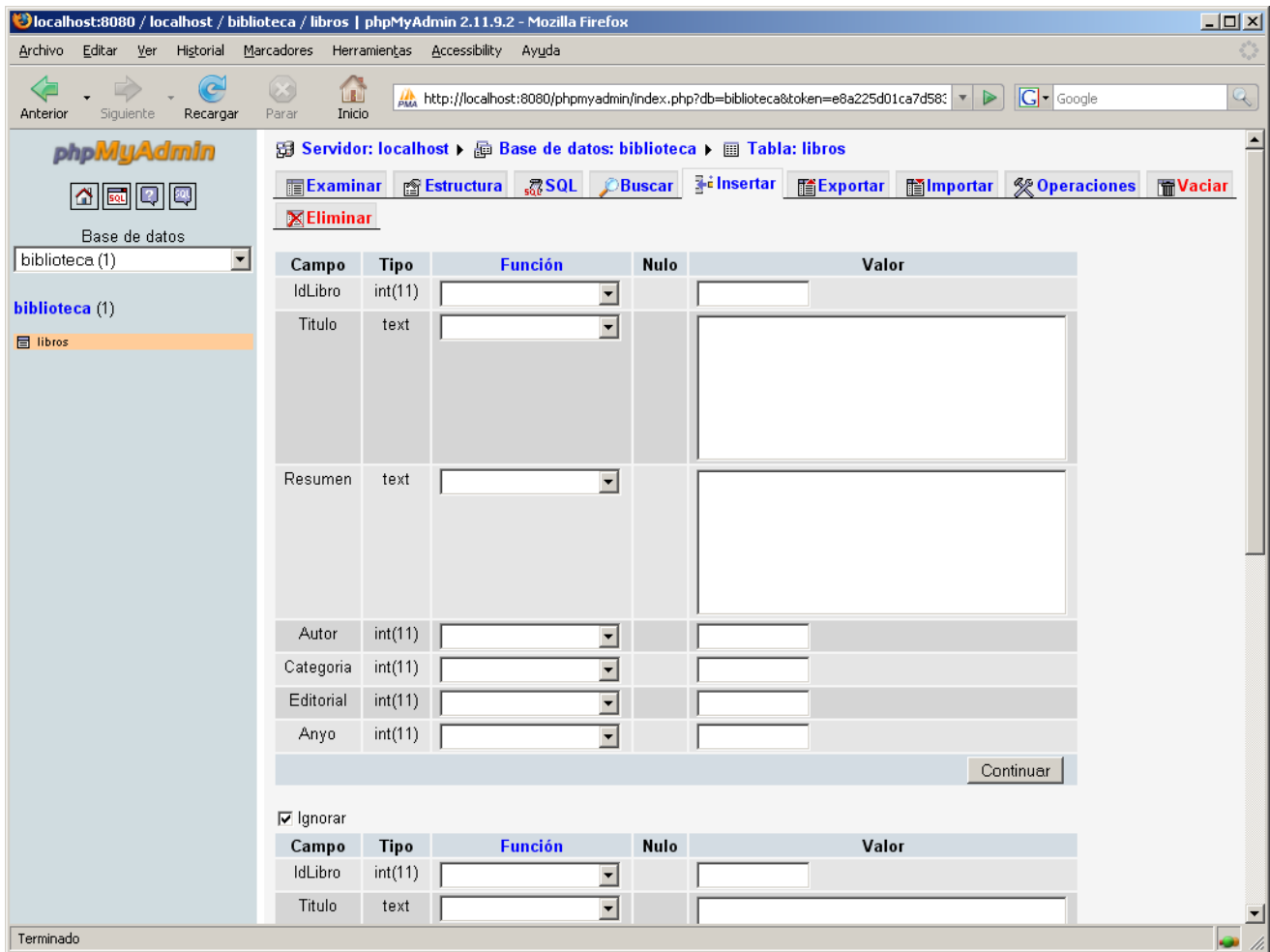


Figura 9: Inserción de datos en una tabla en phpMyAdmin

Por último, y tal como se ha explicado en el apartado anterior, se debe emplear un usuario específico para conectarse desde una página web, que tenga otorgados únicamente los permisos para manipular los datos (SELECT, INSERT, UPDATE y DELETE). En la Figura 10 podemos ver la pantalla de la opción Privilegios, donde se muestran todos los usuarios que existen y los permisos que poseen. Desde esta pantalla se puede acceder a la función agregar un nuevo usuario que vemos en la Figura 11.

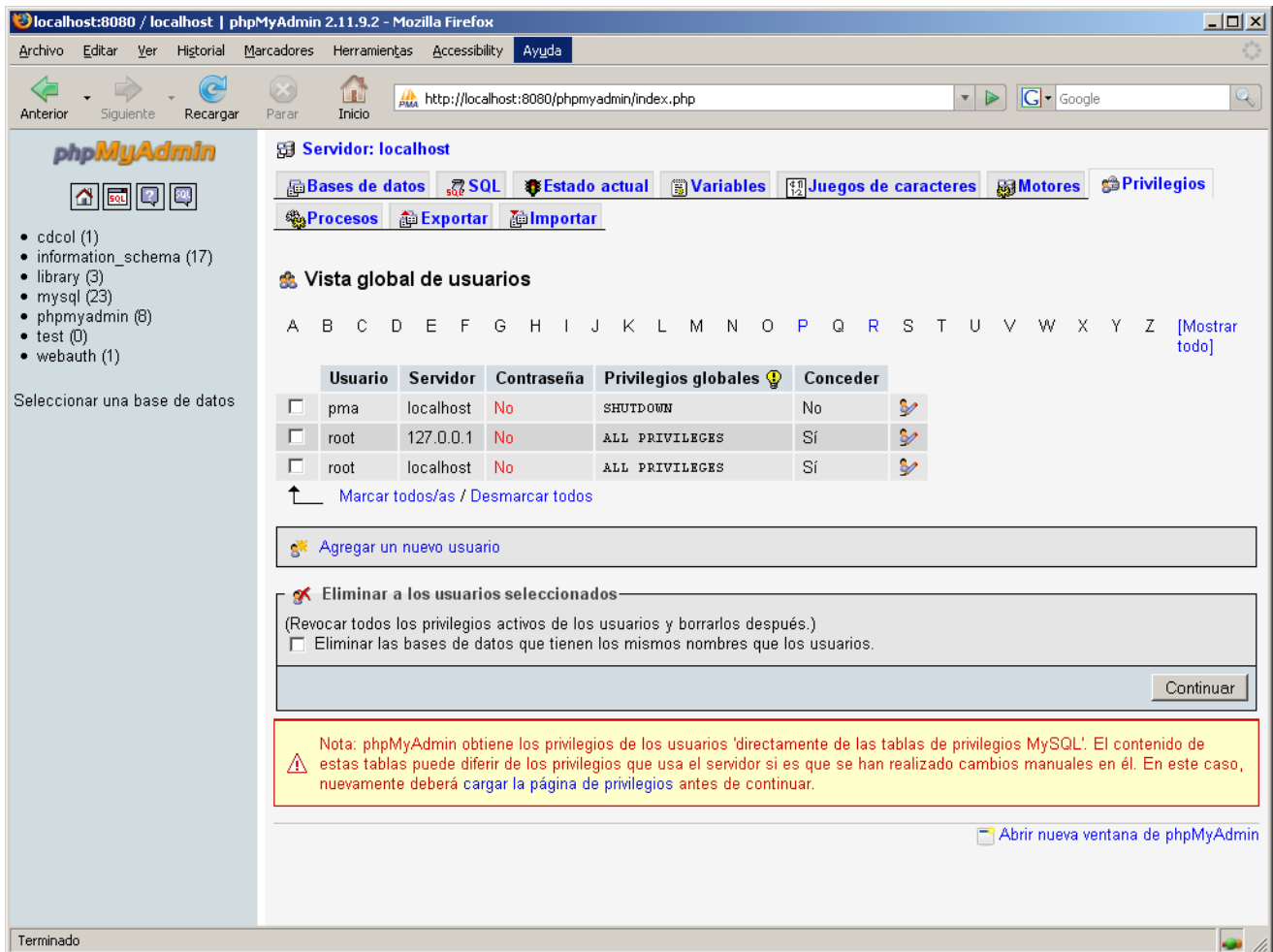


Figura 10: Privilegios en phpMyAdmin

Cuando se agrega un nuevo usuario se tiene que indicar su nombre, su contraseña, desde que ordenador se puede conectar y los privilegios globales de aplicación a cualquier base de datos. Los privilegios están divididos en tres grupos: para manipular los datos (SELECT, INSERT, UPDATE y DELETE), para cambiar la estructura de la base de datos (CREATE, ALTER, etc.) y para administrar la base de datos (GRANT, SHUTDOWN, etc.). En cualquier momento se pueden definir privilegios específicos para bases de datos concretas.

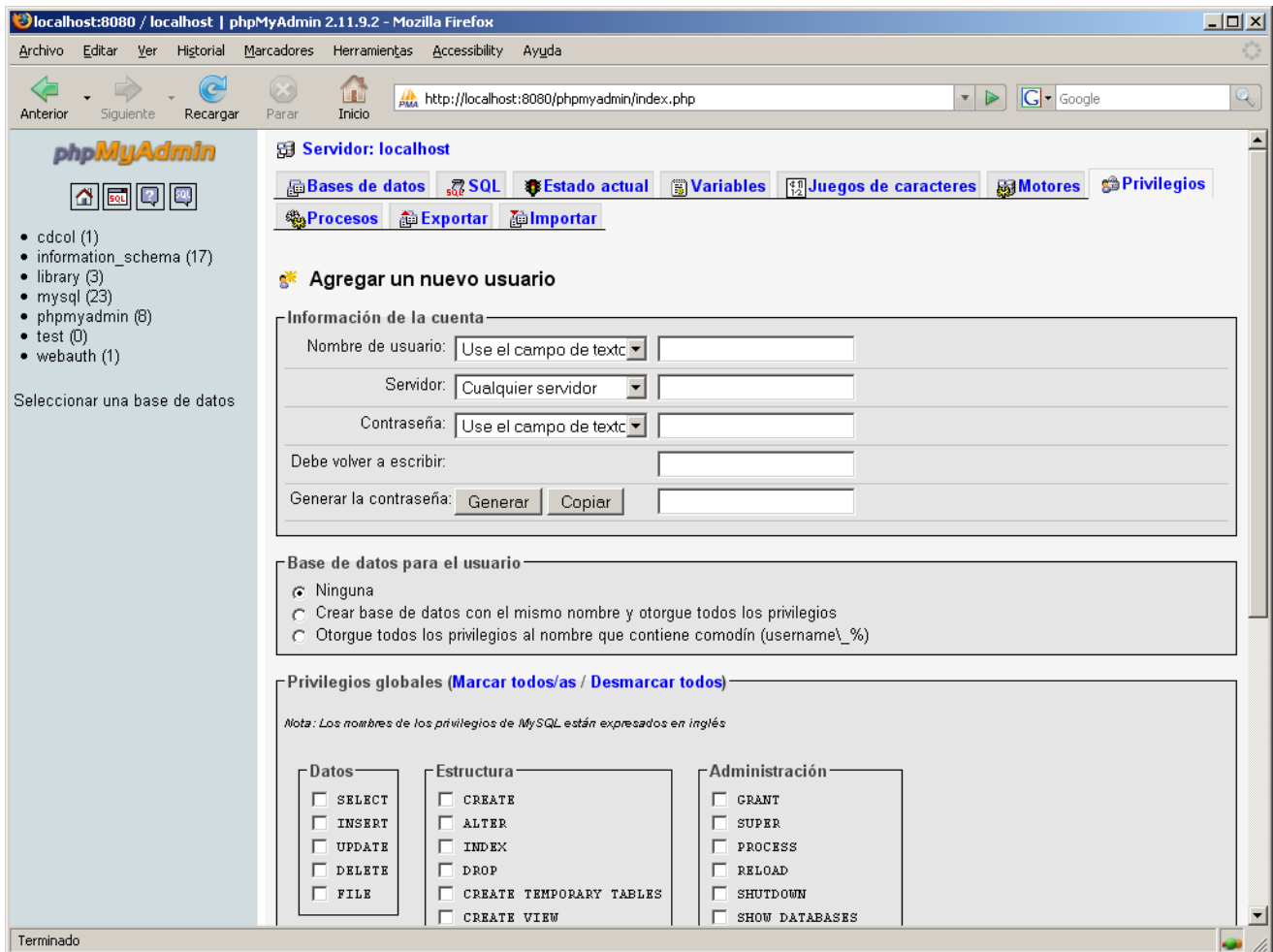


Figura 11: Agregar un nuevo usuario en phpMyAdmin

4.3. Acceso a la base de datos desde PHP

PHP ofrece diferentes mecanismos para acceder a una base de datos. Estos mecanismos se clasifican en dos grupos [\(26\)](#):

- Capas abstractas: son independientes de la base de datos que se esté utilizando, se usan las mismas funciones para realizar consultas y obtener datos.
- Extensiones de bases de datos específicas del proveedor: son específicas del sistema gestor de bases de datos que se esté utilizando, permiten utilizar características específicas y ofrecen el máximo rendimiento.

Existen distintas formas de acceder a una base de datos MySQL desde PHP, todas ellas con sus ventajas y desventajas. Intenta conocerlas todas y utiliza la que más te guste.

De forma nativa, PHP ofrece tres APIs diferentes para conectarse a MySQL [\(27\)](#): mysqli, PDO_MySQL y mysql; en la Figura [12](#) se muestra una comparativa de estas tres APIs. En la actualidad, se recomienda usar las extensiones mysqli o PDO_MySQL.

MUY IMPORTANTE: no se debe usar la antigua extensión mysql para nuevos desarrollos, ya que se considera obsoleta y no se desarrollan nuevas funcionalidades para ella, sólo se mantiene para corregir los fallos que aparecen.

	ext/mysqli	PDO_MySQL	ext/mysql
Versión de PHP donde se introdujo	5.0	5.1	2.0
Incluida con PHP 5.x	Sí	Sí	Sí
Estado de desarrollo	Activo	Activo	Sólo mantenimiento
Ciclo de vida	Activo	Activo	Anunciada como obsoleta durante mucho tiempo
Recomendada para nuevos proyectos	Sí	Sí	No
Interfaz POO	Sí	Sí	No
Interfaz procedimental	Sí	No	Sí
La API soporta el no bloqueo, consultas asíncronas con mysqlnd	Sí	No	No
Conexiones persistentes	Sí	Sí	Sí
La API soporta conjunto de caracteres	Sí	Sí	Sí
La API soporta sentencias preparadas del lado del servidor	Sí	Sí	No
La API soporta sentencias preparadas del lado del cliente	No	Sí	No
La API soporta procedimientos almacenados	Sí	Sí	No
API soporta sentencias múltiples	Sí	La mayoría	No
La API soporta transacciones	Sí	Sí	No
Las transacciones se pueden controlar con SQL	Sí	Sí	Sí
Soporta toda la funcionalidad de MySQL 5.1+	Sí	La mayoría	No

Figura 12: Comparativa mysqli, PDO_MySQL y mysql

4.3.1. Acceso con ext/mysql

¡Cuidado! Este método se explica únicamente por razones históricas: ha sido la API más empleado durante muchos años, existen multitud de libros, tutoriales y ejemplos que hacen uso de esta API, así que es muy probable que alguna vez tengas que trabajar con código que la utiliza. **Desde PHP 5.5.0 está declarada como obsoleta y está desaconsejado su uso; en su lugar se recomienda utilizar ext/mysqli para cualquier desarrollo nuevo.**

Desde PHP se puede acceder fácilmente a una base de datos en MySQL empleando las más de 50 funciones que existen en ext/mysql. Las principales funciones que se emplean para acceder a una base de datos son:

- `mysql_connect(servidorBD, usuario, contraseña)`: abre una conexión con un servidor de bases de datos de MySQL, devuelve un identificador que se emplea en algunas de las siguientes funciones o FALSE en caso de error.
- `mysql_close(identificador)`: cierra una conexión con un servidor de MySQL, devuelve TRUE en caso de éxito y FALSE en caso contrario.
- `mysql_ping(identificador)`: verifica que la conexión con el servidor de bases de datos funciona, devuelve TRUE en caso de éxito y FALSE en caso contrario.
- `mysql_select_db(nombreBD, identificador)`: selecciona una base de datos, devuelve TRUE en caso de éxito y FALSE en caso contrario.
- `mysql_query(sentencia, identificador)`: ejecuta una sentencia SQL y devuelve un resultado (SELECT, SHOW, EXPLAIN o DESCRIBE, ...) o TRUE (INSERT, UPDATE, DELETE, ...) si todo es correcto, o FALSE en caso contrario.
- `mysql_fetch_array(resultado)`: recorre un resultado, devuelve un array que representa una fila (registro) o FALSE en caso de error (por ejemplo, llegar al final del resultado); al array se puede acceder de forma numérica (posición de la columna) o asociativa (nombre de la columna).
- `mysql_fetch_assoc(resultado)` y `mysql_fetch_row(resultado)`: ambas funciones son similares a la anterior `mysql_fetch_array(resultado)`, pero sólo permiten el acceso como array asociativo o con índices numéricos respectivamente.
- `mysql_affected_rows(identificador)`: devuelve el número de filas (tuplas) afectadas por la última operación si fue del tipo INSERT, UPDATE, etc., que no devuelven un resultado.

- `mysql_num_rows(resultado)`: devuelve el número de filas (tuplas) afectadas por la última operación si fue del tipo `SELECT`.
- `mysql_free_result(resultado)`: libera la memoria empleada por un resultado; en principio, se libera automáticamente al finalizar la página, es necesario si en una misma página se realizan varias consultas con resultados muy grandes.
- `mysqli_errno(identificador)`: devuelve el código del último error.
- `mysqli_error(identificador)`: devuelve una descripción del último error.

El siguiente ejemplo muestra como visualizar todo el contenido de una tabla en una página web. En concreto, se conecta al servidor local con el usuario `wwwwdata` sin contraseña, selecciona la base de datos `biblioteca`, recupera todo el contenido de la tabla `libros` y muestra los campos `Título` y `Resumen`:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de SELECT y MySQL</title>
</head>
<body>
<?php
// Se conecta al SGBD
if(!($iden = mysql_connect("localhost", "wwwwdata", "")))
    die("Error: No se pudo conectar");

// Selecciona la base de datos
if(!mysql_select_db("biblioteca", $iden))
    die("Error: No existe la base de datos");

// Sentencia SQL: muestra todo el contenido de la tabla "books"
$sentencia = "SELECT * FROM libros";
// Ejecuta la sentencia SQL
$resultado = mysql_query($sentencia, $iden);
if(!$resultado)
    die("Error: no se pudo realizar la consulta");

echo '<table>';
while($fila = mysql_fetch_assoc($resultado))
{
    echo '<tr>';
    echo '<td>' . $fila['Titulo'] . '</td><td>' . $fila['Resumen'] . '</td>';
    echo '</tr>';
}
echo '</table>';

// Libera la memoria del resultado
mysql_free_result($resultado);

// Cierra la conexión con la base de datos
mysql_close($iden);
?>
</body>
</html>
```

Nota: el ejemplo anterior no está mal, pero no es la mejor forma de combinar en una página web el código HTML y el código PHP de acceso a una base de datos. Lo correcto es emplear el patrón de arquitectura de software modelo-vista-controlador [\(28\)](#) (MVC).

El siguiente ejemplo es similar al anterior, pero emplea una función llamada `sql_dump_result(resultado)` que visualiza todo el contenido del resultado de una consulta `SELECT` en forma de tabla de HTML, sin tener que indicar uno a uno los campos que componen el resultado; además, la primera fila de la tabla creada contiene los nombres de los campos a modo de encabezados de las columnas de la tabla:

```

<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de SELECT y MySQL</title>
</head>
<body>
<?php
// Devuelve todas las filas de una consulta a una tabla de una base de datos
// en forma de tabla de HTML
function sql_dump_result($result)
{
    $line = '';
    $head = '';

    while($temp = mysql_fetch_assoc($result))
    {
        if(empty($head))
        {
            $keys = array_keys($temp);
            $head = '<tr><th>' . implode('</th><th>', $keys). '</th></tr>';
        }

        $line .= '<tr><td>' . implode('</td><td>', $temp). '</td></tr>';
    }

    return '<table>' . $head . $line . '</table>';
}

// Se conecta al SGBD
if(!$iden = mysql_connect("localhost", "wwwdata", ""))
    die("Error: No se pudo conectar");

// Selecciona la base de datos
if(!mysql_select_db("biblioteca", $iden))
    die("Error: No existe la base de datos");

// Sentencia SQL: muestra todo el contenido de la tabla "books"
$sentencia = "SELECT * FROM libros";
// Ejecuta la sentencia SQL
$resultado = mysql_query($sentencia, $iden);
if(!$resultado)
    die("Error: no se pudo realizar la consulta");

// Muestra el contenido de la tabla como una tabla HTML
echo sql_dump_result($resultado);

// Libera la memoria del resultado
mysql_free_result($resultado);

// Cierra la conexión con la base de datos
mysql_close($iden);
?>
</body>
</html>

```

4.3.2. Acceso con ext/mysqli

mysqli es una extensión de PHP que permite acceder a ciertas funciones disponibles a partir de MySQL 4.1 que no se pueden emplear con la extensión tradicional **mysql**. ext/mysqli proporciona una mayor velocidad, una mayor seguridad, una interfaz procedural u orientado a objetos y el empleo del nuevo protocolo binario de MySQL 4.1 que permite ciertas funciones como la ejecución de sentencias preparadas. Si se emplea la interfaz orientada a objetos, el programador puede desarrollar sus propias

clases de acceso a la base de datos que hereden de las proporcionadas por la extensión, lo que permite un mayor control y adecuación a las necesidades de cada uno.

El cambio de ext/mysql a ext/mysqli es inmediato, ya que en la interfaz procedural existen las mismas funciones pero con el prefijo mysqli. Las principales funciones que se emplean en la interfaz procedural son:

- `mysqli_connect(servidorBD, usuario, contraseña, nombreBD)`: abre una conexión con un servidor de bases de datos de MySQL, devuelve un identificador que se emplea en algunas de las siguientes funciones o FALSE en caso de error. Presenta una diferencia respecto a la misma función de ext/mysql: permite indicar la base de datos que se desea seleccionar.
- `mysqli_connect_errno()`: devuelve el código del último error de conexión. Esta función no existe en ext/mysql.
- `mysqli_connect_error()`: devuelve una descripción del último error de conexión. Esta función no existe en ext/mysql.
- `mysqli_close(identificador)`: cierra una conexión con un servidor de MySQL, devuelve TRUE en caso de éxito y FALSE en caso contrario.
- `mysqli_ping(identificador)`: verifica que la conexión con el servidor de bases de datos funciona, devuelve TRUE en caso de éxito y FALSE en caso contrario.
- `mysqli_select_db(identificador, nombreBD)`: selecciona una base de datos, devuelve TRUE en caso de éxito y FALSE en caso contrario. **¡Cuidado!**, el orden de los parámetros es distinto, en ext/mysql el orden es (nombreBD, identificador).
- `mysqli_query(identificador, sentencia)`: ejecuta una sentencia SQL y devuelve un resultado (SELECT, SHOW, EXPLAIN o DESCRIBE, ...) o TRUE (INSERT, UPDATE, DELETE, ...) si todo es correcto, o FALSE en caso contrario. **¡Cuidado!**, el orden de los parámetros es distinto, en ext/mysql el orden es (sentencia, identificador).
- `mysqli_fetch_array(resultado)`: recorre un resultado, devuelve un array que representa una fila (registro) o FALSE en caso de error (por ejemplo, llegar al final del resultado); al array se puede acceder de forma numérica (posición de la columna) o asociativa (nombre de la columna).
- `mysqli_fetch_assoc(resultado)` y `mysqli_fetch_row(resultado)`: ambas funciones son similares a la anterior `mysqli_fetch_array(resultado)`, pero sólo permiten el acceso como array asociativo o con índices numéricos respectivamente.
- `mysqli_fetch_object(resultado)`: recorre un resultado, devuelve un objeto que representa la fila actual o NULL si no hay más filas en el conjunto de resultados. en caso de error
- `mysqli_affected_rows(identificador)`: devuelve el número de filas (tuplas) afectadas por la última operación si fue del tipo INSERT, UPDATE, etc., que no devuelven un resultado.
- `mysqli_num_rows(resultado)`: devuelve el número de filas (tuplas) afectadas por la última operación si fue del tipo SELECT.
- `mysqli_free_result(resultado)`: libera la memoria empleada por un resultado. **¡Cuidado!**, mientras que en ext/mysql no se indica que sea necesario llamar a `mysqli_free_result(resultado)`, en ext/mysqli se dice que “siempre se debe liberar el resultado con `mysqli_free_result(resultado)`, cuando el objeto del resultado ya no es necesario”.
- `mysqli_errno(identificador)`: devuelve el código del último error.
- `mysqli_error(identificador)`: devuelve una descripción del último error.

El siguiente ejemplo muestra como se realiza un acceso a una base de datos mediante la interfaz procedural; se emplea el operador de control de errores “@” para evitar que se muestren en la página posibles mensajes de error:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de SELECT y mysqli procedural</title>
</head>
<body>
<?php
```

```
// Conecta con el servidor de MySQL
$link = @mysqli_connect(
    'localhost',    // El servidor
    'wwwdata',     // El usuario
    '',            // La contraseña
    'biblioteca'); // La base de datos

if(!$link) {
    echo '<p>Error al conectar con la base de datos: ' . mysqli_connect_error();
    echo '</p>';
    exit;
}

// Ejecuta una sentencia SQL
$sentencia = 'SELECT * FROM libros';
if(!($resultado = @mysqli_query($link, $sentencia))) {
    echo '<p>Error al ejecutar la sentencia <b>$sentencia</b>: ' . mysqli_error($link);
    echo '</p>';
    exit;
}

echo '<table><tr>';
echo '<th>IdLibro</th><th>Título</th><th>Resumen</th>';
echo '<th>Autor</th><th>Categoría</th><th>Editorial</th><th>Anyo</th>';
echo '</tr>';
// Recorre el resultado y lo muestra en forma de tabla HTML
while($fila = mysqli_fetch_assoc($resultado)) {
    echo '<tr>';
    echo '<td>' . $fila['IdLibro'] . '</td>';
    echo '<td>' . $fila['Titulo'] . '</td>';
    echo '<td>' . $fila['Resumen'] . '</td>';
    echo '<td>' . $fila['Autor'] . '</td>';
    echo '<td>' . $fila['Categoría'] . '</td>';
    echo '<td>' . $fila['Editorial'] . '</td>';
    echo '<td>' . $fila['Anyo'] . '</td>';
    echo '</tr>';
}
echo '</table>';

// Libera la memoria ocupada por el resultado
mysqli_free_result($resultado);
// Cierra la conexión
mysqli_close($link);
?>
</body>
</html>
```

A continuación se muestra el ejemplo anterior pero programado con la interfaz orientada a objetos de ext/mysqli. En este caso, se emplea `new mysqli()` para crear la conexión y lo que antes eran llamadas a funciones se convierten en llamadas a métodos del objeto creado por `new mysqli()`; para acceder a los métodos y propiedades de un objeto se emplea el operador “->”:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de SELECT y mysqli orientado a objetos</title>
</head>
<body>
<?php
// Conecta con el servidor de MySQL
$mysqli = @new mysqli(
    'localhost',    // El servidor
    'wwwdata',     // El usuario
    '',            // La contraseña
    'biblioteca'); // La base de datos
```

```

if($mysqli->connect_errno) {
    echo '<p>Error al conectar con la base de datos: ' . $mysqli->connect_error;
    echo '</p>';
    exit;
}

// Ejecuta una sentencia SQL
$sentencia = 'SELECT * FROM libros';
if(!($resultado = $mysqli->query($sentencia))) {
    echo "<p>Error al ejecutar la sentencia <b>$sentencia</b>: " . $mysqli->error;
    echo '</p>';
    exit;
}

echo '<table><tr>';
echo '<th>IdLibro</th><th>Título</th><th>Resumen</th>';
echo '<th>Autor</th><th>Categoría</th><th>Editorial</th><th>Anyo</th>';
echo '</tr>';
// Recorre el resultado y lo muestra en forma de tabla HTML
while($fila = $resultado->fetch_assoc()) {
    echo '<tr>';
    echo '<td>' . $fila['IdLibro'] . '</td>';
    echo '<td>' . $fila['Titulo'] . '</td>';
    echo '<td>' . $fila['Resumen'] . '</td>';
    echo '<td>' . $fila['Autor'] . '</td>';
    echo '<td>' . $fila['Categoria'] . '</td>';
    echo '<td>' . $fila['Editorial'] . '</td>';
    echo '<td>' . $fila['Anyo'] . '</td>';
    echo '</tr>';
}
echo '</table>';

// Libera la memoria ocupada por el resultado
$resultado->close();
// Cierra la conexión
$mysqli->close();
?>
</body>
</html>

```

Y el siguiente ejemplo es similar al anterior, pero muestra cómo recorrer el resultado mediante objetos en vez de arrays y con la interfaz orientada a objetos de ext/mysql:

```

<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8" />
<title>Prueba de SELECT y mysql orientado a objetos</title>
</head>
<body>
<?php
// Conecta con el servidor de MySQL
$mysqli = @new mysqli(
    'localhost', // El servidor
    'wwwdata', // El usuario
    '', // La contraseña
    'biblioteca'); // La base de datos

if($mysqli->connect_errno) {
    echo '<p>Error al conectar con la base de datos: ' . $mysqli->connect_error;
    echo '</p>';
    exit;
}

// Ejecuta una sentencia SQL
$sentencia = 'SELECT * FROM libros';

```

```

if(!$resultado = $mysqli->query($sentencia)) {
    echo "<p>Error al ejecutar la sentencia <b>$sentencia</b>: " . $mysqli->error;
    echo '</p>';
    exit;
}

echo '<table><tr>';
echo '<th>IdLibro</th><th>Título</th><th>Resumen</th>';
echo '<th>Autor</th><th>Categoría</th><th>Editorial</th><th>Anyo</th>';
echo '</tr>';
// Recorre el resultado y lo muestra en forma de tabla HTML
while($fila = $resultado->fetch_object()) {
    echo "<tr>";
    echo "<td>$fila->IdLibro</td>";
    echo "<td>$fila->Título</td>";
    echo "<td>$fila->Resumen</td>";
    echo "<td>$fila->Autor</td>";
    echo "<td>$fila->Categoría</td>";
    echo "<td>$fila->Editorial</td>";
    echo "<td>$fila->Anyo</td>";
    echo "</tr>";
}
echo '</table>';

// Libera la memoria ocupada por el resultado
$resultado->close();
// Cierra la conexión
$mysqli->close();
?>
</body>
</html>

```

Fíjate que en el ejemplo anterior se ha cambiado la forma de mostrar el contenido: la variable `$fila` que representa al objeto que se devuelve de `$resultado` se ha escrito directamente dentro de una cadena con comillas dobles y la variable se ha expandido a su valor [\(29\)](#).

`ext/mysqli` añade nuevas características que no existen en `ext/mysql`. Por ejemplo, las sentencias preparadas (*prepared statements*) permiten escribir un código más claro y más robusto, ya que evitan la inyección de SQL. Con la interfaz orientada a objetos de `ext/mysqli` y el mecanismo de la herencia se pueden desarrollar clases propias de acceso a bases de datos.

La base de datos y la conexión a la base de datos también emplean un juego de caracteres [\(30\)](#). Si todos los elementos de una aplicación web no emplean el mismo juego de caracteres, el desastre está asegurado. En “Conjunto de caracteres” [\(31\)](#) y “`mysqli::set_charset`” [\(32\)](#) se explica cómo lo debes hacer.

4.4. Ficheros de configuración

Los datos de conexión a la base de datos (nombre de usuario, contraseña, dirección del servidor, nombre de la base de datos, etc.) se utilizan en muchas páginas de una aplicación web, cada vez que es necesario establecer una conexión con la base de datos. Estos datos se deben almacenar en un fichero independiente que se debe incluir en aquellas páginas donde se necesite. De esta forma, cuando se tenga que cambiar alguno de los datos de configuración, sólo se tendrá que hacer en un fichero.

Una forma de hacerlo es mediante la inclusión de ficheros con `include()` o `require()`. En estos ficheros se pueden definir valores constantes con `define()`:

```

define("dbServer", "localhost");
define("dbUser", "wwwdata");
define("dbPassword", "");
define("dbDatabase", "biblioteca");

```

Sin embargo, existe una forma mejor y más sencilla, los ficheros de configuración de tipo `.ini`. Un fichero `.ini` se compone de diferentes secciones que a su vez poseen opciones que son parejas propiedad = valor. Por ejemplo:

```
; Este es un ejemplo de archivo de configuración
; Los comentarios empiezan con ';', como en php.ini

; En un fichero .ini se pueden crear diferentes secciones
[DB]
Server = "localhost"
User = "wwwdata"
; También vale Password = ""
Password = ""
Database = "biblioteca"

; Esto es otra sección que está vacía
[Otra_sección]
```

Para leer un fichero de configuración se emplea la función `parse_ini_file()`, que devuelve un array con todas las opciones definidas en el fichero. El primer parámetro es la ruta del fichero de configuración; el segundo parámetro permite indicar si se quiere obtener una matriz multidimensional con los nombres de las secciones.

5. Recomendaciones

Recuerda que las páginas que contengan código PHP tienen que tener la extensión `.php`. Si modificas alguna página web que ya tengas hecha de prácticas anteriores para añadirle código PHP, tendrás que cambiarle la extensión y corregir todos los enlaces que apunten a esa página.

Recuerda que para que el código PHP se ejecute, la página web tiene que pasar por el servidor web para que éste se la pase al intérprete de PHP. Para ello, tienes que cargar la página a través del servidor web: la URL de conexión tiene que tener la forma `http://localhost`. Si en la barra de direcciones del navegador ves escrito algo del estilo `file:///D:/...`, el código PHP no se ejecutará.

El manual de PHP te lo puedes descargar en diferentes formatos de su sitio web [\(33\)](#) para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente. También puedes acceder a través de Internet a la ayuda de cualquier función de PHP escribiendo el nombre de la función a continuación de la URL `http://php.net/`. Por ejemplo, `http://php.net/header` muestra la ayuda de la función `header()`.

La página web “Build Your Own Database Driven Website Using PHP & MySQL” [\(34\)](#) explica en cuatro partes (*Introduction*, *The Database Server*, *PHP server-side scripting language* y *Displaying Information on a Web page*) cómo construir un sitio web con información procedente de una base de datos en MySQL.

Recuerda que XAMPP es una plataforma de desarrollo y no está orientada a ser usada en producción, ya que no está configurada para obtener un máximo rendimiento y puede tener problemas de seguridad. Por ejemplo, el usuario “root” de la base de datos no tiene contraseña, eso es un tremendo error en un servidor puesto en producción.

Recuerda: nunca te conectes a una base de datos con el usuario “root” desde una página web. Emplea un usuario específico que tenga el mínimo posible de permisos. Por ejemplo, si en una página web sólo necesitas mostrar el contenido de las tablas de una base de datos, pero no necesitas actualizar o insertar nuevos datos, utiliza un usuario que sólo tenga el permiso `SELECT` sobre esa base de datos. De este modo reducirás los posibles problemas de seguridad.

Fíjate que en varias páginas se pide *en la lista desplegable para seleccionar el país, mostrar los países a partir de la tabla **Países** de la base de datos*. Cuando una parte de una página web se repita en varias páginas, lo mejor es aislar el código común en un fichero independiente e incluirlo en aquellos puntos donde haga falta.

Aunque no se pide implementarlo en esta práctica, cuando se muestra un listado a partir de una consulta a una base de datos y no existe un límite para el número de resultados devueltos es conveniente mostrar el listado paginado para evitar que se devuelva una página web enorme que aumente su tiempo de transmisión y dificulte su lectura. ¿Te atreves a hacerlo?

El manual de MySQL te lo puedes descargar en diferentes formatos de su sitio web [\(35\)](#) para tenerlo siempre a mano y poder hacer las búsquedas de información rápidamente. Cuando lo descargues, elige la versión correspondiente a tu servidor de MySQL.

Si quieres guardar una copia de seguridad de una base de datos, puedes emplear la opción Exportar de phpMyAdmin que se muestra en la Figura [13](#).

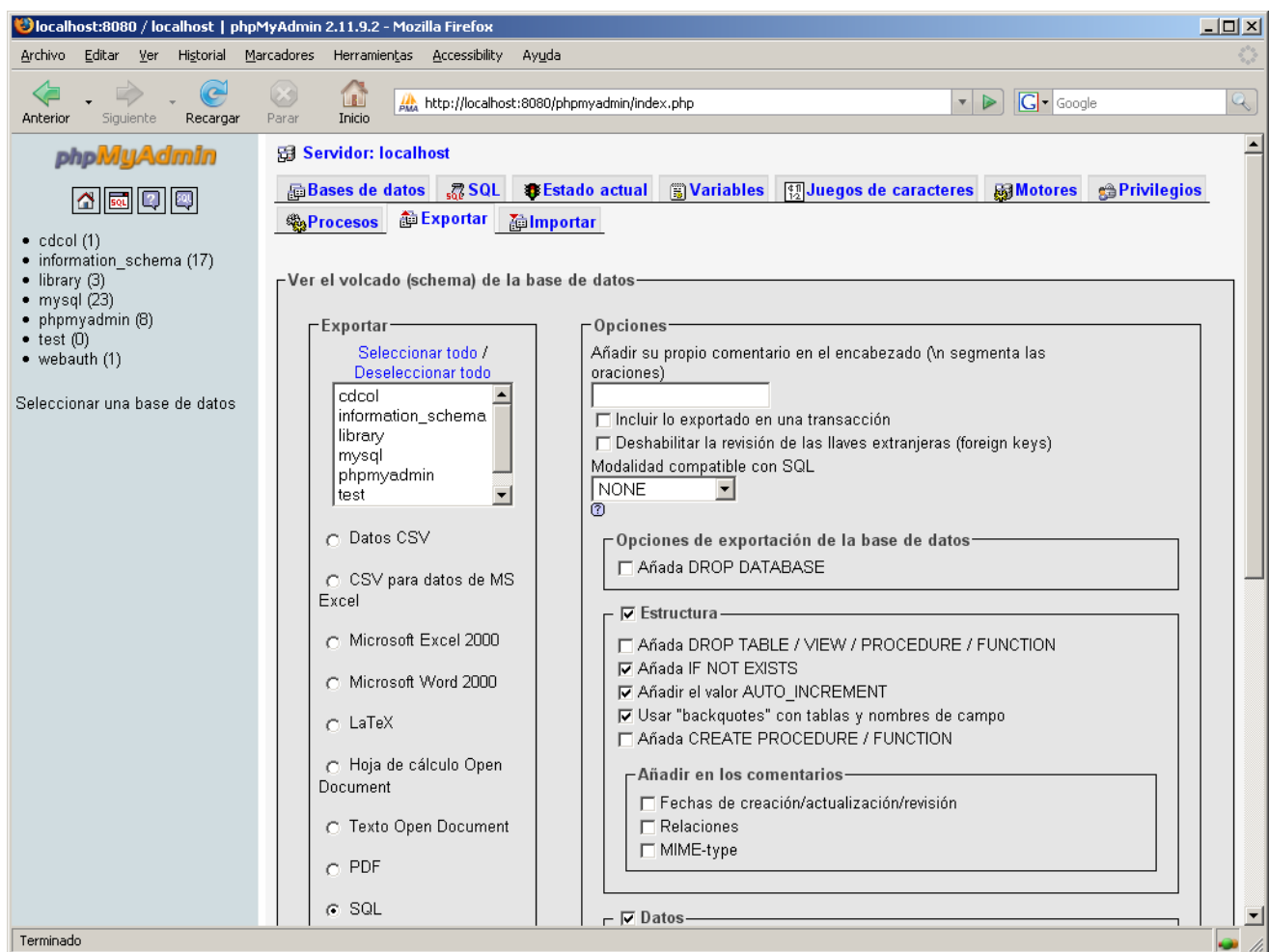


Figura 13: Exportar desde phpMyAdmin

Además de los datos de conexión a la base de datos, puedes tener otros datos que se utilicen en varias páginas de tu aplicación web. Almacena todos estos datos en el fichero de configuración para que su mantenimiento sea más sencillo.

Almacenar las contraseñas de los usuarios sin encriptar es una práctica muy mala, ya que supone un agujero de seguridad enorme. Utilizar técnicas simples como base 64 o MD5 también son malas prácticas.

¿Cómo se debe hacer? Investiga un poco e impleméntalo en la práctica; si lo haces, puedes cambiar la longitud del campo Clave para permitir que se almacene la contraseña encriptada en su totalidad.

- (1) <http://dev.mysql.com/>
- (2) <http://www.addedbytes.com/cheat-sheets/mysql-cheat-sheet/>
- (3) <http://www.nparikh.org/unix/mysql.php>
- (4) <http://www.digilife.be/quickreferences/ORC/MySQL-4.02a.pdf>
- (5) <http://www.nparikh.org/unix/mysql.php>
- (6) <http://es.wikipedia.org/wiki/SQL>
- (7) <http://www.w3schools.com>
- (8) <http://www.w3schools.com/sql/default.asp>
- (9) <http://www.sqlcourse.com/>
- (10) http://www.3gwt.net/demo/SQL_redux.html
- (11) <http://www.phpmyadmin.net/>
- (12) <http://phpminadmin.sourceforge.net/>
- (13) <http://dev.mysql.com/downloads/gui-tools/>
- (14) <http://dev.mysql.com/downloads/workbench/>
- (15) <http://www.php.net/manual/es/book.mysql.php>
- (16) <http://www.php.net/manual/es/book.mysqli.php>
- (17) http://www.w3schools.com/php/php_mysql_intro.asp
- (18) <http://devzone.zend.com/239/ext-mysqli-part-i-overview-and-prepared-statements/>
- (19) <http://devzone.zend.com/255/using-ext-mysqli-part-ii-extending-mysqli/>
- (20) http://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL
- (21) <http://www.php-es.com/security/database/sql-injection.html>
- (22) <http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/>
- (23) *New XAMPP with MariaDB:* https://www.apachefriends.org/blog/new_xampp_20151019.html
- (24) https://en.wikipedia.org/wiki/Drop-in_replacement
- (25) *MariaDB versus MySQL - Compatibility:* <https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-compatibility/>
- (26) <http://www.php.net/manual/es/refs.database.php>
- (27) <http://www.php.net/manual/es/mysqlinfo.api.choosing.php>
- (28) <https://es.wikipedia.org/wiki/Modelo%2Fvista%2Fcontrolador>
- (29) <http://php.net/manual/es/language.types.string.php#language.types.string.parsing>
- (30) *Revisa el vídeo “HTML: juego de caracteres”:* https://youtu.be/_MwDNB3j0x0
- (31) <http://php.net/manual/es/mysqlinfo.concepts.charset.php>
- (32) <http://php.net/manual/es/mysqli.set-charset.php>
- (33) <http://www.php.net/download-docs.php>
- (34) <http://www.databasejournal.com/features/mysql/article.php/1383591/-Build-Your-Own-Database-Driven-Website-Using-PHP--MySQL-Pt-1.htm>
- (35) <http://dev.mysql.com/doc/>