

TEMA 3.6

ACCESO A BBDD MYSQL

1. [BASES DE DATOS EN LA WEB](#)
2. [MYSQL](#)
3. [INSTALACIÓN Y CONFIGURACIÓN DE MYSQL](#)
4. [HERRAMIENTAS DE ADMINISTRACIÓN: PHPMYADMIN](#)
5. [LENGUAJE SQL](#)
6. [FUNCIONES DE PHP PARA ACCESO A BBDD MYSQL](#)

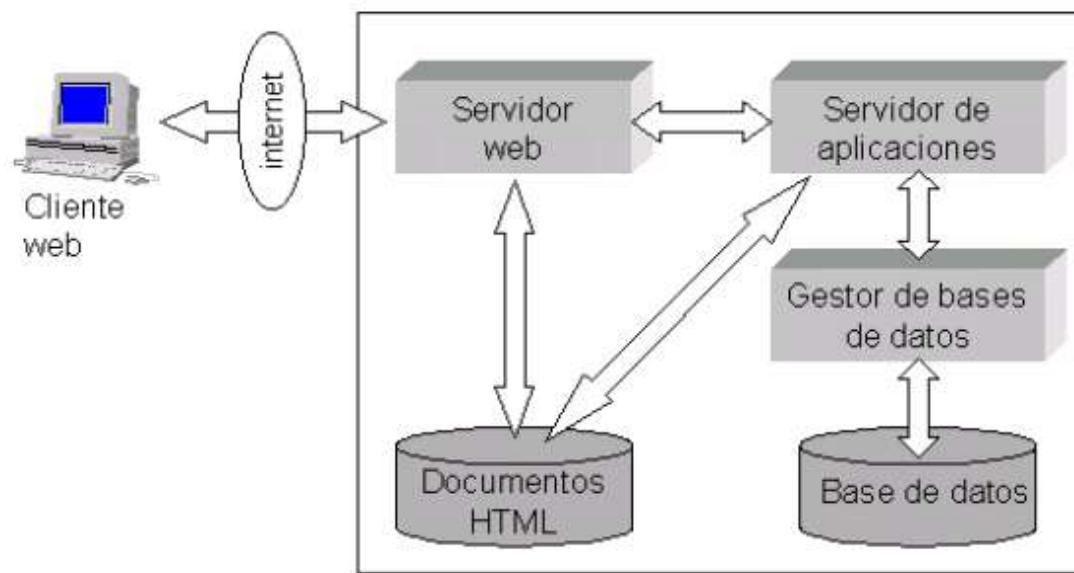
1. BASES DE DATOS EN LA WEB

Los bases de datos permiten almacenar de una forma estructurada y eficiente toda la información de un sitio web .

Ventajas

- Proporcionar información actualizada
- Facilitar la realización de vueltas.
- Disminuir los costes de mantenimiento.
- Implementar sistemas de control de acceso.
- Almacenar preferencias de los usuarios

Esquema básico de un sitio web soportado por bases de datos :



Los servidores de bbdd pueden estar físicamente en el mismo servidor web o en un otro con el cual es comunica a través de TCP /IP.

Un *SERVIDOR DE BBDD RELACIONALES* es una aplicación que se encarga de gestionar el acceso a los datos.

Los programas que necesitan recuperar o almacenar información no acceden directamente a los archivos de la bbdd, sino que se comunican con el SGBD y delegan en él este trabajo.

2. INTRODUCCIÓN A MYSQL

Manuales Mysql en:

<http://dev.mysql.com/doc/>

<https://dev.mysql.com/doc/refman/8.0/en/>

MySQL es un gestor de base de datos sencillo de usar y increíblemente rápido. También es uno de los motores de base de datos más usados en Internet, la principal razón de esto es que es gratuito para aplicaciones no comerciales.

Los **características principales** de MySQL son:

- ❑ **Es un gestor de base de datos** . Una base de datos es un conjunto de datos y un gestor de base de datos es una aplicación capaz de manejar este conjunto de datos de manera eficiente y cómoda.
- ❑ **Es una base de datos relacional**. Conjunto de datos que están almacenados en tablas entre los cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para usar y gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.
- ❑ **Es Open Source**. El código fuente de MySQL es puede descargar y está accesible a cualquier , de otra banda, usa la licencia GPL para aplicaciones no comerciales.
- ❑ **Es una base de datos muy rápida**, segura y fácil de usar . Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por esto es una de las bases de datos más usados en Internet.
- ❑ **Existe una gran cantidad de software que lo usa** .

Los **tipos de datos** admitidos por MySQL son:

➤ **Numéricos:**

- *TinyInt*: N.º entero con o sin signo. Con signo, el rango de valores es encuentra entre -128 y 127. Sin signo, el rango de valores válidos es encuentra entre 0 a 255
- *Bit*: N.º entero el valor del cual puede ser 0 o 1
- *Bool*: N.º entero el valor del cual puede ser 0 o 1
- *SmallInt*: N.º entero con o sin signo. Con signo, el rango de valores es encuentra entre -32768 y 32767. Sin signo el rango de valores es encuentra entre 0 y 65535
- *MediumInt*: N.º entero con o sin signo. Con signo el rango de valores válidos es encuentra entre 8388608 y 8388607. Sin signo el rango de valores válidos es encuentra entre 0 y 6777215.
- *Integer, Int*: N.º entero con o sin signo. Con signo, el rango de valores válidos es encuentra entre -2147483648 y 2147483647
- *BigInt*: N.º entero con o sin signo. Con signo, el rango de valores válidos es encuentra entre -9223372036854775808 y 9223372036854775807.
- *Float*: N.º pequeño en coma flotante de precisión simple. El rango de valores válidos es encuentra entre -3.402823466E+38 y 1.175494351E-38
- *xReal*: N.º en coma flotante de precisión doble. El rango de valores válidos es encuentra entre 1.7976931348623157E +308 y 2.2250738585072014E -308,
- *Double*: N.º en coma flotante de precisión doble. El rango de valores válidos es encuentra entre 1.7976931348623157E +308 y -2.2250738585072014E -308,
- *Decimal, Debo, Numeric*: N.º en coma flotante desempaquetado. En este caso el número se almacena como una cadena.

➤ **Fecha y hora:**

- *Time*: almacena una hora. El rango de valores válidos para la misma es 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“
- *DateTime*: combinación de fecha y hora. El rango de valores válidos es encuentra entre los 0 horas, 0 minutos y 0 segundos del 1 de enero de 1 001 y los 23 horas, 59 minutos y 59 segundos del 31 de diciembre de 9999. El formato de almacenamiento es de año mas día horas:minutos:según
- *TimeStamp*: Combinación de fecha y hora. El rango de valores válidos de la misma es encuentra entre el 1 de enero de 1 970 y el 31 de diciembre de 2037.

El formato de almacenamiento depende del tamaño del campo :

- `TIMESTAMP(14)`: AñoMesDíaHoraMinutosSegundo (aaaammddhhmmss)
- `TIMESTAMP(12)`: AñoMesDíaHoraMinutosSegundo (aammddhhmmss)
- `TIMESTAMP(8)`: AñoMesDía (aaaammdd)
- `TIMESTAMP(6)`: AñoMesDía (aammdd)
- `TIMESTAMP(4)`: AñoMes (aamm)
- `TIMESTAMP(2)`: Año (aa)

➤ Cadena

- *Char(n)*: Almacena una cadena de longitud fija el nombre de caracteres de la cual puede estar comprendido entre 0 y 255
- *VarChar(n)*: Almacena una cadena de longitud variable el nombre de caracteres de la cual puede estar comprendido entre 0 y 255
- *Texto*: se ordena sin tener en cuenta mayúsculas y minúsculas
 - *TinyText*: columna con una longitud máxima de 255 caracteres.
 - *Texto*: texto con una longitud máxima de 65535 caracteres
 - *MediumText*: texto con una longitud máxima de 16.777.215 caracteres.
 - *LongText*: texto con una longitud máxima de 4.294.967.295 caracteres
- *BLOB* (Binary Large Object):
 - *TinyBlob*: columna con una longitud máxima de 255 caracteres
 - *Blob*: columna con una longitud máxima de 65535 caracteres
 - *MediumBlob*: texto con una longitud máxima de 16.777.215 caracteres
 - *LongBlob*: texto con una longitud máxima de 4.294.967.295 caracteres.
- *Enum*: campo que puede obtener un único valor de una lista de posibles valores donde se especifica . Este tipo acepta hasta 65535 valores diferentes.
- *Siete*: Camp que puede contener ninguno, uno, o bien varios valores de una lista que puede contener un máximo de 64 valores.

Nota: Tiene que elegir-se adecuadamente el tipo y el tamaño de cada campo

3. INSTALACIÓN Y CONFIGURACIÓN DE MYSQL

Igual que sucedía con PHP, tenemos dos maneras de instalar MySQL, una manual y una otra automática.

La forma automática es la que se instala por defecto con una herramienta tipo *XAMPP*, *WAMPP* o *AppServer* y es la que utilizaremos en clase.

Descarga de XAMPP desde:

<https://www.apachefriends.org/es/index.html>

Con la instalación de XAMPP se incluye el de la herramienta **PHPMYADMIN** que es verá en el siguiente punto.

La otra alternativa es la de instalar **MySQL** manualmente desde:

<https://www.mysql.com>

Vamos a mostrar el proceso de instalación de MySQL:

INSTALACIÓN DE MYSQL

Ver archivo adjunto del tema llamado: *Instalar Mysql y Configura PhpMyAdmin.doc*

CREACIÓN DE UNA BASE DE DATOS EJEMPLO (BD CLIENTES). EJECUCIÓN DE SCRIPTS

Realmente el que es hará será crear un esquema de base de datos.

Para esto seguiremos los siguientes pasos:

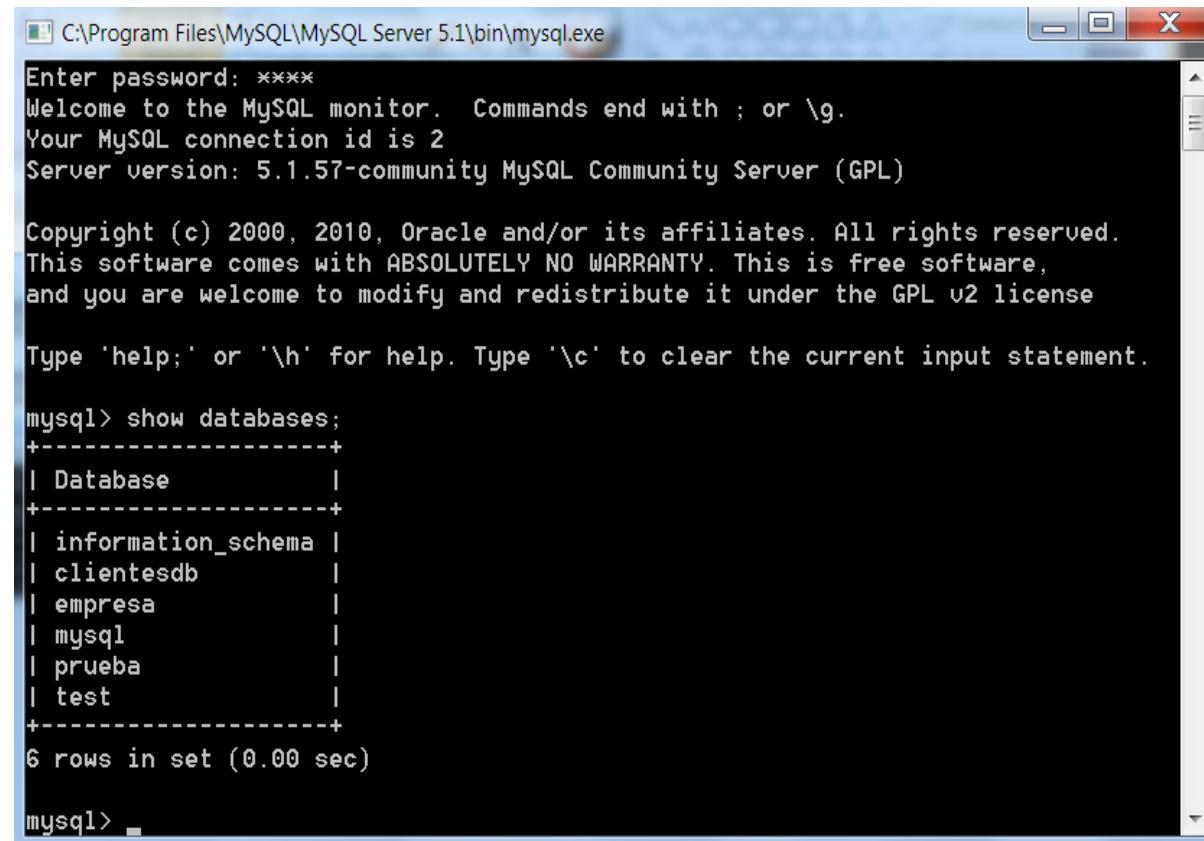
- Acceder al símbolo del sistema (mediante comando **cmd**)
- Ejecutar el monitor de mysql, para esto teclearemos

Mysql -o root -p

- Nos pedirá el password, que lo habíamos establecido antes en **root**
- Creamos un esquema de base de datos, mediante **create database ClientesDB_DWES;**

Nota: Recordar que antes de ejecutar el script de creación de tablas, habrá que crear con la sentencia anterior la BD a la cual queremos que pertenezcan las tablas generadas en dicho script

- Es puede pedir que nos muestre todos los esquemas de bases de datos disponibles mediante **show databases;**



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.57-community MySQL Community Server (GPL)

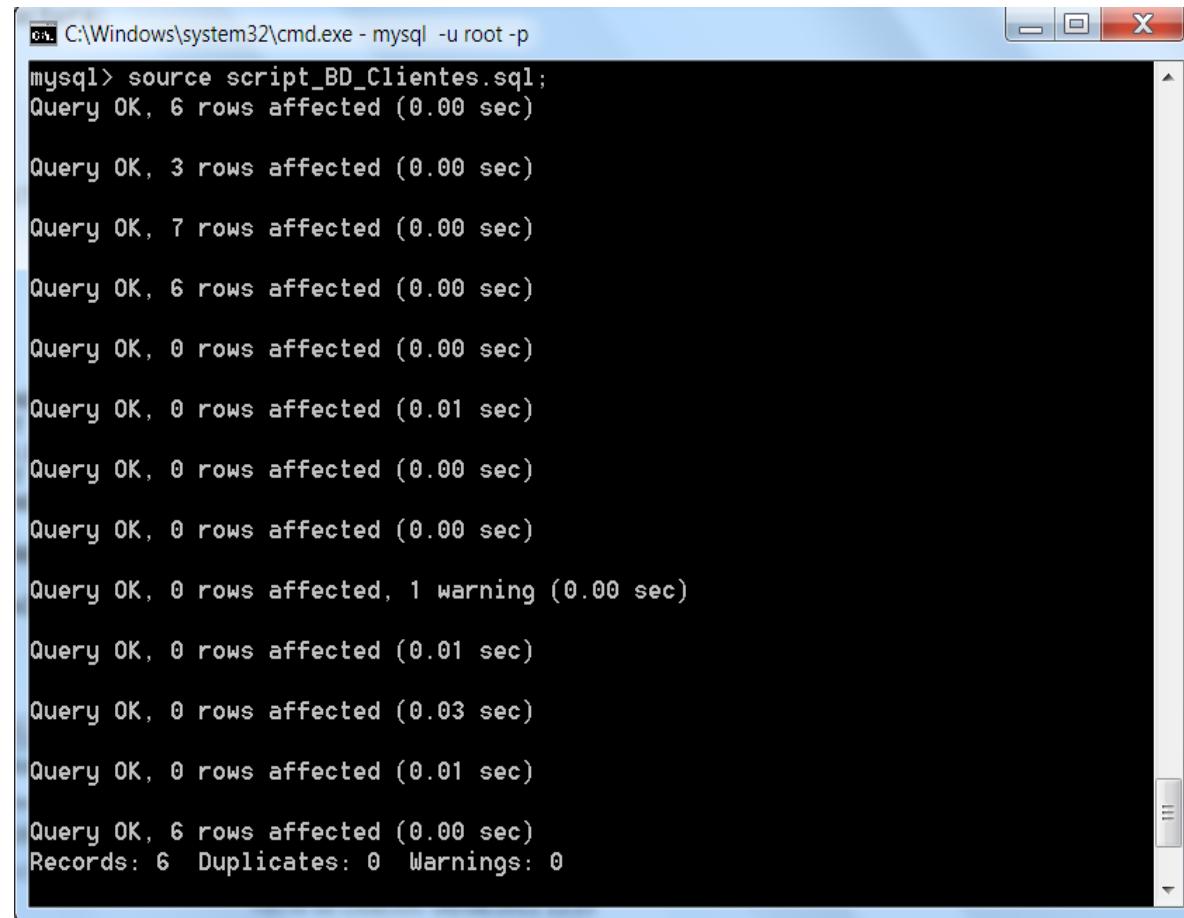
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+----+-----+
| information_schema |
| clientesdb         |
| empresa            |
| mysql              |
| prueba             |
| test               |
+----+-----+
6 rows in set (0.00 sec)

mysql>
```

- Nos situamos en la base de datos creada mediante : **uso ClientesDB_IAW;**
- Desde este esquema de base de datos, podemos ejecutar un script que contenga sql de creación de tablas y inserción de datos intermediendo: **source c:\ script_BD_Clientes_DWES.sql**



```
mysql> source script_BD_Clientes.sql;
Query OK, 6 rows affected (0.00 sec)

Query OK, 3 rows affected (0.00 sec)

Query OK, 7 rows affected (0.00 sec)

Query OK, 6 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

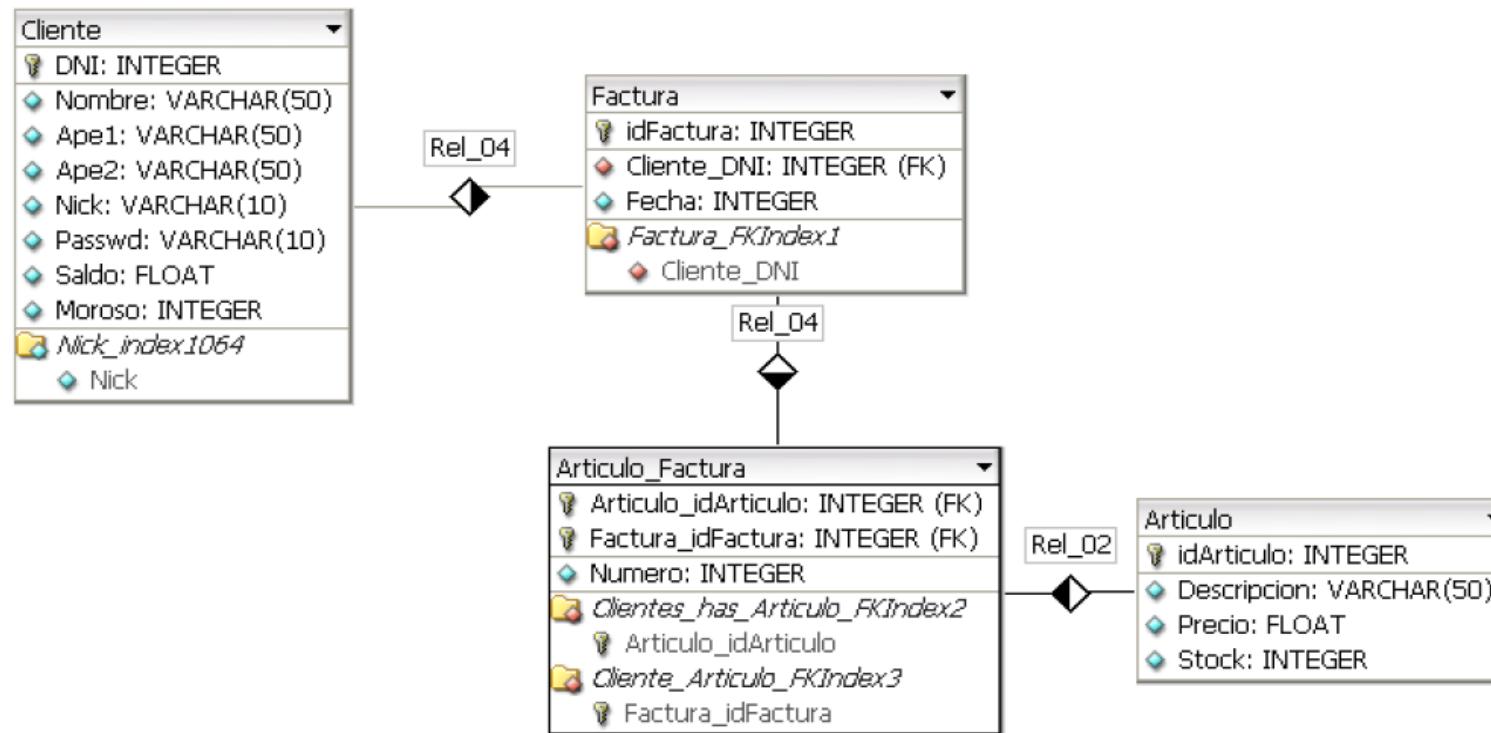
Query OK, 0 rows affected (0.03 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0   Warnings: 0
```

- Para esto el script tendrá que estar situado en la ruta indicada.

- Después de la ejecución de dicho script, los tablas que se han generado son los que se muestran en la siguiente figura:



Recordatorio:

En este entorno podremos crear todas las tablas de la forma que ya conocemos

- ***Uso nombre_bd;***
- ***Create database nombre_bd;***
- ***Create table nombre_tabla....***
- ***Insert into nombre_tabla***
- ***Show tables;***
- ***El resto de sintaxis lo tenemos disponible en la herramienta PHPMYADMIN que acabamos de instalar.***
- Salimos del monitor de mysql mediante ***quit o exit***

4. HERRAMIENTAS De ADMINISTRACIÓN: PHPMYADMIN

Ver archivo anexo del tema ***PhpMyAdmin.pdf***

Para administrar la nuestra base de datos *MySQL*, disponemos de herramientas que nos facilitan bastante el trabajo por entorno gráfico, que por descontado es más amigable que trabajar en línea de comandos.

Nosotros utilizaremos ***phpMyAdmin***, que voz preinstal·lada con los paquetes **XAMPP**, WAMP y AppServer.

En caso de no ser así, la podemos descargar desde :

<https://www.phpmyadmin.net/>

PhpMyAdmin es una herramienta para la administración del servidor de bases de datos MySQL.

Características de PhpMyAdmin :

- Dispone de una interfaz gráfica y es de libre distribución
- Permite realizar todo tipo de operaciones sobre bases de datos :
 - crear , borrar y modificar tablas
 - consultar , insertar, modificar y eliminar datos
 - definir usuarios y asignar permisos
 - realizar copias de seguridad
 - etc
- Está escrita en php y se ejecuta desde del navegador.
- Si está instal·lada en la carpeta *phpmyadmin*, se ejecuta escribiendo en la barra de direcciones del navegador la url:
<http://localhost/phpmyadmin/>
- Puede administrar bases de datos locales y remotes

El aspecto que presenta es el siguiente:

The screenshot shows the phpMyAdmin 3.5.2 interface running in Mozilla Firefox. The title bar indicates the URL is `localhost / localhost | phpMyAdmin 3.5.2 - Mozilla Firefox`. The main menu bar includes Archivo, Editar, Ver, Historial, Marcadores, Herramientas, and Ayuda. Below the menu is a toolbar with icons for PMA, localhost, and other functions. The address bar shows `localhost/phpmyadmin/`. The browser's status bar indicates `Más visitados`, `Comenzar a usar Firefox`, `Últimas noticias`, and `saxos`. The search bar has `Google` selected.

The left sidebar features the **phpMyAdmin** logo and navigation links for Home, MySQL, Databases, SQL, Current status, Users, Export, Import, Configuration, Synchronize, and More.

The main content area displays the **Bases de datos** (Databases) page. On the left, there is an **Error** panel containing the following information:

- consulta SQL:** `SELECT `tables`
FROM
`phpmyadmin`.`pma_recen
WHERE `username` =
'root'`
- MySQL ha dicho:** #1146 - Table 'phpmyadmin.pma_recent' doesn't exist
- La conexión para controluser, como está definida en su configuración, fracasó.**

The main database list shows the following databases:

Base de datos	Opciones
clientesdb	Comprobar los privilegios
empresa	Comprobar los privilegios
information_schema	Comprobar los privilegios
mysql	Comprobar los privilegios
prueba	Comprobar los privilegios
prueba1_php	Comprobar los privilegios
test	Comprobar los privilegios

Total: 7

Below the database list, there is a note: "Marcar todos / Desmarcar todos Para los elementos que están marcados: Eliminar".

At the bottom, there is a link to "Activar las estadísticas".

Fichero de configuración de phpMyAdmin:.

C:\xampp\phpMyAdmin\config.inc.php

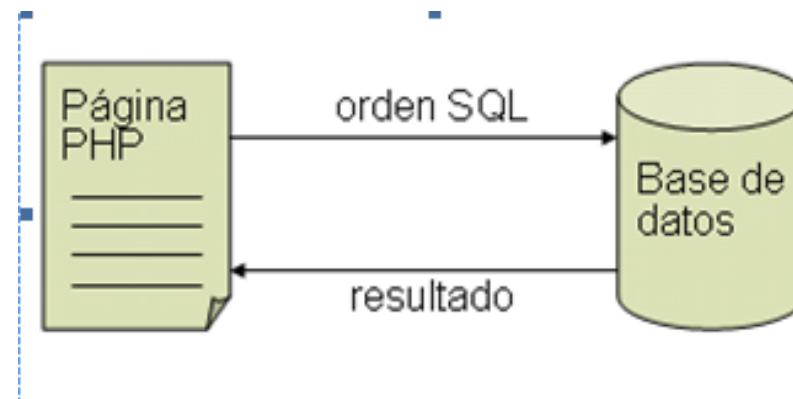
En este archivo se encuentran definidas las variables de acceso a phpmyadmin:

- usuario root,
- contraseña
- control de sesiones por cookies
- etc

5. LENGUAJE SQL

SQL (*Structured Query Language*) es el lenguaje que se utiliza para comunicarse con la base de datos.

El procedimiento de comunicación con la base de datos es puede ver en la siguiente figura:



Los instrucciones más habituales son **SELECT, INSERT, UPDATE, DELETE**

Veamos la suya sintaxis básica y algunos ejemplos de uso , para esto utilizaremos una tabla **NOTICIAS** con cinco campos: un identificador único de la noticia, el título de la noticia, el texto de la noticia, la categoría de la noticia y la fecha de publicación de la noticia

noticias
id
título
texto
categoria
fecha

SELECT

Sintaxis:

```
SELECT expresión  
FROM tabla  
[WHERE condición]  
[ORDER BY {unsigned_integer |  
          col_name |  
          formula}  
[ASC | DESC] ,...]  
[LÍMITE [offset,]  
row_count |  
row_count OFFSET offset]
```

Ejemplo:

```
SELECT  
from NOTICIAS  
WHERE fecha=CURDATE() LIMIT 10  
ORDER BY fecha DESC
```

Obtiene las noticias del día con un valor máximo de 10, ordenadas de la más reciente a la más antigua

INSERT

Sintaxis:

```
INSERT [INTO] nombre_tabla [(nombre_columna,...)]  
VALUES ((expresión | DEFAULT),...), (...),...
```

```
INSERT [INTO] nombre_tabla  
SIETE nombre_columna=(expresión | DEFAULT), ...
```

Ejemplo:

```
INSERT INTO NOTICIAS (id, título, texto, categoría, fecha)  
VALUES (37, "Nueva promoción", "145 viviendas", "promociones", CURDATE())
```

Insertada una noticia con los valores indicados

UPDATE

Sintaxis:

```
UPDATE nombre_tabla  
SIETE nombre_columna1=expr1 [, nombre_columna2=expr2 ...]  
[WHERE condición]  
[ORDER BY ...]  
[LÍMITE row_count]
```

Ejemplo:

```
UPDATE NOTICIAS  
SIETE categoría = "ofertas"  
WHERE id=37
```

Modifica la categoría de la noticia con vayáis=37 de la tabla

DELETE

Sintaxis:

```
DELETE FROM nombre_tabla  
[WHERE condición]  
[ORDER BY ...]  
[LÍMITE row_count]
```

Ejemplo:

```
DELETE  
FROM NOTICIAS  
WHERE fecha < CURDATE()-10
```

Borra las noticias con más de 10 días de antigüedad

6. FUNCIONES DE PHP PARA ACCESO A BBDD MYSQL

Ver *ejemplo_Completo_Mysqli.php*:

Hay 3 APIs, de los cuales, una está desfasada:

- API Clásica → está desfasada
- API de integración **MySQLi** → **Es la que usaremos**
- API de integración **PDO_SQL**

Ver el siguiente enlace:

<https://www.php.net/manual/es/ref.mysql.php>

Los **PASOS** para acceder desde PHP en una base de datos son los siguientes:

1. **Conectar** con el servidor de bases de datos
2. **Seleccionar** una base de datos
3. Enviar la **instrucción SQL** a la base de datos
4. Obtener y procesar los **resultados**
5. **Cerrar la conexión** con el servidor de bases de datos

Establecer la Conexión

MySQL clásica	MySQLi	PDO
mysql_connect() : Función para abrir la conexión	new mysqli Crear nuevo objeto de conexión con los cadenas de conexión.	new PDO: Crear un nuevo objeto de conexión con los cadenas de conexión. La cadena “mysql”: al inicio es un estándar de conexión, así que inclúyela siempre.
mysql_select_db() : Selecciona la base de datos	La base de datos es seleccionada en el constructor PDO	La base de datos es seleccionada en el constructor PDO
Usar retorno booleano de mysql_connect() .	Usa el atributo connect_errno para comprobar la existencia de errores	Usaremos try-catch para manejar los excepciones de tipos PDOException
mysql_close() : Función que cierra la conexión	mysqli_close() : Método de la clase mysqli para cerrar la conexión.	Para cerrar la conexión asignas NULL a fin de conexión creada.

Realizar Consultas

MySQL clásica	MySQLi	PDO
<p>mysql_fetch_array(): Función que obtiene una fila de la consulta. El parámetro indica qué tipo de array será devuelto.</p> <p>MYSQL_NUM: Array de retorno con índices numéricos.</p> <p>MYSQL_ASSOC: Array de retorno con índices asociativos.</p> <p>MYSQL_BOTH: Array de retorno con ambos tipos de índices.</p>	<p>fetch_assoc() Método de la clase mysqli_result que obtiene una fila de la consulta en forma de array asociativo.</p>	<p>query() Método de la clase PDO que devuelve en un objeto PDOStatement que contiene los resultados de una consulta. Recorreremos cada elemento del objeto con un bucle <i>foreach.fetch()</i>: Método de la clase PDO para obtener una fila de una consulta.</p>
mysql_free_result() Libera la memoria asociada a los resultados de la consulta	free() : Libera la memoria asociada	Asigna NULL a la variable que recibió la referencia del resultado para liberar la memoria.

API DE INTEGRACIÓN MYSQLI

CONEXIÓN A la BD

Archivo T3/*conecta.php*

```
// Datos de conexión a la base de datos
$hostname='localhost';
$username='root';
$password='';
$database='clientesdb_dwes';

$link=mysqli_connect($hostname,$username,$password,$database);

if (!$link){
    echo "Error: No se pudo conectar a MySQL".PHP_EOL;
    echo "Error de depuración: ".mysqli_connect_errno(). '<br>';
}
else
{
    echo "Conexión Exitosa a bd:<br><b>".$database."</b><br>";
    // Cerramos la BD
    mysqli_close($link);
}
```

INSERCIÓN DE UN REGISTRO

Ingresar una tupla entra en una tabla.

Archivo: T3/**conecta_inserta.php**

```
//Datos de conexión a la base de datos
$hostname = 'localhost';
$username = 'root';
$password = '';
$database = 'clientesdb_dwes';

$link = mysqli_connect($hostname, $username,$password, $database);

if (!$link) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else
{
    echo "Conexión exitosa!<br>";
    echo "<b>BD: ">".$database."</b><br><br>";

    /* Inserta filas */
    $insert_query="INSERT INTO articulo "
        . "(idArticulo, Descripcion,Precio,Stock )"
        . " VALUES(12,'Articulo12',12,12)";

    echo $insert_query.'<br>';
    mysqli_query($link, $insert_query);
    printf("Affected rows (INSERT): %d\n", mysqli_affected_rows($link));

    // cerramos la BD
    mysqli_close($link)  ;
}
```

ACTUALIZACIÓN

Archivo: T3/conecta_Actualiza.php

```
//Datos de conexión a la base de datos
$hostname = 'localhost';
$username = 'root';
$password = '';
$database = 'clientesdb_dwes';

$link = mysqli_connect($hostname, $username,$password, $database);

if (!$link) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else
{
    echo "Conexión exitosa!<br>";
    echo "<b>BD: ">".$database."</b><br><br>";

    /* Actualiza filas */
    $update_query="UPDATE articulo SET "
        . " Descripcion='Chaleco_modif' WHERE idArticulo=1";

    echo $update_query."<br>";
    $result=mysqli_query($link, $update_query);
    $num_filas=mysqli_affected_rows($link);
    printf("Affected rows (UPDATE): %d\n", $num_filas);

    // cerramos la BD
    mysqli_close($link) ;
}
```

BORRADO DE REGISTROS

Archivo: T3/conecta_BorraRegistros.php

```
//Datos de conexión a la base de datos
$hostname = 'localhost';
$username = 'root';
$password = '';
$database = 'clientesdb_dwes';

$link = mysqli_connect($hostname, $username,$password, $database);

if (!$link) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else
{
    echo "Conexión exitosa!<br>";
    echo "<b>BD: ".$database."</b><br>";

    // Eliminar contenido
    $idElegido=2;
    $sql_del = "DELETE FROM articulo WHERE idArticulo =" . $idElegido;
    //ejecutamos la sentencia sql
    $resultDelete = mysqli_query($link,$sql_del);
    $numRegistrosBorrados= mysqli_affected_rows($link);
    if ($result){
        echo "<h3>Registros Borrados: ".$numRegistrosBorrados."</h3>";
    }
    else
    {
        echo "<h3>No se ha podido Borrar Artículo ".$idElegido."</h3>";
    }
}
```

BORRADO DE TABLA ENTERA

Archivo: T3/conecta_BorraTabla.php

```
//Datos de conexión a la base de datos
$hostname = 'localhost';
$username = 'root';
$password = '';
$database = 'clientesdb_dwes';

$link = mysqli_connect($hostname, $username,$password, $database);

if (!$link) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else
{
    echo "Conexión exitosa!<br>";
    echo "<b>BD: ".$database."</b><br>";
    // Eliminar tabla
    $drop_table_query = "DROP TABLE articulo";
    //ejecutamos la sentencia sql
    $result=mysqli_query($link,$drop_table_query);
    if ($result){
        echo "Tabla borrada";
    }
    else
    {
        echo "No se ha podido Borrar la tabla";
    }
    // cerramos la BD
    mysqli_close($link) ;
```

CONSULTA DE REGISTROS (SELECT) .

Mostrar en pantalla a partir de un bucle los registros seleccionados por una sentencia SQL:

Ejemplo 1:

Archivo: T3I/conecta_select.php

```
//Datos de conexión a la base de datos
$hostname='localhost';
$username='root';
$password='';
$database='clientesdb_dwes';

$link=mysqli_connect($hostname,$username,$password,$database);

if (!{$link}) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else
{
echo "Conexión exitosa!<br>";
echo "<b>BD: ".$database."</b><br><br>";
// Selecciona todas las filas
$select_query="SELECT * FROM articulo";
$result = mysqli_query($link, $select_query);
$num_filas_selected=mysqli_affected_rows($link);
printf("Affected rows (SELECT): %d\n", $num_filas_selected);
echo '<br>';
// Obtener todas las filas en un array asociativo
$rows = mysqli_fetch_all($result, MYSQLI_ASSOC);
// Recorro y visualizo el array de filas
foreach( $rows as $fila_actual)
{
    print_r( $fila_actual);
    echo '<br>';
}
mysqli_close($link);
```

Ejemplo 2:

Archivo: T3/conecta_Select2.php

```
//Datos de conexión a la base de datos
$hostname = 'localhost';
$username = 'root';
$password = '';
$database = 'clientesdb_iaw';

$link = mysqli_connect($hostname, $username,$password, $database);

if (!$link) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else
{
    echo "Conexión exitosa!<br>";
    echo "<b>BD: ". $database ."</b><br><br>";

    // OTRA FORMA de recorrer y visualizar el array de filas
    /* Selecciona todas las filas */
    $select_query="SELECT * FROM articulo";
    $result = mysqli_query($link, $select_query);
    $num_filas_selected=mysqli_affected_rows($link);
    printf("Affected rows (SELECT): %d\n", $num_filas_selected);

    echo '<br>';
    while ($fila_actual = mysqli_fetch_array($result, MYSQLI_ASSOC))
    {
        echo $fila_actual['idArticulo']. '<br>';
        echo $fila_actual['Descripcion']. '<br>';
        echo $fila_actual['Precio']. '<br>';
        echo $fila_actual['Stock']. '<br>'.'<br>';
    }

    // cerramos la BD
    mysqli_close($link) ;
}
```

OPERACIONES PREPARADAS:

Son aquellas que toman argumentos en tiempos de ejecución. Normalmente toman valores de un formulario y después se utilizan estos para construir la sentencia SQL

Ejemplo: *select from ARTICULO where idArticulo= valor_en_ejecución*

Forma 1: T3l/conecta_Preparada1.php

```
$hostname='localhost';
$username='root';
$password='';
$database='clientesdb_dwes';
$link=mysqli_connect($hostname,$username,$password,$database);
if (!$link) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else
{
echo "Conexión exitosa!<br>";
echo "<b>BD: ".$database."</b><br><br>";
/*CONSULTA preparada: una sentencia INSERT */
$prepared_insert_query = "INSERT INTO articulo "
    . "(idArticulo, Descripcion,Precio,Stock )"
    . " VALUES (?,?,?,?,?)";
echo $prepared_insert_query.'<br>';
$sentencia = mysqli_prepare($link, $prepared_insert_query);
mysqli_stmt_bind_param($sentencia, "isid", $val1, $val2, $val3,$val4);
$val1 = 13;           // idArticulo tipo integer (i)
$val2 = 'Articulo13'; // Descripción tipo String (s)
$val3 = 133.0;        // Precio tipo double(d)
$val4 = 13;           // stock tipo integer (i)
/* Ejecutar la sentencia */
$res=mysqli_stmt_execute($sentencia);
$numRegistrosInsertados= mysqli_affected_rows($link);
echo 'Registros insertados: '.$numRegistrosInsertados| mysqli_stmt_close($sentencia);
// cerramos la BD
mysqli_close($link);
}
```

Forma 2: PHP3/ej_AccesoMysql/conecta_Preparada2.php

```
$hostname='localhost';
$username='root';
$password='';
$database='clientesdb_dwes';
$link=mysqli_connect($hostname,$username,$password,$database);
if (!$link) {
    echo "Error: No se pudo conectar a MySQL." . PHP_EOL;
    echo "errno de depuración: " . mysqli_connect_errno()."<br>";
}
else{
    echo "Conexión exitosa!<br>";
    echo "<b>BD: ".$database."</b><br><br>";
/* otra forma de CONSULTA PREPARADA:crear una sentencia preparada */
if ($stmt = mysqli_prepare($link, "SELECT idArticulo FROM Articulo "
    ."WHERE Descripcion = ?"))
{// buscamos un producto a partir de su descripción
$descrip='Mochila M28';
/* ligar parámetros para marcadores */
mysqli_stmt_bind_param($stmt, "s", $descrip);
/* ejecutar la consulta */
mysqli_stmt_execute($stmt);
/* ligar variables de resultado */
mysqli_stmt_bind_result($stmt, $idArt);
/* obtener valor */
mysqli_stmt_fetch($stmt);
// mostramos el idarticulo asociado a la descripción
printf("%s es el idArticulo del producto %s\n" ,$idArt, $descrip );
/* cerrar sentencia */
mysqli_stmt_close($stmt);
}
// cerramos la BD
mysqli_close($link);
```

mysqli_prepare (stmt , types)

Prepara una sentencia SQL para la suya ejecución. Ver como se indica cada parámetro desconocido en tiempo de ejecución mediante un **interrogante ?**

mysqli_stmt_bind_params (stmt , types)

Se encarga de asociar los valores a los parámetros de la sentencia preparada.

- **stmt:**

Es la sentencia SQL con el formato que se ha visto en el ejemplo, es a decir, con un **interrogante ?** en cada valor que es desconoce en tiempo de compilación

- **types:**

Es un carácter que indica el tipo de dada del parámetro desconocido en cada posición. Es pueden usar los que se indican en l atabla siguiente:

Carácter	Descripción
i	la variable correspondiente es de tipo entero
d	la variable correspondiente es de tipo double
s	la variable correspondiente es de tipo string
b	la variable correspondiente es un blob y se envía en paquetes

mysqli_stmt_execute (stmt)

Ejecuta una consulta preparada

- **stmt:**

Es la sentencia SQL con el formato que se ha visto en el ejemplo, es a decir, con un **interrogante ?** en cada valor que es desconoce en tiempo de compilación

mysqli_stmt_bind_result (stmt , [col1], [col2],)

Vincula variables a una sentencia preparada para el almacenamiento de resultados

- **stmt:**

Es la sentencia SQL con el formato que se ha visto en el ejemplo, es a decir, con un **interrogante ?** en cada valor que es desconoce en tiempo de compilación

- **col1:**

recoge los resultados de la primera columna de la select

- **col2:**

recoge los resultados de la segunda columna de la select

mysqli_stmt_fetch (stmt)

Vincula/Obtiene los resultados de una sentencia preparadas en los variables vinculadas

stmt:

Es la sentencia SQL con el formato que se ha visto en el ejemplo, es a decir, con un **interrogante ?** en cada valor que es desconoce en tiempo de compilación

mysqli_stmt_close (stmt)

Cierra la sentencia preparada

stmt:

Es la sentencia SQL con el formato que se ha visto en el ejemplo, es a decir, con un **interrogante ?** en cada valor que es desconoce en tiempo de compilación

FUNCIONES DE LA API MYSQLI:

Los funciones concretas de MySQLi que realizan estas operaciones es encuentran en el enlace:

<https://www.php.net/manual/es/book.mysql.php>

El resumen es encuentra en:

<https://www.php.net/manual/es/mysql.summary.php>

API DE INTEGRACIÓN PDO_MYSQL

Ver el siguiente enlace:

<https://www.php.net/manual/es/ref.mysql.php>

CONEXIÓN A la BD

Archivo T3/**conecta_PDO.php**

```
<?php
//Datos de conexión PDO a la base de datos
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$port = 3306;
$nombreTabla = 'ARTICULO';
//cadena_conexion = "mysql:dbname=$database;host=$hostname";
$cadenaConexion = "mysql:"
    . "host=$hostname;" 
    . "dbname:$database;" 
    . "port=$port;";
try {
    $bd = new PDO($cadenaConexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . '<b>' . $database . '</b>' . '<br><b>';
    //Cerrar la conexión
    $bd = null;
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . '<b>' . $database . '</b>' . $e->getMessage();
}
?>
```

INSERCIÓN DE UN REGISTRO

Ingresar una tupla entra en una tabla.

Forma 1: Usando una consulta preparada con parámetros posicionales

Archivo: T3/conecta_inserta_PDO1.php

```
//Datos de conexión PDO a la base de datos
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Insert con parámetros posicionales</h4><br>';
    //Preparamos los parámetros
    $id = 12;
    $desc = 'articulo insert';
    $prec = 12;
    $stoc = 12;
    $insert_query = "INSERT INTO Articulo"
        . "(idArticulo, Descripcion, Precio, Stock) "
        . "VALUES "
        . "(?, ?, ?, ?)";
    //echo $insert_query.'<br>';
    //Preparamos la consulta
    $stmt = $pdo->prepare($insert_query);
    //Hacemos Bind da cada parámetro con su valor
    $stmt->bindParam(1, $id);
    $stmt->bindParam(2, $desc);
    $stmt->bindParam(3, $prec);
    $stmt->bindParam(4, $stoc);
    //Ejecutamos execute sin parámetros
    $stmt->execute();
    $num_filas = $stmt->rowCount();
    printf("Affected rows (INSERT): %d\n", $num_filas);
    echo '<br>';
    //cerramos bd
    $pdo=null;
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

Forma 2: Inserción con parámetros nominales (requiere un array asociativo que tiene como índice los nombres y con el valor, el de los parámetros)

Archivo: T3/**conecta_inserta_PDO2.php**

```
//Datos de conexión PDO a la base de datos
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Insert con parámetros nominales
        (Requiere un array asociativo)</h4><br>';
    //Preparamos los parámetros
    $arrDatos = [
        'id' => 11,
        'desc' => 'articulo insert',
        'prec' => 11,
        'stoc' => 11
    ];
    $insert_query = "INSERT INTO Articulo"
        . "(idArticulo, Descripcion, Precio, Stock) "
        . "VALUES "
        . "(:id,:desc,:prec,:stoc)";
    //Preparamos la consulta
    $stmt = $pdo->prepare($insert_query);
    //Ejecutamos execute con parámetros
    $stmt->execute($arrDatos);
    $num_filas = $stmt->rowCount();
    printf("Affected rows (INSERT): %d\n", $num_filas);
    echo '<br>';
    //cerramos bd
    $pdo=null;
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

TRANSACCIÓN

INSERCIÓN DE VARIOS REGISTROS

Forma 3: Inserción Múltiple. Varias filas. **Mediante Transacción** (usa array de filas(matriz))

Archivo: T3/conecta_insertaMult_PDO.php

```
//Datos de conexión PDO a la base de datos
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
$pdo = new PDO($cadena_conexion, $usuario, $clave);
echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
echo '<h4>Insert de múltiple filas <br>' .
    'con TRANSACCIÓN</h4><br>';
echo "<h5>Iniciamos TRANSACCIÓN....</h5>";
//Preparamos los datos en una matriz
$matDatos = [
    [20, 'articulo insert', 20, 20],
    [21, 'articulo insert', 21, 21]
];
$insert_query = "INSERT INTO Articulo"
    . "(idArticulo, Descripcion, Precio, Stock) "
    . "VALUES "
    . "(?, ?, ?, ?, ?)";
$stmt = $pdo->prepare($insert_query);
try {
    //Iniciamos transacción
    $pdo->beginTransaction();
    //Ejecuta cada inserción de fila
    foreach ($matDatos as $row) {
        $stmt->execute($row);
    }
    //Validamos
    $pdo->commit();
    //cerramos bd
    $pdo=null;
} catch (PDOException $e) {
    //Rollback si hay algún problema
    $pdo->rollBack();
    echo 'Error con la base de datos: <br>' . $database . $e->getMessage();
}
```

ACTUALIZACIÓN

Archivo: T3/conecta_Actualiza_PDO.php

```
//Datos de conexión PDO a la base de datos
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadenaConexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadenaConexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Actualiza con parámetros posicionales</h4><br>';
    $updateQuery = "UPDATE Articulo "
        . "SET descripcion=?, precio=?, stock=? "
        . "WHERE idArticulo=?";
    //Preparamos los parámetros
    $desc='articulo modificado';
    $prec=10;
    $stoc=100;
    $id=20; |
    //Preparamos la consulta
    $stmt = $pdo->prepare($updateQuery);

    //Hacemos Bind de cada parámetro con su valor
    $stmt->bindParam(1, $desc);
    $stmt->bindParam(2, $prec);
    $stmt->bindParam(3, $stoc);
    $stmt->bindParam(4, $id);
    //Ejecutamos execute sin parámetros
    $stmt->execute();
    $numFilas = $stmt->rowCount();
    printf("Affected rows (UPDATE): %d\n", $numFilas);
    echo '<br>';
    //cerramos bd
    $pdo=null;
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

BORRADO DE REGISTROS

Archivo: T3/conecta_Borra_PDO.php

```
//Datos de conexión PDO a la base de datos
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Delete con parámetros posicionales</h4><br>';
    //Preparamos los parámetros
    $id = 12;
    $delete_query = "DELETE FROM Articulo"
        . " WHERE idArticulo =?";
    //Preparamos la consulta
    $stmt = $pdo->prepare($delete_query);
    //Hacemos Bind de cada parámetro con su valor
    $stmt->bindParam(1, $id);
    //Ejecutamos execute sin parámetros
    $stmt->execute();
    $num_filas = $stmt->rowCount();
    printf("Affected rows (DELETE): %d\n", $num_filas);
    //cerramos bd
    $pdo = null;
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . '<b>' . $database . '</b>' . $e->getMessage();
}
```

BORRADO DE TABLA ENTERA

Archivo: PHP3/conecta_BorraTabla_PDO.php

```
-----  
//Datos de conexión PDO a la base de datos  
$hostname = 'localhost';  
$usuario = 'root';  
$clave = '';  
$database = 'clientesdb_dwes';  
$nombreTabla = 'ARTICULO';  
$cadenaConexion = "mysql:dbname=$database;host=$hostname";  
try {  
    $pdo = new PDO($cadenaConexion, $usuario, $clave);  
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';  
    echo '<h4>Drop Table</h4><br>';  
    $dropTable_query = "DROP TABLE articulo6";  
    //Preparamos la consulta  
    $stmt = $pdo->prepare($dropTable_query);  
    //Ejecutamos execute sin parámetros  
    if ($stmt->execute()) {  
        echo "Tabla eliminada satisfactoriamente";  
    } else {  
        print_r($sql->errorInfo());  
    }  
    //cerramos bd  
    $pdo = null;  
} catch (PDOException $e) {  
    echo 'Error con la base de datos: ' . '<b>' . $database . '</b>' . $e->getMessage();  
}
```

CONSULTA DE REGISTROS (SELECT) .

Mostrar en pantalla a partir de un bucle los registros seleccionados por una sentencia SQL:

Ejemplo 1: Select con parámetros posicionales

Archivo: T3I/conecta_select_PDO1.php

```
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Select con parámetros posicionales</h4><br>';
    //Preparamos los parámetros
    $id = 1;
    $select_query = "SELECT * FROM articulo WHERE idArticulo=?";
    //Preparamos la consulta
    $stmt = $pdo->prepare($select_query);
    $stmt->execute([$id]);
    $num_filas_selected = $stmt->rowCount();
    printf("Affected rows (SELECT): %d\n", $num_filas_selected);
    echo '<br><br>';
    echo 'Visualizamos datos:<br>';
    $data = $stmt->fetchAll();
    foreach ($data as $row) {
        echo $row['idArticulo'] . "<br />\n";
        echo $row['Descripcion'] . "<br />\n";
        echo $row['Precio'] . "<br />\n";
        echo $row['Stock'] . "<br />\n";
    }
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

Ejemplo 2: Select con parámetros nominales

Archivo: T3I/conecta_select_PDO2.php

```
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Select con parámetros nominales</h4><br>';
    //Preparamos los parámetros
    $id = 1;
    $select_query = "SELECT * FROM articulo"
        . " WHERE idArticulo=:id";
    //Preparamos la consulta
    $stmt = $pdo->prepare($select_query);
    $stmt->execute(['id' => $id]);
    $num_filas_selected = $stmt->rowCount();
    printf("Affected rows (SELECT): %d\n", $num_filas_selected);
    echo '<br>';
    $data = $stmt->fetchAll();
    foreach ($data as $row) {
        echo $row['idArticulo'] . "<br />\n";
        echo $row['Descripcion'] . "<br />\n";
        echo $row['Precio'] . "<br />\n";
        echo $row['Stock'] . "<br />\n";
    }
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

CURSORES

Definir cursos:

<https://www.php.net/manual/es/pdostatement.fetch.php>

Sintaxis:

PDOStatement::fetch — Obtiene la siguiente fila de un conjunto de resultados

```
public PDOStatement::fetch(int $mode = PDO::FETCH_DEFAULT, int $cursorOrientation = PDO::FETCH_ORI_NEXT, int  
$cursorOffset = 0): mixed
```

Obtiene una fila de un conjunto de resultados asociado al objeto PDOStatement. El parámetro mode determina cómo PDO devuelve la fila.

Parámetros

mode

Controla cómo se devolverá la siguiente fila al llamador. Este valor debe ser una de las constantes **PDO::FETCH_***, estando predeterminado **PDO::ATTR_DEFAULT_FETCH_MODE** (el cual por defecto es **PDO::FETCH_BOTH**).

- **PDO::FETCH_ASSOC**: devuelve un array indexado por los nombres de las columnas del conjunto de resultados.
- **PDO::FETCH_BOTH** (predeterminado): devuelve un array indexado tanto por nombre de columna, como numéricamente con índice de base 0 tal como fue devuelto en el conjunto de resultados.
- **PDO::FETCH_BOUND**: devuelve **true** y asigna los valores de las columnas del conjunto de resultados a las variables de PHP a las que fueron vinculadas con el método [PDOStatement::bindColumn\(\)](#).
- **PDO::FETCH_CLASS**: devuelve una nueva instancia de la clase solicitada, haciendo corresponder las columnas del conjunto de resultados con los nombres de las propiedades de la clase, y llamando al constructor después, a menos que también se proporcione **PDO::FETCH_PROPS_LATE**. Si mode incluye PDO::FETCH_CLASSTYPE (por ejemplo, PDO::FETCH_CLASS | PDO::FETCH_CLASSTYPE), entonces el nombre de la clase se determina a partir del valor de la primera columna.

- **PDO::FETCH INTO**: actualiza una instancia existente de la clase solicitada, haciendo coincidir el nombre de las columnas con los nombres de las propiedades de la clase.
- **PDO::FETCH LAZY**: combina **PDO::FETCH BOTH** y **PDO::FETCH OBJ**, creando los nombres de la variables del objeto tal como se accedieron.
- **PDO::FETCH NAMED**: devuelve un array con la misma forma que **PDO::FETCH ASSOC**, excepto que si hubiera múltiples columnas con el mismo nombre, el valor al que hace referencia dicha clave será un array con todos los valores de la fila de tuviera ese nombre de columna.
- **PDO::FETCH NUM**: devuelve un array indexado por el número de columna tal como fue devuelto en el conjunto de resultados, comenzando por la columna 0.
- **PDO::FETCH OBJ**: devuelve un objeto anónimo con nombres de propiedades que se corresponden a los nombres de las columnas devueltas en el conjunto de resultados.
- **PDO::FETCH PROPS LATE**: cuando se usa con **PDO::FETCH CLASS**, se llama al constructor de la clase antes de que las proiedades sean asignadas desde los valores de la columna respectiva.

cursorOrientation

Para un objeto PDOStatement que represente un cursor desplazable, este valor determina qué columna será devuelta por el llamador. Este valor debe ser una de las constantes **PDO::FETCH_ORI_***, siendo la predeterminada **PDO::FETCH_ORI_NEXT**. Para solicitar un cursor desplazable para el objeto PDOStatement, se debe establecer el atributo **PDO::ATTR_CURSOR** a **PDO::CURSOR_SCROLL** cuando se prepare la sentencia SQL con **PDO::prepare()**.

cursorOffset

Para un objeto PDOStatement que represente un cursor desplazable para el cual el parámetro cursorOrientation está establecido a PDO::FETCH_ORI_ABS, este valor especifica el número absoluto de la fila del conjunto de resultados que se desea obtener.

Para un objeto PDOStatement que represente un cursor desplazable para el cual el parámetro cursorOrientation está establecido a PDO::FETCH_ORI_REL, este valor especifica la fila a obtener relativa a la posición del cursor antes de que se llame a **PDOStatement::fetch()**.

Valores devueltos

El valor de retorno de esta función en caso de éxito depende del tipo de obtención. En todos los casos, se devuelve false en caso de error o si no existen más filas por devolver. Errors/Exceptions ¶

Ejemplo 3: Usando un CURSOR y accediendo a los datos a través de array indexado por **número** de columna.

Recorrido ASCENDENTE

Archivo: T3I/conecta_select_PDO3.php

```
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Select CURSOR fetch</h4><br>';
    echo("Devolver la siguiente fila como un array "
        . "indexado por número de columnna<br>");
    echo 'RECORRER FORWARD<br>';

    $select_query = "SELECT * FROM articulo";
    //Preparamos la consulta
    $stmt = $pdo->prepare($select_query, [PDO::ATTR_CURSOR => PDO::CURSOR_SCROLL]);
    //Ejecutar sin parámetros
    $stmt->execute();
    while ($row = $stmt->fetch(PDO::FETCH_NUM, PDO::FETCH_ORI_NEXT))
    {
        echo $row[0] . '<br>';
        echo $row[1] . '<br>';
        echo $row[2] . '<br>';
        echo $row[3] . '<br><br>';
    }
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

Ejemplo 4: Usando un CURSOR y accediendo a los datos a través de array indexado por **nombre** de columna.

Recorrido ASCENDENTE

Archivo: T3I/conecta_select_PDO4.php

```
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Select CURSOR fetch</h4><br>';
    echo("Devolver la siguiente fila como un array "
        . "indexado por NOMBRE de columna<br>");
    echo 'RECORRER FORWARD<br>';
    $select_query = "SELECT * FROM articulo";
    //Preparamos la consulta
    $stmt = $pdo->prepare($select_query, [PDO::ATTR_CURSOR => PDO::CURSOR_SCROLL]);
    # Ejecutar sin parámetros
    $stmt->execute();
    while ($row = $stmt->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_NEXT)) {
        echo $row['idArticulo']. '<br>';
        echo $row['Descripcion']. '<br>';
        echo $row['Precio']. '<br>';
        echo $row['Stock']. '<br><br>';
        //print_r($row);
    }
    $num_filas_selected = $stmt->rowCount();
    printf("Affected rows (SELECT): %d\n", $num_filas_selected);
    echo '<br>';
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

Ejemplo 5:

Usando un CURSOR y accediendo a los datos a través de array indexado por **nombre** de columna. Recorrido DESCENDENTE ➔ NO FUNCIONA CORRECTAMENTE

Archivo: T3I/conecta_select_PDO5.php

```
$hostname = 'localhost';
$usuario = 'root';
$clave = '';
$database = 'clientesdb_dwes';
$nombreTabla = 'ARTICULO';
$cadena_conexion = "mysql:dbname=$database;host=$hostname";
try {
    $pdo = new PDO($cadena_conexion, $usuario, $clave);
    echo 'Conexión Exitosa a bd ' . $database . '<br><b>';
    echo '<h4>Select CURSOR fetch</h4><br>';
    echo("Devolver la siguiente fila como array indexado por NOMBRE de columnna<br>");
    echo 'RECORRER BACKWARDS<br>';
    $select_query = "SELECT * FROM articulo";
    //Preparamos la consulta
    $stmt = $pdo->prepare($select_query, [PDO::ATTR_CURSOR => PDO::CURSOR_SCROLL]);
    // Ejecutar sin parámetros
    $stmt->execute();
    //posicionamos acceso a última fila
    $row = $stmt->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_LAST);
    print_r($row);
    do{
        echo $row['idArticulo']. '<br>';
        echo $row['Descripcion']. '<br>';
        echo $row['Precio']. '<br>';
        echo $row['Stock']. '<br><br>';
        //print_r($row);
    }while ($row = $stmt->fetch(PDO::FETCH_ASSOC, PDO::FETCH_ORI_PRIOR));
} catch (PDOException $e) {
    echo 'Error con la base de datos: ' . $database . $e->getMessage();
}
```

FUNCIONES DE LA API PDO:

Los funciones concretas de PDO que realizan estas operaciones es encuentran en el enlace:

<https://www.php.net/manual/es/book pdo.php>

El resumen es encuentra en:

<https://www.php.net/manual/es/mysqli.summary.php>