TEMA 3

Contenido

Trabajar con bases de datos en PHP.

1.- Acceso a bases de datos desde PHP.

Una de las aplicaciones más frecuentes de PHP es generar un interface web para acceder y gestionar la información almacenada en una base de datos. Usando PHP podemos mostrar en una página web información extraída de la base de datos, o enviar sentencias al gestor de la base de datos para que elimine o actualice algunos registros.

PHP soporta más de 15 sistemas gestores de bases de datos: SQLite, Oracle, SQL Server, PostgreSQL, IBM DB2, MySQL, etc. Hasta la versión 5 de PHP, el acceso a las bases de datos se hacía principalmente utilizando extensiones específicas para cada sistema gestor de base de datos (extensiones nativas). Es decir, que si queríamos acceder a una base de datos de PostgreSQL, deberíamos instalar y utilizar la extensión de ese gestor en concreto. Las funciones y objetos a utilizar eran distintos para cada extensión.

A partir de la versión 5 de PHP se introdujo en el lenguaje una extensión para acceder de una forma común a distintos sistemas gestores: PDO. La gran ventaja de PDO está clara: podemos seguir utilizando una misma sintaxis aunque cambiemos el motor de nuestra base de datos. Por el contrario, en algunas ocasiones preferiremos seguir usando extensiones nativas en nuestros programas. Mientras PDO ofrece un conjunto común de funciones, las extensiones nativas normalmente ofrecen más potencia (acceso a funciones específicas de cada gestor de base de datos) y en algunos casos también mayor velocidad.

De los distintos SGBD existentes, vas a aprender a utilizar MySQL. MySQL es un gestor de bases de datos relacionales de código abierto bajo licencia GNU GPL. Es el gestor de bases de datos más empleado con el lenguaje PHP. Como ya vimos, es la letra "M" que figura en los acrónimos AMP y XAMPP.

En esta unidad vas a ver cómo acceder desde PHP a bases de datos MySQL utilizando tanto PDO como la extensión nativa MySQLi. Previamente verás una pequeña introducción al manejo de MySQL, aunque para el seguimiento de esta unidad se supone que conoces el lenguaje SQL utilizado en la gestión de bases de datos relacionales.

Además, para el acceso a las funcionalidades de ambas extensiones deberás utilizar objetos. Aunque más adelante verás todas las características que nos ofrece PHP para crear programas orientados a objetos, debemos suponer también en este punto un cierto conocimiento de programación orientada a objetos. Básicamente, debes saber cómo crear y utilizar objetos.

En PHP se utiliza la palabra `new` para crear un nuevo objeto instanciando una clase:

```
$a = new A();
```

Y para acceder a los miembros de un objeto, debes utilizar el operador flecha `->`:

```
$a->fecha();
```

Características básicas de la utilización de objetos en PHP

La programación orientada a objetos es una metodología de programación avanzada y bastante extendida, en la que los sistemas se modelan creando clases, que son un conjunto de datos y funcionalidades. Las clases son definiciones, a partir de las que se crean objetos. Los objetos son ejemplares de una clase determinada y como tal, disponen de los datos y funcionalidades definidos en la clase.

La programación orientada a objetos permite concebir los programas de una manera bastante intuitiva y cercana a la realidad. La tendencia es que un mayor número de lenguajes de programación adopten la programación orientada a objetos como paradigma para modelizar los sistemas. Prueba de ello es la nueva versión de PHP (5), que implanta la programación de objetos como metodología de desarrollo. También Microsoft ha dado un vuelco hacia la programación orientada a objetos, ya que .NET dispone de varios lenguajes para programar y todos orientados a objetos.

Así pues, la programación orientada a objetos es un tema de gran interés, pues es muy utilizada y cada vez resulta más esencial para poder desarrollar en casi cualquier lenguaje moderno. En este artículo vamos ver algunas nociones sobre la programación orientada a objetos en PHP. Aunque es un tema bastante amplio, novedoso para muchos y en un principio, difícil de asimilar, vamos a tratar de explicar la sintaxis básica de PHP para utilizar objetos, sin meternos en mucha teoría de programación orientada a objetos en general.

Las clases: class

Una clase es un conjunto de variables, llamados atributos, y funciones, llamadas métodos, que trabajan sobre esas variables. Las clases son, al fin y al cabo, una definición: una especificación de propiedades y funcionalidades de elementos que van a participar en nuestros programas.

Por ejemplo, la clase "Caja" tendría como atributos características como las dimensiones, color, contenido y cosas semejantes. Las funciones o métodos que podríamos incorporar a la clase "caja" son las funcionalidades que deseamos que realice la caja, como `introduce()`, `muestra contenido()`, `comprueba si cabe()`, `vaciate()`...

Las clases en PHP se definen de la siguiente manera:

```
<?
class Caja{
   var $alto;
   var $ancho;
   var $largo;
   var $contenido;
   var $color;

   function introduce($cosa){
      $this->contenido = $cosa;
   }
```

```
   function muestra_contenido(){
      echo $this->contenido;
   }
}
?>
```

En este ejemplo se ha creado la clase Caja, indicando como atributos el ancho, alto y largo de la caja, así como el color y el contenido. Se han creado, para empezar, un par de métodos, uno para introducir un elemento en la caja y otro para mostrar el contenido.

Si nos fijamos, los atributos se definen declarando unas variables al principio de la clase. Los métodos se definen declarando funciones dentro de la clase. La variable `$this`, utilizada dentro de los métodos la explicaremos un poco más abajo.

Utilizar la clase

Las clases solamente son definiciones. Si queremos utilizar la clase tenemos que crear un ejemplar de dicha clase, lo que corrientemente se le llama instanciar un objeto de una clase.

```
$micaja = new Caja;
```

Con esto hemos creado, o mejor dicho, instanciado, un objeto de la clase Caja llamado `$micaja`.

```
$micaja->introduce("algo");
$micaja->muestra_contenido();
```

Con estas dos sentencias estamos introduciendo "`algo`" en la caja y luego estamos mostrando ese contendido en el texto de la página. Nos fijamos que los métodos de un objeto se llaman utilizando el código "`->`".

```
nombre_del_objeto->nombre_de_metodo()
```

Para acceder a los atributos de una clase también se accede con el código "`->`". De esta forma:

```
nombre_del_objeto->nombre_del_atributo
```

La variable $this

Dentro de un método, la variable `$this` hace referencia al objeto sobre el que invocamos el método. En la invocación `$micaja->introduce("algo")` se está llamando al método introduce sobre el objeto `$micaja`. Cuando se está ejecutando ese método, se vuelca el valor que recibe por parámetro en el atributo contenido. En ese caso `$this->contenido` hace referencia al atributo contenido del objeto `$micaja`, que es sobre el que se invocaba el método.

Constructores

Los constructores son funciones, o métodos, que se encargan de realizar las tareas de inicialización de los objetos al ser instanciados. Es decir, cuando se crean los objetos a partir de las clases, se llama a un constructor que se encarga de inicializar los atributos del objeto y realizar cualquier otra tarea de inicialización que sea necesaria.

No es obligatorio disponer de un constructor, pero resultan muy útiles y su uso es muy habitual. En el ejemplo de la caja, que comentábamos anteriormente, lo normal sería inicializar las variables como color o las relacionadas con las dimensiones y, además, indicar que el contenido de la caja está vacío. Si no hay un constructor no se inicializan ninguno de los atributos de los objetos.

El constructor se define dentro de la propia clase, como si fuera otro método. El único detalle es que el constructor debe tener el mismo nombre que la clase. Atentos a PHP, que diferencia entre mayúsculas y minúsculas.

Para la clase Caja definida anteriormente, se podría declarar este constructor:

```
function Caja($alto=1,$ancho=1,$largo=1,$color="negro"){
   $this->alto=$alto;
   $this->ancho=$ancho;
   $this->largo=$largo;
   $this->color=$color;
   $this->contenido="";
}
```

En este constructor recibimos por parámetro todos los atributos que hay que definir en una caja.
Es muy útil definir unos valores por defecto en los parámetros que recibe el constructor, igualando el parámetro a un valor dentro de la declaración de parámetros de la función constructora, pues así, aunque se llame al constructor sin proporcionar parámetros, se inicializará con los valores por defecto que se hayan definido.

Es importante señalar que en los constructores no se tienen por qué recibir todos los valores para inicializar el objeto. Hay algunos valores que pueden inicializarse a vacío o a cualquier otro valor fijo, como en este caso el contenido de la caja, que inicialmente hemos supuesto que estará vacía.

2.- MySQL.

MySQL es un sistema gestor de bases de datos (SGBD) relacionales. Es un programa de código abierto que se ofrece bajo licencia GNU GPL, aunque también ofrece una licencia comercial en caso de que quieras utilizarlo para desarrollar aplicaciones de código propietario. En las últimas versiones (a partir de la 5.1), se ofrecen, de hecho, varios productos distintos: uno de código libre (Community Edition), y otro u otros comerciales (Standard Edition, Enterprise Edition).

Incorpora múltiples motores de almacenamiento, cada uno con características propias: unos son más veloces, otros, aportan mayor seguridad o mejores capacidades de búsqueda. Cuando crees una base de datos, puedes elegir el motor en función de las características propias de la aplicación. Si no lo cambias, el motor que se utiliza por defecto se llama `MyISAM`, que es muy rápido pero a cambio no contempla integridad referencial (*característica de las bases de datos que permite crear relaciones válidas entre dos registros de la misma o de diferentes tablas, y definir las operaciones necesarias para mantener la validez de las relaciones cuando se borra o modifica alguno de los registros*) ni tablas transaccionales (*conjunto de operaciones sobre los datos que se han de realizar de forma conjunta, una sola vez, e independientemente del resto de manipulaciones sobre los datos. Toda transacción debe cumplir cuatro propiedades: atomicidad, consistencia, aislamiento y permanencia*). El motor `InnoDB` es un poco más lento pero sí soporta tanto integridad referencial como tablas transaccionales.

MySQL se emplea en múltiples aplicaciones web, ligado en la mayor parte de los casos al lenguaje PHP y al servidor web **Apache**. Utiliza SQL para la gestión, consulta y modificación de la información almacenada. Soporta la mayor parte de las características de ANSI SQL 99 (*revisión del estándar ANSI SQL del año 1999, que agrega a la revisión anterior (SQL2 o SQL 92) disparadores, expresiones regulares, y algunas características de orientación a objetos*), y añade además algunas extensiones propias.

En las siguientes secciones darás un rápido repaso a lo que debes saber sobre la instalación, configuración y las herramientas de administración de MySQL. Si necesitas ampliar información, puedes consultar el manual en línea de MySQL.

http://dev.mysql.com/doc/refman/5.0/es/index.html

¿A qué hacen referencia las siglas PDO?

- ○ A un motor de almacenamiento utilizado por MySQL.
- ◉ **A una extensión de PHP que permite acceder a varios gestores de bases de datos.**

los motores de almacenamiento de los que hablamos son MyISAM e InnoDB. Aunque no son los únicos que se pueden utilizar con MySQL sí son los más comunes.

2.1.- Instalación y configuración.

En la primera unidad ya viste cómo podías instalar en un único paso una plataforma **LAMP** para desarrollar aplicaciones web en **Ubuntu**. En Linux, la instalación de MySQL se divide básicamente en dos paquetes que puedes instalar de forma individual según tus necesidades:

- ✓ **mysql-server**. Es el servidor en sí. Necesitas instalar este paquete para gestionar las bases de datos y permitir conexiones desde el equipo local o a través de la red.
- ✓ **mysql-client**. Son los programas cliente, necesarios para conectarse a un servidor MySQL. Solo necesitas instalarlos en aquel o aquellos equipos que se vayan a conectar al SGBD (en nuestro caso, las conexiones se realizarán normalmente desde el mismo equipo en el que se ejecuta el servidor).

Una vez instalado, puedes gestionar la ejecución del servicio de la misma forma que cualquier otro servicio del sistema:

```
sudo service mysql status // también start, stop, restart
```

En una instalación típica, el usuario `root` no tiene por defecto contraseña de acceso al servidor. Es importante asignarle una por razones de seguridad:

```
mysqladmin -u root password nueva-contraseña
```

El servidor se ejecuta por defecto en el Puerto TCP 3306. Esto lo debes tener en cuenta para permitir el acceso a través del cortafuegos en configuraciones en red.

El fichero de configuración del servidor MySQL se llama `my.cnf` y se encuentra alojado en `/etc/mysql`. Su contenido se divide en secciones. Las opciones que contiene cada una de las secciones afectan al comportamiento de un módulo concreto. Entre las secciones disponibles destacan:

- ✓ `[client]`. Sus parámetros influyen sobre los distintos clientes que se conectan al servidor MySQL.
- ✓ `[mysqld]`. Contiene opciones relativas a la ejecución del servidor.

Además del fichero global de configuración, las opciones de ejecución se pueden aplicar por línea de comandos y obtener de otros orígenes distintos. Puedes consultar más información en el manual en línea de MySQL.

http://dev.mysql.com/doc/refman/5.0/es/index.html

Entre los parámetros que puedes configurar en el fichero my.cnf tienes:

- ✓ **port**. Indica el puerto TCP en el que escuchará el servidor y con el que se establecerán las conexiones.
- ✓ **user**. Nombre del usuario que se utilizará para ejecutar el servidor.
- ✓ **datadir**. Directorio del servidor en el que se almacenarán las bases de datos.

En la documentación de MySQL tienes también información sobre todas las opciones de configuración disponibles para ajustar su funcionamiento.

http://dev.mysql.com/doc/refman/5.0/es/index.html

2.2.- Herramientas de administración.

Existen muchas herramientas que permiten establecer una conexión con un servidor MySQL para realizar tareas de administración. Algunas herramientas se ejecutan en la línea de comandos, otras presentan un interface gráfico basado en web o propio del sistema operativo en que se ejecuten. Unas se incluyen con el propio servidor, y otras es necesario obtenerlas e instalarlas de forma independiente. Las hay que están orientadas a algún propósito concreto y también que permiten realizar varias funciones de administración.

Con el servidor MySQL se incluyen algunas herramientas de administración en línea de comandos, entre las que debes conocer:

- ✓ **mysql**. Permite conectarse a un servidor MySQL para ejecutar sentencias SQL.
- ✓ **mysqladmin**. Es un cliente específico para tareas de administración.
- ✓ **mysqlshow**. Muestra información sobre bases de datos y tablas.

En la documentación de MySQL tienes información sobre las distintas utilidades que incorpora.

http://dev.mysql.com/doc/refman/5.0/es/client-side-scripts.html

Estas herramientas comparten unas cuantas opciones relativas al establecimiento de la conexión con el servidor. Muchas de estas opciones tienen también una forma abreviada:

- ✓ `--user=nombre_usuario` (`-u nombre_usuario`). Indica un nombre de usuario con permisos para establecer la conexión. Si no se especifica se usará el nombre de usuario actual del sistema operativo.
- ✓ `--password=contraseña` (`-pcontraseña`). Contraseña asociada al nombre de usuario anterior. Si se utiliza la opción abreviada, debe figurar justo a continuación de la letra p, sin espacios intermedios. Si es necesario introducir una contraseña y no se indica ninguna, se pedirá para establecer la conexión.
- ✓ `--host=equipo_servidor` (`-h equipo_servidor`). Nombre del equipo con el que se establecerá la conexión. Si no se indica nada, se usara "`localhost`".

Por ejemplo, para establecer una conexión al servidor local con la herramienta mysql, podemos hacer:

```
mysql -u root -p
```

Conviene no indicar nunca la contraseña en la misma línea de comandos. En caso de que la cuenta esté convenientemente protegida por una contraseña, es mejor utilizar solo la opción -p como en el ejemplo anterior. De esta forma, la herramienta solicita la introducción de la contraseña y ésta no queda almacenada en ningún registro como puede ser el historial de comandos del sistema.

De entre el resto de herramientas de administración independientes que podemos utilizar con MySQL, podemos destacar dos:

- ✓ **MySQL Workbench** es una herramienta genérica con interface gráfico nativo que permite administrar tanto el servidor como las bases de datos que éste gestiona. Ha sido desarrollada por los creadores de MySQL y se ofrece en dos ediciones, una de ellas de código abierto bajo licencia GPL.

  http://dev.mysql.com/doc/workbench/en/index.html
- ✓ **phpMyAdmin** es una aplicación web muy popular para la administración de servidores MySQL. Presenta un interface web de administración programado en PHP bajo licencia GPL. Su objetivo principal es la administración de las bases de datos y la gestión de la información que maneja el servidor.

  http://www.phpmyadmin.net/

Relaciona cada herramienta de administración con el tipo de interface que utiliza:

Herramienta.	Relación.	Tipo de interface.
MySQL Workbench.	**3**	1. Línea de comandos.
mysql.	**1**	2. Web.
phpMyAdmin.	**2**	3. Nativo.
mysqladmin.	**4**	4. Línea de comandos.

2.2.1.- mysql y mysqladmin.

La forma más habitual de utilizar la herramienta `mysql` es en modo interactivo. Una vez te conectas al servidor MySQL, te presenta una línea de órdenes. En esa línea de órdenes puedes introducir sentencias SQL, que se ejecutarán sobre la base de datos seleccionada, y algunos comandos

especiales. Las sentencias SQL deben terminar en el carácter "`;`". Entre los comandos especiales que puedes usar están:

- ✓ `connect`. Establece una conexión con un servidor MySQL.
- ✓ `use`. Permite seleccionar una base de datos.
- ✓ `exit` o `quit`. Termina la sesión interactiva con mysql.
- ✓ `help`. Muestra una pantalla de ayuda con la lista de comandos disponibles.

Por ejemplo, si cuando estás utilizando la herramienta quieres seleccionar la base de datos "dwes", debes hacer:

```
mysql> use dwes
```

Las sentencias SQL que teclees a partir de ese instante se ejecutarán sobre la base de datos "dwes".

Para comprobar la sintaxis de las sentencias SQL admitidas por MySQL, puedes consultar la documentación en línea.

http://dev.mysql.com/doc/refman/5.0/es/sql-syntax.html

También puedes usar mysql en modo de procesamiento por lotes (*ejecución de un conjunto de tareas repetitivas o de forma consecutiva, sin la supervisión directa del usuario*), para ejecutar sobre un servidor MySQL todas las sentencias almacenadas en un fichero de texto (normalmente con extensión .sql). Por ejemplo:

```
mysql –u root –pabc123. < crear_bd_dwes.sql
```

Utiliza las sentencias SQL que contiene el siguiente fichero para crear la estructura de la base de datos "dwes" en tu instalación de MySQL.

```
-- Creamos la base de datos
CREATE DATABASE 'dwes' DEFAULT CHARACTER SET utf8 COLLATE utf8_spanish_ci;
USE 'dwes';

-- Creamos las tablas
CREATE TABLE  'dwes'.'tienda' (
'cod' INT NOT NULL AUTO INCREMENT PRIMARY KEY ,
'nombre' VARCHAR( 100 ) NOT NULL ,
'tlf' VARCHAR( 13 ) NULL
) ENGINE = INNODB;

CREATE TABLE  'dwes'.'producto' (
'cod' VARCHAR( 12 ) NOT NULL ,
'nombre' VARCHAR( 200 ) NULL ,
'nombre_corto' VARCHAR( 50 ) NOT NULL ,
'descripcion' TEXT NULL ,
'PVP' DECIMAL( 10, 2 ) NOT NULL ,
'familia' VARCHAR( 6 ) NOT NULL ,
PRIMARY KEY (  'cod' ) ,
INDEX (  'familia' ) ,
UNIQUE ( 'nombre_corto' )
) ENGINE = INNODB;

CREATE TABLE  'dwes'.'familia' (
'cod' VARCHAR( 6 ) NOT NULL ,
'nombre' VARCHAR( 200 ) NOT NULL ,
PRIMARY KEY (  'cod' )
) ENGINE = INNODB;

CREATE TABLE  'dwes'.'stock' (
'producto' VARCHAR( 12 ) NOT NULL ,
'tienda' INT NOT NULL ,
'unidades' INT NOT NULL ,
PRIMARY KEY (  'producto' ,  'tienda' )
) ENGINE = INNODB;

-- Creamos las claves foráneas
ALTER TABLE 'producto'
ADD CONSTRAINT 'producto_ibfk_1'
FOREIGN KEY ('familia') REFERENCES 'familia' ('cod')
ON UPDATE CASCADE;
```

```
ALTER TABLE 'stock'
ADD CONSTRAINT 'stock ibfk 2'
FOREIGN KEY ('tienda') REFERENCES 'tienda' ('cod')
ON UPDATE CASCADE,
ADD CONSTRAINT 'stock_ibfk_1'
FOREIGN KEY ('producto') REFERENCES 'producto' ('cod')
ON UPDATE CASCADE;

CREATE USER 'dwes'
IDENTIFIED BY 'abc123.';

GRANT ALL ON 'dwes'.*
TO 'dwes';
```

Utilizando la herramienta mysql ejecuta:

```
mysql -u root -p < dwes.sql
```

mysqladmin es una herramienta no interactiva orientada a tareas de administración del propio servidor. Las tareas concretas de administración a llevar a cabo, se indican mediante parámetros en la línea de comandos. Entre las tareas que puedes llevar a cabo con esta utilidad se encuentran:

- ✓ crear y eliminar bases de datos.
- ✓ mostrar la configuración y el estado del servidor.
- ✓ cambiar contraseñas.
- ✓ detener un servidor.

Por ejemplo, si quieres mostrar información sobre el estado actual del servidor local, puedes utilizar el comando **status**:

```
mysqladmin -u root -pabc123. status
```

En la documentación de MySQL tienes información sobre todos los comandos que admite mysqladmin y su significado.

http://dev.mysql.com/doc/refman/5.0/es/mysqladmin.html

Si quieres saber si en una tabla de una base de datos existe o no un registro, ¿qué herramienta en línea de comandos puedes usar?

- ○ mysqladmin.
- ◉ **mysql.**

La herramienta mysqladmin la puedes utilizar para realizar tareas administrativas, pero no para ejecutar consultas sobre el contenido de las bases de datos

2.2.2.- phpMyAdmin.

Al contrario que las dos herramientas anteriores, **phpMyAdmin** no se instala con el servidor MySQL. Debes instalarlo de forma individual, en el caso de Ubuntu utilizando el gestor de paquetes:

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

El proceso de instalación es sencillo. Simplemente te pregunta por el servidor web a utilizar (escoger apache2), y después debes dejar que configure una nueva base de datos propia en el servidor. Una vez instalada la aplicación, podrás acceder vía web con un navegador utilizando la URL "**http://localhost/phpmyadmin/**".

Para poder entrar, debes indicar un nombre de usuario y contraseña válidos. Si realizaste el ejercicio anterior, se habrá creado en tu servidor un **usuario** "**dwes**" con **contraseña** "**abc123.**" con **permisos**

para la base de datos "`dwes`". Si utilizas ese usuario para entrar en la aplicación, ésta te permitirá gestionar la base de datos "`dwes`".

El interface de la aplicación se compone de un panel de navegación a la izquierda, donde se muestran las bases de datos, y un panel principal con un menú en la parte superior y una serie de acciones e información en la parte central. Si seleccionas la base de datos "`dwes`", la información en pantalla cambia.

Utilizando los menús de la parte superior, puedes:

- ✓ Ver y modificar la **estructura** de la base de datos.
- ✓ Ejecutar sentencias **SQL**.
- ✓ **Buscar** información en toda la base de datos o en parte de la misma.
- ✓ **Generar una consulta** utilizando un asistente.
- ✓ **Exportar** e **importar** información, tanto de la estructura como de los datos.
- ✓ **Diseñar** las relaciones existentes entre las tablas.
- ✓ Otras **operaciones**, como hacer una copia de la base de datos.

Si seleccionas una tabla en lugar de la base de datos, podrás efectuar a ese nivel operaciones similares a las anteriores. En la siguiente presentación sobre `phpMyAdmin` tienes información sobre el maneja básico de la aplicación.

En la página web de la aplicación tienes documentación sobre su configuración y utilización.
http://www.phpmyadmin.net/localized_docs/es/Documentation.html

Utiliza `phpMyAdmin` para ejecutar las consultas del siguiente fichero, que rellenan con datos las tablas de la base de datos "`dwes`". Esta información la utilizaremos en los próximos ejercicios.

```
USE `dwes`;

INSERT INTO `tienda` (`cod`, `nombre`, `tlf`) VALUES
(1, 'CENTRAL', '600100100'),
(2, 'SUCURSAL1', '600100200'),
(3, 'SUCURSAL2', NULL);

INSERT INTO `familia` (`cod`, `nombre`) VALUES
('CAMARA', 'Cámaras digitales'),
('CONSOL', 'Consolas'),
('EBOOK', 'Libros electrónicos'),
('IMPRES', 'Impresoras'),
('MEMFLA', 'Memorias flash'),
('MP3', 'Reproductores MP3'),
('MULTIF', 'Equipos multifunción'),
('NETBOK', 'Netbooks'),
('ORDENA', 'Ordenadores'),
('PORTAT', 'Ordenadores portátiles'),
('ROUTER', 'Routers'),
('SAI', 'Sistemas de alimentación ininterrumpida'),
('SOFTWA', 'Software'),
('TV', 'Televisores'),
('VIDEOC', 'Videocámaras');

INSERT INTO `producto` (`cod`, `nombre`, `nombre_corto`, `descripcion`, `PVP`, `familia`)
VALUES
('3DSNG', NULL, 'Nintendo 3DS negro', 'Consola portátil de Nintendo que permitirá disfrutar de
efectos 3D sin necesidad de gafas especiales, e incluirá retrocompatibilidad con el software
de DS y de DSi.', '270.00', 'CONSOL'),
('ACERAX3950', NULL, 'Acer AX3950 I5-650 4GB 1TB W7HP', 'Características:\r\n\r\nSistema
Operativo : Windows® 7 Home Premium Original\r\n\r\nProcesador / Chipset\r\nNúmero de Ranuras
PCI: 1\r\nFabricante de Procesador: Intel\r\nTipo de Procesador: Core i5\r\nModelo de
Procesador: i5-650\r\nNúcleo de Procesador: Dual-core\r\nVelocidad de Procesador: 3,20
GHz\r\nCaché: 4 MB\r\nVelocidad de Bus: No aplicable\r\nVelocidad HyperTransport: No
aplicable\r\nInterconexión QuickPathNo aplicable\r\nProcesamiento de 64 bits: Sí\r\nHyper-
ThreadingSí\r\nFabricante de Chipset: Intel\r\nModelo de Chipset: H57
Express\r\n\r\nMemoria\r\nMemoria Estándar: 4 GB\r\nMemoria Máxima: 8 GB\r\nTecnología de la
```

```
Memoria: DDR3 SDRAM\r\nEstándar de Memoria: DDR3-1333/PC3-10600\r\nNúmero de Ranuras de Memoria (Total): 4\r\nLector de tarjeta memoria: Sí\r\nSoporte de Tarjeta de Memoria: Tarjeta CompactFlash (CF)\r\nSoporte de Tarjeta de Memoria: MultiMediaCard (MMC)\r\nSoporte de Tarjeta de Memoria: Micro Drive\r\nSoporte de Tarjeta de Memoria: Memory Stick PRO\r\nSoporte de Tarjeta de Memoria: Memory Stick\r\nSoporte de Tarjeta de Memoria: CF+\r\nSoporte de Tarjeta de Memoria: Tarjeta Secure Digital (SD)\r\n\r\nStorage\r\nCapcidad Total del Disco Duro: 1 TB\r\nRPM de Disco Duro: 5400\r\nTipo de Unidad Óptica: Grabadora DVD\r\nCompatibilidad de Dispositivo Óptico: DVD-RAM/±R/±RW\r\nCompatibilidad de Medios de Doble Capa: Sí', '410.00', 'ORDENA'),
('ARCLPMP32GBN', NULL, 'Archos Clipper MP3 2GB negro', 'Características:\r\n\r\nAlmacenamiento Interno Disponible en 2 GB*\r\nCompatibilidad Windows o Mac y Linux (con soporte para almacenamiento masivo)\r\nInterfaz para ordenador USB 2.0 de alta velocidad\r\nBattería2 11 horas música\r\nReproducción Música3 MP3\r\nMedidas Dimensiones: 52mm x 27mm x 12mm, Peso: 14 Gr', '26.70', 'MP3'),
('BRAVIA2BX400', NULL, 'Sony Bravia 32IN FULLHD KDL-32BX400', 'Características:\r\n\r\nFull HD: Vea deportes películas y juegos con magníficos detalles en alta resolución gracias a la resolución 1920x1080.\r\n\r\nHDMI®: 4 entradas (3 en la parte posterior, 1 en el lateral)\r\n\r\nUSB Media Player: Disfrute de películas, fotos y música en el televisor.\r\n\r\nSintonizador de TV HD MPEG-4 AVC integrado: olvídese del codificador y acceda a servicios de TV que incluyen canales HD con el sintonizador DVB-T y DVB-C integrado con decodificador MPEG4 AVC (dependiendo del país y sólo con operadores compatibles)\r\n\r\nSensor de luz: ajusta automáticamente el brillo según el nivel de la iluminación ambiental para que pueda disfrutar de una calidad de imagen óptima sin consumo innecesario de energía.\r\n\r\nBRAVIA Sync: controle su sistema de ocio doméstico entero con un mismo mando a distancia universal que le permite reproducir contenidos o ajustar la configuración de los dispositivos compatibles con un solo botón.\r\n\r\nBRAVIA ENGINE 2: experimente colores y detalles de imagen increíblemente nítidos y definidos. \r\n\r\nLive Colour™: seleccione entre cuatro modos: desactivado, bajo, medio y alto, para ajustar el color y obtener imágenes vivas y una calidad óptima. \r\n\r\n24p True Cinema™: reproduzca una auténtica experiencia cinemática y disfrute de películas exactamente como el director las concibió a 24 fotogramas por segundo.', '356.90', 'TV'),
('EEEPC1005PXD', NULL, 'Asus EEEPC 1005PXD N455 1 250 BL', 'Características:\r\nProcesador: 1660 MHz, N455, Intel Atom, 0.5 MB. \r\nMemoria: 1024 MB, 2 GB, DDR3, SO-DIMM, 1 x 1024 MB. \r\nAccionamiento de disco: 2.5 ", 250 GB, 5400 RPM, \r\nSerial ATA, Serial ATA II, 250 GB. \r\nMedios de almacenaje: MMC, SD, SDHC. \r\nExhibición: 10.1 ", 1024 x 600 Pixeles, LCD TFT. \r\nCámara fotográfica: 0.3 MP. \r\nRed: 802.11 b/g/n, 10, 100 Mbit/s, \r\nFast Ethernet. \r\nAudio: HD. \r\nSistema operativo/software: Windows 7 Starter. \r\nColor: Blanco. \r\nContro de energía: 8 MB/s, Litio-Ion, 6 piezas, 2200 mAh, 48 W. \r\nPeso y dimensiones: 1270 g, 178 mm, 262 mm, 25.9 mm, 36.5 mm', '245.40', 'NETBOK'),
('HPMIN1103120', NULL, 'HP Mini 110-3120 10.1LED N455 1GB 250GB W7S negro', 'Características:\r\nSistema operativo instalado \r\nWindows® 7 Starter original 32 bits \r\n\r\nProcesador \r\nProcesador Intel® Atom™ N4551,66 GHz, Cache de nivel 2, 512 KB \r\n\r\nChipset NM10 Intel® + ICH8m \r\n\r\nMemoria \r\nDDR2 de 1 GB (1 x 1024 MB) \r\nMemoria máxima \r\nAdmite un máximo de 2 GB de memoria DDR2 \r\n\r\nRanuras de memoria \r\n1 ranura de memoria accesible de usuario \r\n\r\nUnidades internas \r\nDisco duro SATA de 250 GB (5400 rpm) \r\n\r\nGráficos \r\nTamaño de pantalla (diagonal) \r\nPantalla WSVGA LED HP Antirreflejos de 25,6 cm (10,1") en diagonal \r\n\r\nResolución de la pantalla \r\n1024 x 600 ', '270.00', 'NETBOK'),
('IXUS115HSAZ', NULL, 'Canon Ixus 115HS azul', 'Características:\r\nHS System (12,1 MP) \r\nZoom 4x, 28 mm. IS Óptico \r\nCuerpo metálico estilizado \r\nPantalla LCD PureColor II G de 7,6 cm (3,0") \r\nFull HD. IS Dinámico. HDMI \r\nModo Smart Auto (32 escenas) ', '196.70', 'CAMARA'),
('KSTDT101G2', NULL, 'Kingston DataTraveler 16GB DT101G2 USB2.0 negro', 'Características:\r\nCapacidades — 16GB\r\nDimensiones — 2.19" x 0.68" x 0.36" (55.65mm x 17.3mm x 9.05mm)\r\nTemperatura de Operación — 0° hasta 60° C / 32° hasta 140° F\r\nTemperatura de Almacenamiento — -20° hasta 85° C / -4° hasta 185° F\r\nSimple — Solo debe conectarlo a un puerto USB y está listo para ser utilizado\r\nPractico — Su diseño sin tapa giratorio, protege el conector USB; sin tapa que perder\r\nGarantizado — Cinco años de garantía', '19.20', 'MEMFLA'),
('KSTDTG332GBR', NULL, 'Kingston DataTraveler G3 32GB rojo', 'Características:\r\n\r\nTipo de producto Unidad flash USB\r\nCapacidad almacenamiento32GB\r\nAnchura 58.3 mm\r\nProfundidad 23.6 mm\r\nAltura 9.0 mm\r\nPeso 12 g\r\nColor incluido RED\r\nTipo de interfaz USB', '40.00', 'MEMFLA'),
('KSTMSDHC8GB', NULL, 'Kingston MicroSDHC 8GB', 'Kingston tarjeta de memoria flash 8 GB microSDHC\r\nÍndice de velocidad Class 4\r\nCapacidad almacenamiento 8 GB\r\nFactor de forma MicroSDHC\r\nAdaptador de memoria incluido Adaptador microSDHC a SD\r\nGarantía del fabricante Garantía limitada de por vida', '10.20', 'MEMFLA'),
('LEGRIAFS306', NULL, 'Canon Legria FS306 plata', 'Características:\r\n\r\nGrabación en tarjeta de memoria SD/SDHC \r\nLa cámara de vídeo digital de Canon más pequeña nunca vista \r\nInstantánea de Vídeo (Video Snapshot) \r\nZoom Avanzado de 41x \r\nGrabación Dual (Dual Shot) \r\nEstabilizador de la Imagen con Modo Dinámico \r\nPre grabación (Pre REC) \r\nSistema 16:9 de alta resolución realmente panorámico \r\nBatería inteligente y Carga Rápida \r\nCompatible con grabador de DVD DW-100 \r\nSISTEMA DE VÍDEO\r\nSoporte de grabación: Tarjeta de memoria extraíble (SD/SDHC)\r\nTiempo máximo de grabación: Variable, dependiendo del tamaño de la tarjeta de memoria.\r\nTarjeta SDHC de 32 GB: 20 horas 50 minutos', '175.00', 'VIDEOC'),
('LGM237WDP', NULL, 'LG TDT HD 23 M237WDP-PC FULL HD', 'Características:\r\n\r\nGeneral\r\nTamaño (pulgadas): 23\r\nPantalla LCD: Sí\r\nFormato:
```

16:9\r\nResolución: 1920 x 1080\r\nFull HD: Sí\r\nBrillo (cd/m2): 300\r\nRatio Contraste: 50.000:1\r\nTiempo Respuesta (ms): 5\r\nÁngulo Visión (°): 170\r\nNúmero Colores (Millones): 16.7\r\n\r\nTV\r\nTDT: TDT HD\r\nConexiones\r\nD-Sub: Sí\r\nDVI-D: Sí\r\nHDMI: Sí\r\nEuroconector: Sí\r\nSalida auriculares: Sí\r\nEntrada audio: Sí\r\nUSB Servicio: Sí\r\nRS-232C Servicio: Sí\r\nPCMCIA: Sí\r\nSalida óptico: Sí', '186.00', 'TV'),
('LJPROP1102W', NULL, 'HP Laserjet Pro Wifi P1102W', 'Impesora laserjet P1102W es facil de instalar y usar, ademas de que te ayudara a ahorrar energia y recursos. \r\nOlviadte de los cables y disfura de la libertad que te proporcina su tecnologia WIFI, imprime facilmente desdes cualquier de tu oficina. \r\n\r\nFormato máximo aceptado A4 A2 No\r\nA3 NoA4 Si\r\nA5 SiA6 Si\r\nB5 SiB6 Si\r\nSobres C5 (162 x 229 mm) SiSobres C6 (114 x 162 mm) Si', '99.90', 'IMPRES'),
('OPTIOLS1100', NULL, 'Pentax Optio LS1100', 'La LS1100 con funda de transporte y tarjeta de memoria de 2GB incluidas \r\nes la compacta digital que te llevarás a todas partes. \r\nEsta cámara diseñada por Pentax incorpora un sensor CCD de 14,1 megapíxeles y un objetivo gran angular de 28 mm.\r\n', '104.80', 'CAMARA'),
('PAPYRE62GB', NULL, 'Lector ebooks Papyre6 con SD2GB + 500 ebooks', 'Marca Papyre \r\nModelo Papyre 6.1 \r\nUso Lector de libros electrónicos \r\nTecnología e-ink (tinta electrónica, Vizplez) \r\nCPU Samsung Am9 200MHz \r\nMemoria - Interna: 512MB \r\n- Externa: SD/SDHC (hasta 32GB) \r\nFormatos PDF, RTF, TXT, DOC, HTML, MP3, CHM, ZIP, FB2, Formatos de imagen \r\nPantalla 6" (600x800px), blanco y negro, 4 niveles de grises ', '205.50', 'EBOOK'),
('PBELLI810323', NULL, 'Packard Bell I8103 23 I3-550 4G 640GB NVIDIAG210', 'Características:\r\n\r\nCPU CHIPSET\r\n\r\nProcesador : Ci3-550\r\nNorthBridge : Intel H57\r\n\r\nMEMORIA\r\nMemoria Rma : Ddr3 4096 MB\r\n\r\nDISPOSITIVOS DE ALMACENAMIENTO\r\nDisco Duro: 640Gb 7200 rpm\r\nÓptico : Slot Load siper multi Dvdrw\r\nLector de Tarjetas: 4 in 1 (XD, SD, HC, MS, MS PRO, MMC)\r\n\r\ndispositivos gráficos\r\nMonitor: 23 fHD\r\nTarjeta Gráfica: Nvidia G210M D3 512Mb\r\nMemoria Máxima: Hasta 1918Mb\r\n\r\nAUDIO\r\nAudio Out: 5.1 Audio Out\r\nAudio In: 1 jack\r\nHeasphone in: 1x jack\r\nAltavoces: Stereo\r\n\r\nACCESORIOS\r\nTeclado: Teclado y ratón inalámbrico\r\nMando a distancia: EMEA Win7 WMC\r\n\r\n\r\nCOMUNICACIONES\r\nWireless: 802.11 b/g/n mini card \r\nTarjeta de Red: 10/100/1000 Mbps\r\nBluetooth: Bluethoot\r\nWebcam: 1Mpixel Hd (1280x720)\r\nTv tuner: mCARD/SW/ DVB-T\r\n\r\nMONITOR\r\nTamaño: 23"\r\ncontraste: 1000:1\r\nTiempo de respuesta: 5MS\r\nResolución: 1920 X 1080\r\n\r\nPUERTOS E/S\r\nUsb 2.0 : 6\r\nMini Pci-e : 2\r\nEsata: 1\r\n\r\nSISTEMA OPERATIVO\r\nO.S: Microsoft Windows 7 Premium', '761.80', 'ORDENA'),
('PIXMAIP4850', NULL, 'Canon Pixma IP4850', 'Características:\r\n\r\nTipo: chorro de tinta cartuchos independientes\r\nConexión: Hi-Speed USB\r\nPuerto de impresión directa desde camaras\r\nResolución máxima: 9600x2400 ppp\r\nVelocidad impresión: 11 ipm (negro) / 9.3 ipm (color)\r\nTamaño máximo papel: A4\r\nBandeja entrada: 150 hojas\r\nDimensiones: 43.1 cm x 29.7 cm x 15.3 cm', '97.30', 'IMPRES'),
('PIXMAMP252', NULL, 'Canon Pixma MP252', 'Características:\r\n\r\nFunciones: Impresora, Escáner , Copiadora\r\nConexión: USB 2.0\r\nDimensiones:444 x 331 x 155 mm\r\nPeso: 5,8 Kg\r\n\r\nIMPRESORA\r\nResolución máxima: 4800 x 1200 ppp\r\nVelocidad de impresión:\r\nNegro/color: 7,0 ipm / 4,8 ipm\r\nTamaño máximo papel: A4\r\nCARTUCHOS\r\nNegro: PG-510 / PG-512\r\nColor: CL-511 / CL-513\r\n\r\nESCANER\r\nResolución máxima: 600 x 1200 ppp (digital: 19200 x 19200)\r\nProfundidad de color: 48/24 bits\r\nArea máxima de escaneado: A4\r\n\r\nCOPIA\r\nTiempo salida 1ª copia: aprox 39 seg.', '41.60', 'MULTIF'),
('PS3320GB', NULL, 'PS3 con disco duro de 320GB', 'Este Pack Incluye:\r\n- La consola Playstation 3 Slim Negra 320GB\r\n- El juego Killzone 3\r\n', '380.00', 'CONSOL'),
('PWSHTA3100PT', NULL, 'Canon Powershot A3100 plata', 'La cámara PowerShot A3100 IS, inteligente y compacta, presenta la calidad de imagen de Canon en un cuerpo\r\ncompacto y ligero para capturar fotografías sin esfuerzo; es tan fácil como apuntar y disparar.\r\nCaracterísticas:\r\n12,1 MP \r\nZoom óptico 4x con IS \r\nPantalla LCD de 6,7 cm (2,7") ', '101.40', 'CAMARA'),
('SMSGCLX3175', NULL, 'Samsung CLX3175', 'Características:\r\n\r\nFunción: Impresión color, copiadora, escáner\r\nImpresión \r\nVelocidad (Mono)Hasta 16 ppm en A4 (17 ppm en Carta)\r\nVelocidad (Color)Hasta 4 ppm en A4 (4 ppm en Carta)\r\nSalida de la Primer Página (Mono)Menos de 14 segundos (Desde el Modo Listo)\r\nResoluciónHasta 2400 x 600 dpi de salida efectiva\r\nSalida de la Primer Página (Color)Menos de 26 segundos (Dese el Modo Listo)\r\nDuplexManual\r\nEmulaciónSPL-C (Lenguaje de color de impresión SAMSUNG)\r\n\r\nCopiado \r\nSalida de la Primer Página (Mono)18 segundos\r\nMulticopiado1 ~ 99\r\nZoom25 ~ 400 %\r\nFunciones de CopiadoCopia ID, Clonar Copia, Copia N-UP, Copiar Poster\r\nResoluciónTexto, Texto / Foto, Modo Revista: hasta 600 x 600 ppp, Modo Foto: Hasta 1200 x 1200 ppp\r\nVelocidad (Mono)Hasta 17 ppm en Carta (16 ppm en A4)\r\nVelocidad (Color)Hasta 4 ppm en Carta (4 ppm en A4)\r\nSalida de la Primer Página (Color)45 segundos\r\n\r\nEscaneado \r\n\r\nCompatibilidadNorma TWAIN, Norma WIA (Windows2003 / XP / Vista)\r\nMétodoEscáner plano color\r\nResolución (Óptica)1200 x 1200 dpi\r\nResolución (Mejorada)4800 x 4800 dpi\r\nEscaneado a Escanear a USB / Carpeta', '190.00', 'MULTIF'),
('SMSN150101LD', NULL, 'Samsung N150 10.1LED N450 1GB 250GB BAT6 BT W7 R', 'Características:\r\n\r\nSistema Operativo Genuine Windows® 7 Starter \r\n\r\nProcesador Intel® ATOM Processor N450 (1.66GHz, 667MHz, 512KB) \r\n\r\nChipset Intel® NM10\r\n\r\nMemoria del Sistema 1GB (DDR2 / 1GB x 1) Ranura de Memoria 1 x SODIMM \r\n\r\nPantalla LCD 10.1" WSVGA (1024 x 600), Non-Gloss, LED Back Light Gráficos \r\n\r\nProcesador Gráfico Intel® GMA 3150 DVMT \r\nMemoria Gráfica Shared Memory (Int. Grahpic) \r\n\r\nMultimedia \r\nSonido HD (High Definition) Audio \r\nCaracterísticas de Sonido SRS 3D Sound Effect \r\nAltavoces 3W Stereo Speakers (1.5W x 2) \r\nCámara Integrada Web Camera \r\n\r\nAlmacenamiento \r\nDisco duro 250GB SATA (5400 rpm S-ATA) \r\n\r\nConectividad\r\nWired Ethernet LAN (RJ45) 10/100 LAN \r\nWireless LAN 802.11 b/g/N\r\n\r\nBluetooth Bluetooth 3.0 High Speed \r\n\r\nI/O Port \r\nVGA \r\nHeadphone-out\r\nMic-in\r\nInternal Mic\r\nUSB (Chargable USB included) 3 x USB

```
2.0 \r\nMulti Card Slot 4-in-1 (SD, SDHC, SDXC, MMC)\r\nDC-in (Power Port)\r\n\r\nTipo de Teclado 84 keys \r\nTouch Pad, Touch Screen Touch Pad (Scroll Scope, Flat Type) \r\n\r\nSeguridad\r\nRecovery Samsung Recovery Solution \r\nVirus McAfee Virus Scan (trial version) \r\nSeguridad BIOS Boot Up Password / HDD Password \r\nBloqueo Kensington Lock Port \r\n\r\nBatería \r\nAdaptador 40 Watt Batería \r\n6 Cell Dimensiones ', '260.60', 'NETBOK'),
('SMSSMXC200PB', NULL, 'Samsung SMX-C200PB EDC ZOOM 10X', 'Características:\r\n\r\nSensor de Imagen Tipo 1 / 6” 800K pixel CCD\r\n\r\nLente Zoom Óptico 10 x optico\r\n\r\nCaracterísticas Grabación Vídeo Estabilizador de Imagen Hiper estabilizador de imagen digital\r\n\r\nInterfaz Tarjeta de Memoria Ranura de Tarjeta SDHC / SD', '127.20', 'VIDEOC'),
('STYLUSSX515W', NULL, 'Epson Stylus SX515W', 'Características:\r\n\r\nResolución máxima5760 x 1440 DPI\r\nVelocidad de la impresión\r\nVelocidad de impresión (negro, calidad normal, A4)36 ppm\r\nVelocidad de impresión (color, calidad normal, A4)36 ppm\r\n\r\nTecnología de la impresión\r\nTecnología de impresión inyección de tinta\r\nNúmero de cartuchos de impresión4 piezas\r\nCabeza de impresoraMicro Piezo\r\n\r\nExploración\r\nResolución máxima de escaneado2400 x 2400 DPI\r\nEscaner color: si\r\nTipo de digitalización Escáner plano\r\nEscanaer integrado: si\r\nTecnología de exploración CIS\r\nWLAN, conexión: si', '77.50', 'MULTIF'),
('TSSD16GBC10J', NULL, 'Toshiba SD16GB Class10 Jewel Case', 'Características:\r\n\r\nDensidad: 16 GB\r\nPINs de conexión: 9 pins\r\nInterfaz: Tarjeta de memoria SD standard compatible\r\nVelocidad de Escritura: 20 MBytes/s* \r\nVelocidad de Lectura: 20 MBytes/s*\r\nDimensiones: 32.0 mm (L) × 24.0 mm (W) × 2.1 mm (H)\r\nPeso: 2g\r\nTemperatura: -25°C a +85°C (Recomendada)\r\nHumedad: 30% to 80% RH (sin condensación)', '32.60', 'MEMFLA'),
('ZENMP48GB300', NULL, 'Creative Zen MP4 8GB Style 300', 'Características:\r\n\r\n8 GB de capacidad\r\nAutonomía: 32 horas con archivos MP3 a 128 kbps\r\nPantalla TFT de 1,8 pulgadas y 64.000 colores\r\nFormatos de audio compatibles: MP3, WMA (DRM9), formato Audible 4\r\nFormatos de foto compatibles: JPEG (BMP, TIFF, GIF y PNG\r\nFormatos de vídeo compatibles: AVI transcodificado (Motion JPEG)\r\nEcualizador de 5 bandas con 8 preajustes\r\nMicrófono integrado para grabar voz\r\nAltavoz y radio FM incorporada', '58.90', 'MP3');

INSERT INTO `stock` (`producto`, `tienda`, `unidades`) VALUES
('3DSNG', 1, 2),
('3DSNG', 2, 1),
('ACERAX3950', 1, 1),
('ARCLPMP32GBN', 2, 1),
('ARCLPMP32GBN', 3, 2),
('BRAVIA2BX400', 3, 1),
('EEEPC1005PXD', 1, 2),
('EEEPC1005PXD', 2, 1),
('HPMIN1103120', 2, 1),
('HPMIN1103120', 3, 2),
('IXUS115HSAZ', 2, 2),
('KSTDT101G2', 3, 1),
('KSTDTG332GBR', 2, 2),
('KSTMSDHC8GB', 1, 1),
('KSTMSDHC8GB', 2, 2),
('KSTMSDHC8GB', 3, 2),
('LEGRIAFS306', 2, 1),
('LGM237WDP', 1, 1),
('LJPROP1102W', 2, 2),
('OPTIOLS1100', 1, 3),
('OPTIOLS1100', 2, 1),
('PAPYRE62GB', 1, 2),
('PAPYRE62GB', 3, 1),
('PBELLI810323', 2, 1),
('PIXMAIP4850', 2, 1),
('PIXMAIP4850', 3, 2),
('PIXMAMP252', 2, 1),
('PS3320GB', 1, 1),
('PWSHTA3100PT', 2, 2),
('PWSHTA3100PT', 3, 2),
('SMSGCLX3175', 2, 1),
('SMSN150101LD', 3, 1),
('SMSSMXC200PB', 2, 1),
('STYLUSSX515W', 1, 1),
('TSSD16GBC10J', 3, 2),
('ZENMP48GB300', 1, 3),
('ZENMP48GB300', 2, 2),
('ZENMP48GB300', 3, 2);
```

Selecciona en el panel de la izquierda la base de datos, y utilizando el menú SQL pega todo el texto del fichero (las consultas a ejecutar) y pulsa el botón "Continuar" para ejecutarlas.

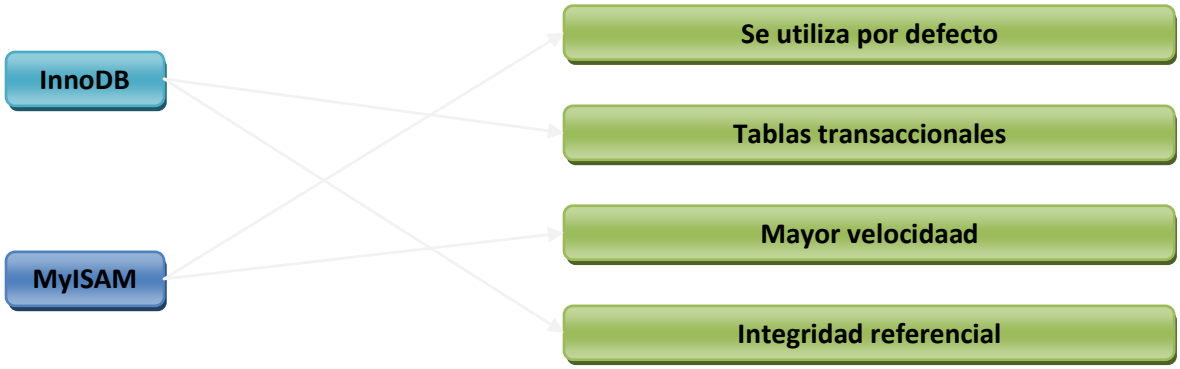
Intenta resolver la siguiente actividad relativa a las bases de datos MySQL.

1. Permite ejecutar sentencias SQL.
2. Muestra información sobre bases de datos y tablas.

3. En línea de comandos, específica para tareas de administración.
4. Programada en lenguaje PHP.
5. Comando de mysql para seleccionar una base de datos.
6. Usuario administrador de MySQL, por defecto.

1 - MySQL 2 - mysqlshow 3 - mysqladmin 4 - phpmyadmin 5 - use 6 - root

Relaciona los motores de almacenamiento de MySQL con sus características principales



3.- Utilización de bases de datos MySQL en PHP.

Como ya viste, existen dos formas de comunicarse con una base de datos desde PHP: utilizar una extensión nativa programada para un SGBD concreto, o utilizar una extensión que soporte varios tipos de bases de datos. Tradicionalmente las conexiones se establecían utilizando la extensión nativa **mysql**. Esta extensión se mantiene en la actualidad para dar soporte a las aplicaciones ya existentes que la utilizan, pero no se recomienda utilizarla para desarrollar nuevos programas. Lo más habitual es elegir entre **MySQLi** (extensión nativa) y **PDO**.

Con cualquiera de ambas extensiones, podrás realizar acciones sobre las bases de datos como:

- ✓ Establecer conexiones.
- ✓ Ejecutar sentencias SQL.
- ✓ Obtener los registros afectados o devueltos por una sentencia SQL.
- ✓ Emplear transacciones.
- ✓ Ejecutar procedimientos almacenados.
- ✓ Gestionar los errores que se produzcan durante la conexión o en el establecimiento de la misma.

PDO y MySQLi (y también la antigua extensión mysql) utilizan un **driver de bajo nivel** para comunicarse con el servidor MySQL. Hasta hace poco el único driver disponible para realizar esta función era **libmysql**, que no estaba optimizado para ser utilizado desde PHP. A partir de la versión 5.3, PHP viene preparado para utilizar también un nuevo driver mejorado para realizar esta función, el Driver Nativo de MySQL, **mysqlnd**.

3.1.- Extensión MySQLi.

Esta extensión se desarrolló para aprovechar las ventajas que ofrecen las versiones 4.1.3 y posteriores de MySQL, y viene incluida con PHP a partir de la versión 5. Ofrece un interface de programación dual, pudiendo accederse a las funcionalidades de la extensión utilizando objetos o funciones de forma indiferente. Por ejemplo, para establecer una conexión con un servidor MySQL y consultar su versión, podemos utilizar cualquiera de las siguientes formas:

```
// utilizando constructores y métodos de la programación orientada a objetos
$conexion = new mysqli('localhost', 'usuario', 'contraseña', 'base_de_datos');
print conexion->server info;

// utilizando llamadas a funciones
$conexion = mysqli connect('localhost', 'usuario', 'contraseña', 'base de datos');
print mysqli_get_server_info($conexion);
```

En ambos casos, la variable `$conexion` es de tipo objeto. La utilización de los métodos y propiedades que aporta la clase **mysqli** normalmente produce un código más corto y legible que si utilizas llamadas a funciones.

Toda la información relativa a la instalación y utilización de la extensión, incluyendo las funciones y métodos propios de la extensión, se puede consultar en el manual de PHP.
http://es.php.net/manual/es/book.mysqli.php

Entre las mejoras que aporta a la antigua extensión mysql, figuran:
- ✓ Interface orientado a objetos.
- ✓ Soporte para transacciones.
- ✓ Soporte para consultas preparadas.
- ✓ Mejores opciones de depuración.

Como ya viste en la primera unidad, las opciones de configuración de PHP se almacenan en el fichero `php.ini`. En este fichero hay una sección específica para las opciones de configuración propias de cada extensión. Entre las opciones que puedes configurar para la extensión MySQLi están:
- ✓ **`mysqli.allow_persistent`.** Permite crear conexiones persistentes.
- ✓ **`mysqli.default_port`**. Número de puerto TCP predeterminado a utilizar cuando se conecta al servidor de base de datos.
- ✓ **`mysqli.reconnect`**. Indica si se debe volver a conectar automáticamente en caso de que se pierda la conexión.
- ✓ **`mysqli.default_host`**. Host predeterminado a usar cuando se conecta al servidor de base de datos.
- ✓ **`mysqli.default_user`**. Nombre de usuario predeterminado a usar cuando se conecta al servidor de base de datos.
- ✓ **`mysqli.default_pw`**. Contraseña predeterminada a usar cuando se conecta al servidor de base de datos.

En la documentación de PHP se incluye una lista completa de las directivas relacionadas con la extensión MySQLi que se pueden utilizar en php.ini.
http://es2.php.net/manual/es/mysqli.configuration.php

¿Qué interface o interfaces de programación admite la extensión MySQLi?

○ Orientado a objetos únicamente.

◉ **Dos interfaces de programación: procedimental y orientado a objetos.**

aunque el interface orientado a objetos es una mejora sobre la antigua extensión mysql, MySQLi mantiene también de forma paralela un interface procedimental.

3.1.1.- Establecimiento de conexiones.

Para poder comunicarte desde un programa PHP con un servidor MySQL, el primer paso es establecer una conexión. Toda comunicación posterior que tenga lugar, se hará utilizando esa conexión.
Si utilizas la extensión MySQLi, establecer una conexión con el servidor significa crear una instancia de la **clase mysqli**. El constructor de la clase puede recibir seis parámetros, todos opcionales, aunque lo más habitual es utilizar los cuatro primeros:
- ✓ El nombre o dirección IP del servidor MySQL al que te quieres conectar.
- ✓ Un nombre de usuario con permisos para establecer la conexión.
- ✓ La contraseña del usuario.
- ✓ El nombre de la base de datos a la que conectarse.
- ✓ El número del puerto en que se ejecuta el servidor MySQL.
- ✓ El socket o la tubería con nombre (named pipe) a usar.

Si utilizas el constructor de la clase, para conectarte a la base de datos "dwes" puedes hacer:

```
// utilizando el constructor de la clase
$dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
```

Aunque también tienes la opción de primero crear la instancia, y después utilizar el método `connect` para establecer la conexión con el servidor:

```
// utilizando el método connect
$dwes = new mysqli();
$dwes->connect('localhost', 'dwes', 'abc123.', 'dwes');
```

Por el contrario, utilizando el interface procedimental de la extensión:

```
// utilizando llamadas a funciones
$dwes = mysqli_connect('localhost', 'dwes', 'abc123.', 'dwes');
```

Es importante verificar que la conexión se ha establecido correctamente. Para comprobar el error, en caso de que se produzca, puedes usar las siguientes propiedades (o funciones equivalentes) de la clase mysqli:

- ✓ **`connect_errno`** (o la función **`mysqli_connect_errno`**) devuelve el número de error o `null` si no se produce ningún error.
- ✓ **`connect_error`** (o la función **`mysqli_connect_error`**) devuelve el mensaje de error o `null` si no se produce ningún error.

Por ejemplo, el siguiente código comprueba el establecimiento de una conexión con la base de datos "`dwes`" y finaliza la ejecución si se produce algún error:

```
@ $dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
$error = $dwes->connect errno;
if ($error != null) {
   echo "<p>Error $error conectando a la base de datos: $dwes->connect_error</p>";
   exit();
}
```

En PHP, como veremos posteriormente con más detalle, puedes anteponer a cualquier expresión el operador de control de errores @ para que se ignore cualquier posible error que pueda producirse al ejecutarla.

http://es.php.net/manual/es/language.operators.errorcontrol.php

Si una vez establecida la conexión, quieres cambiar la base de datos puedes usar el método **`select_db`** (o la función **`mysqli_select_db`** de forma equivalente) para indicar el nombre de la nueva.

```
// utilizando el método connect
$dwes->select_db('otra_bd');
```

Una vez finalizadas las tareas con la base de datos, utiliza el método **`close`** (o la función **`mysqli_close`**) para cerrar la conexión con la base de datos y liberar los recursos que utiliza.

```
$dwes->close();
```

3.1.2.- Ejecución de consultas.

La forma más inmediata de ejecutar una consulta, si utilizas esta extensión, es el método **`query`**, equivalente a la función **`mysqli_query`**. Si se ejecuta una consulta de acción que no devuelve datos (como una sentencia SQL de tipo **`UPDATE`**, **`INSERT`** o **`DELETE`**), la llamada devuelve `true` si se ejecuta correctamente o `false` en caso contrario. El número de registros afectados se puede obtener con la propiedad **`affected_rows`** (o con la función **`mysqli_affected_rows`**).

```
@ $dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
$error = $dwes->connect errno;
if ($error == null) {
   $resultado = $dwes->query('DELETE FROM stock WHERE unidades=0');
   if ($resultado) {
```

```
        print "<p>Se han borrado $dwes->affected_rows registros.</p>";
    }
    $dwes->close();
}
```

En el caso de ejecutar una sentencia SQL que sí devuelva datos (como un `SELECT`), éstos se devuelven en forma de un objeto resultado (de la clase **`mysqli_result`**). En el punto siguiente verás cómo se pueden manejar los resultados obtenidos.

El método query tiene un parámetro opcional que afecta a cómo se obtienen internamente los resultados, pero no a la forma de utilizarlos posteriormente. En la opción por defecto, **`MYSQLI_STORE_RESULT`**, los resultados se recuperan todos juntos de la base de datos y se almacenan de forma local. Si cambiamos esta opción por el valor **`MYSQLI_USE_RESULT`**, los datos se van recuperando del servidor según se vayan necesitando.

```
$resultado = $dwes->query('SELECT producto, unidades FROM stock', MYSQLI_USE_RESULT);
```

Otra forma que puedes utilizar para ejecutar una consulta es el método **real_query** (o la función **mysqli_real_query**), que siempre devuelve true o false según se haya ejecutado correctamente o no. Si la consulta devuelve un conjunto de resultados, se podrán recuperar de forma completa utilizando el método **store_result**, o según vaya siendo necesario gracias al método **use_result**.

http://es.php.net/manual/es/mysqli.real-query.php

Es importante tener en cuenta que los resultados obtenidos se almacenarán en memoria mientras los estés usando. Cuando ya no los necesites, los puedes liberar con el método **`free`** de la clase **`mysqli_result`** (o con la función **`mysqli_free_result`**):

```
$resultado-&gt;free();
```

De las dos opciones que admite el método query, MYSQLI_STORE_RESULT y MYSQLI_USE_RESULT, ¿qué opción será recomendable utilizar para ejecutar una consulta que devuelva una enorme cantidad de datos?

- ○ MYSQLI_STORE_RESULT.
- ◉ **MYSQLI_USE_RESULT.**

Con esta opción se van obteniendo los datos del servidor a medida que se vayan necesitando. Si utilizaras la otra opción, los datos tendrían que transferirse todos juntos al ejecutar la consulta

3.1.3.- Transacciones.

Como ya comentamos, si necesitas utilizar transacciones deberás asegurarte de que estén soportadas por el motor de almacenamiento que gestiona tus tablas en MySQL. Si utilizas X, por defecto cada consulta individual se incluye dentro de su propia transacción. Puedes gestionar este comportamiento con el método **`autocommit`** (función **`mysqli_autocommit`**).

```
$dwes->autocommit(false);   // deshabilitamos el modo transaccional automático
```

Al deshabilitar las transacciones automáticas, las siguientes operaciones sobre la base de datos iniciarán una transacción que deberás finalizar utilizando:

- ✓ **`commit`** (o la función **`mysqli_commit`**). Realizar una operación "`commit`" de la transacción actual, devolviendo `true` si se ha realizado correctamente o `false` en caso contrario.
- ✓ **`rollback`** (o la función **`mysqli_rollback`**). Realizar una operación "`rollback`" de la transacción actual, devolviendo `true` si se ha realizado correctamente o `false` en caso contrario.

```
…
$dwes->query('DELETE FROM stock WHERE unidades=0');  // Inicia una transacción
$dwes->query('UPDATE stock SET unidades=3 WHERE producto="STYLUSSX515W"');
…
```

```
$dwes->commit();  // Confirma los cambios
```

Una vez finalizada esa transacción, comenzará otra de forma automática.

Según la información que figura en la tabla stock de la base de datos dwes, la tienda 1 (CENTRAL) tiene 2 unidades del producto de código 3DSNG y la tienda 3 (SUCURSAL2) ninguno. Suponiendo que los datos son esos (no hace falta que los compruebes en el código), utiliza una transacción para mover una unidad de ese producto de la tienda 1 a la tienda 3.

Deberás hacer una consulta de actualización (para poner unidades=1 en la tienda 1) y otra de inserción (pues no existe ningún registro previo para la tienda 3). Observa el código de la solución. Comprueba que se ejecuta bien solo la primera vez, pues en ejecuciones posteriores ya no es posible insertar la misma fila en la tabla.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejercicio: Transacción con MySQLi -->
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title>Ejercicio: Transacción con MySQLi</title>
   </head>
   <body>
      <?php
         @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes", 3316);
         $error = $dwes->connect_errno;
         if ($error != null) {
            print "<p>Se ha producido el error: $dwes->connect error.</p>";
            exit();
         }
         // Definimos una variable para comprobar la ejecución de las consultas
         $todo bien = true;
         // Iniciamos la transacción
         $dwes->autocommit(false);
         $sql = 'UPDATE stock SET unidades=1 WHERE producto="3DSNG" AND tienda=1';
         if ($dwes->query($sql) != true) $todo bien = false;
         $sql = 'INSERT INTO `stock` (`producto`, `tienda`, `unidades`) VALUES ("3DSNG", 3,
1)';
         if ($dwes->query($sql) != true) $todo_bien = false;
         // Si todo fue bien, confirmamos los cambios
         //  y en caso contrario los deshacemos
         if ($todo bien == true) {
            $dwes->commit();
            print "<p>Los cambios se han realizado correctamente.</p>";
         }else {
            $dwes->rollback();
            print "<p>No se han podido realizar los cambios.</p>";
         }
         $dwes->close();
         unset($dwes);
      ?>
   </body>
</html>
```

En el modo de gestión de transacciones que se utiliza por defecto, ¿es posible revertir los cambios que se aplican al ejecutar una consulta de acción?

(•) **No** ( ) Sí

el modo por defecto indica que se realice un "commit" automático por cada consulta, con lo cual es imposible revertir los cambios que se han realizado.

3.1.4.- Obtención y utilización de conjuntos de resultados.

Ya sabes que al ejecutar una consulta que devuelve datos obtienes un objeto de la clase **mysqli_result**. Esta clase sigue los criterios de ofrecer un interface de programación dual, es decir, una función por cada método con la misma funcionalidad que éste.

Para trabajar con los datos obtenidos del servidor, tienes varias posibilidades:
fetch_array (función **mysqli_fetch_array)**. Obtiene un registro completo del conjunto de resultados y lo almacena en un array. Por defecto el array contiene tanto claves numéricas como asociativas.

Por ejemplo, para acceder al primer campo devuelto, podemos utilizar como clave el número 0 o su nombre indistintamente.

```
$resultado = $dwes->query('SELECT producto, unidades FROM stock WHERE unidades<2');
$stock = $resultado->fetch_array();  // Obtenemos el primer registro
$producto = $stock['producto'];  // O también $stock[0];
$unidades = $stock['unidades'];  // O también $stock[1];
print "<p>Producto $producto: $unidades unidades.</p>";
```

Este comportamiento por defecto se puede modificar utilizando un parámetro opcional, que puede tomar los siguientes valores:

- ✓ **MYSQLI_NUM**. Devuelve un array con claves numéricas.
- ✓ **MYSQLI_ASSOC**. Devuelve un array asociativo.
- ✓ **MYSQLI_BOTH**. Es el comportamiento por defecto, en el que devuelve un array con claves numéricas y asociativas.

fetch_assoc (función **mysqli_fetch_assoc)**. Idéntico a fetch_array pasando como parámetro **MYSQLI_ASSOC**.
fetch_row (función **mysqli_fetch_row)**. Idéntico a fetch_array pasando como parámetro **MYSQLI_NUM**.
fetch_object (función **mysqli_fetch_object)**. Similar a los métodos anteriores, pero devuelve un objeto en lugar de un array. Las propiedades del objeto devuelto se corresponden con cada uno de los campos del registro.

Parar recorrer todos los registros de un array, puedes hacer un bucle teniendo en cuenta que cualquiera de los métodos o funciones anteriores devolverá null cuando no haya más registros en el conjunto de resultados.

```
$resultado = $dwes->query('SELECT producto, unidades FROM stock WHERE unidades<2');
$stock = $resultado->fetch_object();
while ($stock != null) {
   print "<p>Producto $stock->producto: $stock->unidades unidades.</p>";
   $stock = $resultado->fetch_object();
}
```

En el manual de PHP tienes más información sobre los métodos y propiedades de la clase mysqli_result.

http://es.php.net/manual/es/class.mysqli-result.php

Crea una página web en la que se muestre el stock existente de un determinado producto en cada una de las tiendas. Para seleccionar el producto concreto utiliza un cuadro de selección dentro de un formulario en esa misma página. Puedes usar como base los siguientes ficheros.

Plantilla.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

```
  <title>Plantilla para Ejercicios Tema 3</title>
  <link href="dwes.css" rel="stylesheet" type="text/css">
</head>

<body>

<div id="encabezado">
   <h1>Ejercicio: </h1>
   <form id="form_seleccion" action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">
   </form>
</div>

<div id="contenido">
   <h2>Contenido</h2>
</div>

<div id="pie">
</div>
</body>
</html>
```

dwes.css

```
h1 {margin-bottom:0;}
#encabezado {background-color:#ddf0a4;}
#contenido {background-color:#EEEEEE;height:600px;}
#pie {background-color:#ddf0a4;color:#ff0000;height:30px;}
```

Revisa la solución propuesta y verifica que los resultados que obtuviste son correctos.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Conjuntos de datos con MySQLi -->
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title>Ejercicio Tema 3: Conjuntos de resultados en MySQLi</title>
      <link href="dwes.css" rel="stylesheet" type="text/css">
   </head>
   <body>
      <div id="encabezado">
         <h1>Ejercicio: Conjuntos de resultados en MySQLi</h1>
         <form id="form seleccion" action="." method="post">
            <span>Producto: </span>
            <select name="producto">
            <?php
               if (isset($ POST['producto'])) $producto = $ POST['producto'];
               // Rellenamos el desplegable con los datos de todos los productos
               @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes");
               $error = $dwes->connect_errno;
               if ($error == null) {
                  $sql = "SELECT cod, nombre_corto FROM producto";
                  $resultado = $dwes->query($sql);
                  if($resultado) {
                     $row = $resultado->fetch_assoc();
                     while ($row != null) {
                        echo "<option value='${row['cod']}'";
                        // Si se recibió un código de producto lo seleccionamos
                        //  en el desplegable usando selected='true'
                        if (isset($producto) && $producto == $row['cod'])
                           echo " selected='true'";
                        echo ">${row['nombre_corto']}</option>";
                        $row = $resultado->fetch_assoc();
                     }
                     $resultado->close();
                  }
               } else {
                  $mensaje = $dwes->connect_error;
               }
            ?>
            </select>
            <input type="submit" value="Mostrar stock" name="enviar"/>
         </form>
      </div>
      <div id="contenido">
```

```
         <h2>Stock del producto en las tiendas:</h2>
         <?php
            // Si se recibió un código de producto y no se produjo ningún error
            //  mostramos el stock de ese producto en las distintas tiendas
            if ($error == null && isset($producto)) {$sql = <<<SQL
               SELECT tienda.nombre, stock.unidades
               FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
               WHERE stock.producto='$producto'
SQL;
               $resultado = $dwes->query($sql);
               if($resultado) {
                  $row = $resultado->fetch_assoc();
                  while ($row != null) {
                     echo "<p>Tienda ${row['nombre']}: ${row['unidades']} unidades.</p>";
                     $row = $resultado->fetch_assoc();
                  }
                  $resultado->close();
               }
            }
         ?>
      </div>
      <div id="pie">
         <?php
            // Si se produjo algún error se muestra en el pie
            if ($error != null) echo "<p>Se ha producido un error! $mensaje</p>";
            else {
               $dwes->close();
               unset($dwes);
            }
         ?>
      </div>
   </body>
</html>
```

3.1.5.- Consultas preparadas.

Cada vez que se envía una consulta al servidor, éste debe analizarla antes de ejecutarla. Algunas sentencias SQL, como las que insertan valores en una tabla, deben repetirse de forma habitual en un programa. Para acelerar este proceso, MySQL admite consultas preparadas. Estas consultas se almacenan en el servidor listas para ser ejecutadas cuando sea necesario.

Para trabajar con consultas preparadas con la extensión MySQLi de PHP, debes utilizar la clase **mysqli_stmt**. Utilizando el método **stmt_init** de la clase **mysqli** (o la función **mysqli_stmt_init**) obtienes un objeto de dicha clase.

```
$dwes = new mysqli('localhost', 'dwes', 'abc123.', 'dwes');
$consulta = $dwes->stmt_init();
```

Los pasos que debes seguir para ejecutar una consulta preparada son:

- ✓ Preparar la consulta en el servidor MySQL utilizando el método `prepare` (función `mysqli_stmt_prepare`).
- ✓ Ejecutar la consulta, tantas veces como sea necesario, con el método `execute` (función `mysqli_stmt_execute`).
- ✓ Una vez que ya no se necesita más, se debe ejecutar el método `close` (función `mysqli_stmt_close`).

Por ejemplo, para preparar y ejecutar una consulta que inserta un nuevo registro en la tabla familia:

```
$consulta = $dwes->stmt_init();
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES ("TABLET", "Tablet PC")');
$consulta->execute();
$consulta->close();
$dwes->close();
```

El problema que ya habrás observado, es que de poco sirve preparar una consulta de inserción de datos como la anterior, si los valores que inserta son siempre los mismos. Por este motivo las

consultas preparadas admiten parámetros. Para preparar una consulta con parámetros, en lugar de poner los valores debes indicar con un signo de interrogación su posición dentro de la sentencia SQL.

```
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
```

Y antes de ejecutar la consulta tienes que utilizar el método **bind_param** (o la función **mysqli_stmt_bind_param**) para sustituir cada parámetro por su valor. El primer parámetro del método bind_param es una cadena de texto en la que cada carácter indica el tipo de un parámetro, según la siguiente tabla.

Caracteres indicativos del tipo de los parámetros en una consulta preparada.	
Carácter.	**Tipo del parámetro.**
I.	Número entero.
D.	Número real (doble precisión).
S.	Cadena de texto.
B.	Contenido en formato binario (BLOB).

En el caso anterior, si almacenas los valores a insertar en sendas variables, puedes hacer:

```
$consulta = $dwes->stmt_init();
$consulta->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
$cod_producto = "TABLET";
$nombre_producto = "Tablet PC";
$consulta->bind_param('ss', $cod_producto, $nombre_producto);
$consulta->execute();
$consulta->close();
$dwes->close();
```

Cuando uses **bind_param** para enlazar los parámetros de una consulta preparada con sus respectivos valores, deberás usar siempre variables como en el ejemplo anterior. Si intentas utilizar literales, por ejemplo:

```
$consulta->bind_param('ss', 'TABLET', 'Tablet PC');  // Genera un error
```

Obtendrás un error. El motivo es que los parámetros del método bind_param se pasan por referencia. Aprenderás a usar paso de parámetros por referencia en una unidad posterior.

El método **bind_param** permite tener una consulta preparada en el servidor MySQL y ejecutarla tantas veces como quieras cambiando ciertos valores cada vez. Además, en el caso de las consultas que devuelven valores, se puede utilizar el método **bind_result** (función **mysqli_stmt_bind_result**) para asignar a variables los campos que se obtienen tras la ejecución. Utilizando el método **fetch** (**mysqli_stmt_fetch**) se recorren los registros devueltos. Observa el siguiente código:

```
$consulta = $dwes->stmt_init();
$consulta->prepare('SELECT producto, unidades FROM stock WHERE unidades<2');
$consulta->execute();
$consulta->bind_result($producto, $unidades);
while($consulta->fetch()) {
   print "<p>Producto $producto: $unidades unidades.</p>";
}
$consulta->close();
$dwes->close();
```

En el manual de PHP tienes más información sobre consultas preparadas y la clase **mysqli_stmt**.

http://es.php.net/manual/es/class.mysqli-stmt.php

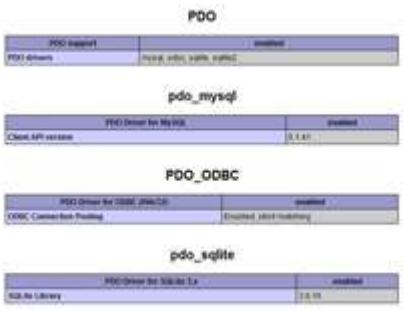
A partir de la página web obtenida en el ejercicio anterior, añade la opción de modificar el número de unidades del producto en cada una de las tiendas. Utiliza una consulta preparada para la actualización de registros en la tabla stock. No es necesario tener en cuenta las tareas de inserción (no existían unidades anteriormente) y borrado (si el número final de unidades es cero).

En esta ocasión es necesario crear un nuevo formulario en la página, en la sección donde se muestra el número de unidades por tienda. Cuando se envía ese formulario, hay que preparar la consulta y ejecutarla una vez por cada registro de la tabla stock (una vez por cada tienda en la que exista stock de ese producto). Revisa el código de la solución y pruébalo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Consultas preparadas con MySQLi -->
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title>Ejercicio Tema 3: Consultas preparadas en MySQLi</title>
      <link href="dwes.css" rel="stylesheet" type="text/css">
   </head>
   <body>
      <?php
         if (isset($ POST['producto'])) $producto = $ POST['producto'];
            @ $dwes = new mysqli("localhost", "dwes", "abc123.", "dwes");
            $error = $dwes->connect_errno;
            if ($error == null) {
               // Comprobamos si tenemos que actualizar los valores
               if (isset($ POST['actualiz'])) {
                  // Preparamos la consulta
                  $tienda = $_POST['tienda'];
                  $unidades = $_POST['unidades'];
                  $consulta = $dwes->stmt_init();
                  $sql    =    "UPDATE    stock    SET    unidades=?    WHERE    tienda=?    AND
producto='$producto'";
                  $consulta->prepare($sql);
                  // La ejecutamos dentro de un bucle, tantas veces como tiendas haya
                  for($i=0;$i<count($tienda);$i++) {
                     $consulta->bind_param('ii', $unidades[$i], $tienda[$i]);
                     $consulta->execute();
                  }
                  $mensaje = "Se han actualizado los datos.";
                  $consulta->close();
               }
            }
         // Si no se ha podido establecer la conexión, generamos un mensaje de error
         else $mensaje = $dwes->connect error;
      ?>
      <div id="encabezado">
         <h1>Ejercicio: Consultas preparadas con MySQLi</h1>
         <form id="form_seleccion" action="." method="post">
            <span>Producto: </span>
            <select name="producto">
               <?php
                  // Rellenamos el desplegable con los datos de todos los productos
                  if ($error == null) {
                     $sql = "SELECT cod, nombre_corto FROM producto";
                     $resultado = $dwes->query($sql);
                     if($resultado) {
                        $row = $resultado->fetch_assoc();
                        while ($row != null) {echo "<option value='${row['cod']}'";
                           // Si se recibió un código de producto lo seleccionamos
                           //  en el desplegable usando selected='true'
                           if (isset($producto) && $producto == $row['cod'])
                              echo " selected='true'";
                           echo ">${row['nombre_corto']}</option>";
                           $row = $resultado->fetch_assoc();
                        }
                        $resultado->close();
                     }
                  }
               ?>
               </select>
            <input type="submit" value="Mostrar stock" name="enviar"/>
         </form>
      </div>
      <div id="contenido">
         <h2>Stock del producto en las tiendas:</h2>
         <?php
```

```
            // Si se recibió un código de producto y no se produjo ningún error
            //  mostramos el stock de ese producto en las distintas tiendas
            if ($error == null && isset($producto)) {
               // Ahora necesitamos también el código de tienda
               $sql = <<<SQL
               SELECT tienda.cod, tienda.nombre, stock.unidades
               FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
               WHERE stock.producto='$producto'
SQL;
               $resultado = $dwes->query($sql);
               if($resultado) {
                  // Creamos un formulario con los valores obtenidos
                  echo '<form id="form actualiz" action="." method="post">';
                  $row = $resultado->fetch assoc();
                  while ($row != null) {
                     // Metemos ocultos el código de producto y los de las tiendas
                     echo "<input type='hidden' name='producto' value='$producto'/>";
                     echo "<input type='hidden' name='tienda[]' value='".$row['cod']."'/>";
                     echo "<p>Tienda ${row['nombre']}: ";
                     // El número de unidades ahora va en un cuadro de texto
                     echo "<input type='text' name='unidades[]' size='4' ";
                     echo "value='".$row['unidades']."'/> unidades.</p>";
                     $row = $resultado->fetch_assoc();
                  }
                  $resultado->close();
                  echo "<input type='submit' value='Actualizar' name='actualiz'/>";
                  echo "</form>";
               }
            }
         ?>
      </div>
      <div id="pie">
         <?php
            // Si se produjo algún error se muestra en el pie
            if ($error != null) echo "<p>Se ha producido un error! $mensaje</p>";
            else {
               echo $mensaje;
               $dwes->close();
               unset($dwes);
            }
         ?>
      </div>
   </body>
</html>
```

3.2.- PHP Data Objects (PDO).



Si vas a programar una aplicación que utilice como sistema gestor de bases de datos MySQL, la extensión **MySQLi** que acabas de ver es una buena opción. Ofrece acceso a todas las características del motor de base de datos, a la vez que reduce los tiempos de espera en la ejecución de sentencias.

Sin embargo, **si en el futuro tienes que cambiar el SGBD** por otro distinto, tendrás que volver a programar gran parte del código de la misma. Por eso, antes de comenzar el desarrollo, es muy importante revisar las características específicas del proyecto. En el caso de que exista la posibilidad, presente o futura, de utilizar otro servidor como almacenamiento, deberás adoptar una capa de abstracción para el acceso a los datos. Existen varias alternativas como ODBC, pero sin duda la opción más recomendable en la actualidad es **PDO**.

El objetivo es que si llegado el momento necesitas cambiar el servidor de base de datos, las modificaciones que debas realizar en tu código sean mínimas. Incluso es posible desarrollar aplicaciones preparadas para utilizar un almacenamiento u otro según se indique en el momento de la ejecución, pero éste no es el objetivo principal de PDO. PDO no abstrae de forma completa el sistema gestor que se utiliza. Por ejemplo, no modifica las sentencias SQL para adaptarlas a las características específicas de cada servidor. Si esto fuera necesario, habría que programar una capa de abstracción completa.

La extensión PDO debe utilizar un driver o controlador específico para el tipo de base de datos que se utilice. Para consultar los controladores disponibles en tu instalación de PHP, puedes utilizar la información que proporciona la función `phpinfo`.

PDO se basa en las características de orientación a objetos de PHP pero, al contrario que la extensión MySQLi, no ofrece un interface de programación dual. Para acceder a las funcionalidades de la extensión tienes que emplear los objetos que ofrece, con sus métodos y propiedades. No existen funciones alternativas.

3.2.1.- Establecimiento de conexiones.

Para establecer una conexión con una base de datos utilizando PDO, debes instanciar un objeto de la clase PDO pasándole los siguientes parámetros (solo el primero es obligatorio):

- ✓ Origen de datos (DSN). Es una cadena de texto que indica qué controlador se va a utilizar y a continuación, separadas por el carácter dos puntos, los parámetros específicos necesarios por el controlador, como por ejemplo el nombre o dirección IP del servidor y el nombre de la base de datos.
- ✓ Nombre de usuario con permisos para establecer la conexión.
- ✓ Contraseña del usuario.
- ✓ Opciones de conexión, almacenadas en forma de array.

Por ejemplo, podemos establecer una conexión con la base de datos 'dwes' creada anteriormente de la siguiente forma:

```
$dwes = new PDO('mysql:host=localhost;dbname=dwes', 'dwes', 'abc123.');
```

Si como en el ejemplo, se utiliza el controlador para MySQL, los parámetros específicos para utilizar en la cadena DSN (separadas unas de otras por el carácter punto y coma) a continuación del prefijo **mysql:** son los siguientes:

- ✓ **`host`**. Nombre o dirección IP del servidor.
- ✓ **`port`**. Número de puerto TCP en el que escucha el servidor.
- ✓ **`dbname`**. Nombre de la base de datos.
- ✓ **`unix_socket`**. Socket de MySQL en sistemas Unix.

Si quisieras indicar al servidor MySQL utilice codificación UTF-8 para los datos que se transmitan, puedes usar una opción específica de la conexión:

```
$opciones = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
$dwes = new PDO('mysql:host=localhost;dbname=dwes', 'dwes', 'abc123.', $opciones);
```

En el manual de PHP puedes consultar más información sobre los controladores existentes, los parámetros de las cadenas DSN y las opciones de conexión particulares de cada uno.
http://es.php.net/manual/es/pdo.drivers.php

Una vez establecida la conexión, puedes utilizar el método **`getAttribute`** para obtener información del estado de la conexión y **`setAttribute`** para modificar algunos parámetros que afectan a la misma. Por ejemplo, para obtener la versión del servidor puedes hacer:

```
$version = $dwes->getAttribute(PDO::ATTR_SERVER_VERSION);
print "Versión: $version";
```

Y si quieres por ejemplo que te devuelva todos los nombres de columnas en mayúsculas:

```
$version = $dwes->setAttribute(PDO::ATTR_CASE, PDO::CASE_UPPER);
```

En el manual de PHP, las páginas de las funciones getAttribute y setAttribute te permiten consultar los posibles parámetros que se aplican a cada una.

http://es.php.net/manual/es/pdo.getattribute.php
http://es.php.net/manual/es/pdo.setattribute.php

Para establecer una conexión con MySQL utilizando PDO, ¿dónde se puede indicar el número de puerto TCP?

- (●) **En la cadena DSN que indica el origen de datos.**
- ( ) En el array en que figuran las opciones específicas de conexión con el servidor.

Se incluye como parte de la cadena DSN utilizando el parámetro "port"

3.2.2.- Ejecución de consultas.

Para ejecutar una consulta SQL utilizando PDO, debes diferenciar aquellas sentencias SQL que no devuelven como resultado un conjunto de datos, de aquellas otras que sí lo devuelven.

En el caso de las consultas de acción, como **`INSERT`**, **`DELETE`** o **`UPDATE`**, el método **`exec`** devuelve el número de registros afectados.

```
$registros = $dwes->exec('DELETE FROM stock WHERE unidades=0');
print "<p>Se han borrado $registros registros.</p>";
```

Si la consulta genera un conjunto de datos, como es el caso de `SELECT`, debes utilizar el método **`query`**, que devuelve un objeto de la clase **`PDOStatement`**.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
```

Por defecto PDO trabaja en modo "`autocommit`", esto es, confirma de forma automática cada sentencia que ejecuta el servidor. Para trabajar con transacciones, PDO incorpora tres métodos:

- ✓ **`beginTransaction`**. Deshabilita el modo "`autocommit`" y comienza una nueva transacción, que finalizará cuando ejecutes uno de los dos métodos siguientes.
- ✓ **`commit`**. Confirma la transacción actual.
- ✓ **`rollback`**. Revierte los cambios llevados a cabo en la transacción actual.

Una vez ejecutado un `commit` o un `rollback`, se volverá al modo de confirmación automática.

```
$ok = true;
$dwes->beginTransaction();
if($dwes->exec('DELETE …') == 0) $ok = false;
if($dwes->exec('UPDATE …') == 0) $ok = false;
…
if ($ok) $dwes->commit();  // Si todo fue bien confirma los cambios
else $dwes->rollback();   //  y si no, los revierte
```

Ten en cuenta que no todos los motores soportan transacciones. Tal es el caso, como ya viste, del motor MyISAM de MySQL. En este caso concreto, PDO ejecutará el método `beginTransaction` sin errores, pero naturalmente no será capaz de revertir los cambios si fuera necesario ejecutar un **`rollback`**.

De una forma similar al anterior ejercicio de transacciones, utiliza PDO para repartir entre las tiendas las tres unidades que figuran en stock del producto con código PAPYRE62GB.
En esta ocasión, para comprobar si los cambios se hacen correctamente en la base de datos y confirmamos la transacción, se revisa el número de registros afectados por la ejecución de las consultas. Revisa el código de la página y comprueba que la segunda vez que intentas ejecutarlo no actualizará los datos, tal y como sucedía en el ejercicio equivalente de la extensión MySQLi.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejercicio: Transacción con PDO -->
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title>Ejercicio: Transacción con PDO</title>
   </head>
   <body>
      <?php
         $dwes = new PDO('mysql:host=localhost;dbname=dwes', 'dwes', 'abc123.');
         // Definimos una variable para comprobar la ejecución de las consultas
         $todo_bien = true;
         // Iniciamos la transacción
         $dwes->beginTransaction();
         $sql = 'UPDATE stock SET unidades=1 WHERE producto="PAPYRE62GB" AND tienda=1';
         if ($dwes->exec($sql) == 0) $todo_bien = false;
         $sql = 'INSERT INTO `stock` (`producto`, `tienda`, `unidades`) VALUES
            ("PAPYRE62GB", 2, 1)';
         if ($dwes->exec($sql) == 0) $todo_bien = false;
         // Si todo fue bien, confirmamos los cambios
         //  y en caso contrario los deshacemos
         if ($todo_bien == true) {
            $dwes->commit();
            print "<p>Los cambios se han realizado correctamente.</p>";
         } else {
            $dwes->rollback();
            print "<p>No se han podido realizar los cambios.</p>";
         }
         unset($dwes);
      ?>
   </body>
</html>
```

Si programas tu aplicación correctamente utilizando "beginTransaction" antes de realizar un cambio, ¿siempre será posible revertirlo utilizando "rollback"?

- ○ Sí
- ◉ **No**

depende de si el motor de almacenamiento que estás utilizando soporta o no transacciones.

3.2.3.- Obtención y utilización de conjuntos de resultados.

Al igual que con la extensión MySQLi, en PDO tienes varias posibilidades para tratar con el conjunto de resultados devuelto por el método **`query`**. La más utilizada es el método **`fetch`** de la clase `PDOStatement`. Este método devuelve un registro del conjunto de resultados, o `false` si ya no quedan registros por recorrer.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
while ($registro = $resultado->fetch()) {
   echo "Producto ".$registro['producto'].": ".$registro['unidades']."<br />";
}
```

Por defecto, el método **`fetch`** genera y devuelve, a partir de cada registro, un array con claves numéricas y asociativas. Para cambiar su comportamiento, admite un parámetro opcional que puede tomar uno de los siguientes valores:

- ✓ **`PDO::FETCH_ASSOC`**. Devuelve solo un array asociativo.
- ✓ **`PDO::FETCH_NUM`**. Devuelve solo un array con claves numéricas.
- ✓ **`PDO::FETCH_BOTH`**. Devuelve un array con claves numéricas y asociativas. Es el comportamiento por defecto.
- ✓ **`PDO::FETCH_OBJ`**. Devuelve un objeto cuyas propiedades se corresponden con los campos del registro.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
```

```
while ($registro = $resultado->fetch(PDO::FETCH_OBJ)) {
  echo "Producto ".$registro->producto.": ".$registro->unidades."<br />";
}
```

- ✓ `PDO::FETCH_LAZY`. Devuelve tanto el objeto como el array con clave dual anterior.
- ✓ `PDO::FETCH_BOUND`. Devuelve true y asigna los valores del registro a variables, según se indique con el método `bindColumn`. Este método debe ser llamado una vez por cada columna, indicando en cada llamada el número de columna (empezando en 1) y la variable a asignar.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$resultado = $dwes->query("SELECT producto, unidades FROM stock");
$resultado->bindColumn(1, $producto);
$resultado->bindColumn(2, $unidades);
while ($registro = $resultado->fetch(PDO::FETCH OBJ)) {
    echo "Producto ".$producto.": ".$unidades."<br />";
}
```

Modifica la página web que muestra el stock de un producto en las distintas tiendas, obtenida en un ejercicio anterior utilizando MySQLi, para que use PDO. Omite la gestión de errores, que veremos en el último punto de este tema.

Comprueba en la solución propuesta los cambios realizados respecto a la solución del ejercicio equivalente que utilizaba MySQLi.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Conjuntos de datos con PDO -->
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title>Ejercicio Tema 3: Conjuntos de resultados en PDO</title>
      <link href="dwes.css" rel="stylesheet" type="text/css">
   </head>
   <body>
      <div id="encabezado">
         <h1>Ejercicio: Conjuntos de resultados en PDO</h1>
         <form id="form_seleccion" action="." method="post">
            <span>Producto: </span>
            <select name="producto">
               <?php
                  if (isset($ POST['producto'])) $producto = $ POST['producto'];
                  // Rellenamos el desplegable con los datos de todos los productos
                  $dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
                  $sql = "SELECT cod, nombre_corto FROM producto";
                  $resultado = $dwes->query($sql);
                  if($resultado) {
                     $row = $resultado->fetch();
                     while ($row != null) {
                        echo "<option value='${row['cod']}'";
                        // Si se recibió un código de producto lo seleccionamos
                        //  en el desplegable usando selected='true'
                        if (isset($producto) && $producto == $row['cod'])
                           echo " selected='true'";
                        echo ">${row['nombre_corto']}</option>";
                        $row = $resultado->fetch();
                     }
                  }
               ?>
            </select>
            <input type="submit" value="Mostrar stock" name="enviar"/>
         </form>
      </div>
      <div id="contenido">
         <h2>Stock del producto en las tiendas:</h2>
         <?php
            // Si se recibió un código de producto
            //  mostramos el stock de ese producto en las distintas tiendas
            if (isset($producto)) {
               $sql = <<<SQL
                  SELECT tienda.nombre, stock.unidades
                  FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
                  WHERE stock.producto='$producto'
SQL;
               $resultado = $dwes->query($sql);
```

```
                if($resultado) {
                   $row = $resultado->fetch();
                   while ($row != null) {
                      echo "<p>Tienda ${row['nombre']}: ${row['unidades']} unidades.</p>";
                      $row = $resultado->fetch();
                   }
                }
             }
          ?>
       </div>
       <div id="pie">
          <?php
             unset($dwes);
          ?>
       </div>
    </body>
</html>
```

¿Cuál es el comportamiento por defecto del método "fetch"?

- (•) Devuelve un array con claves numéricas y asociativas.
- ( ) Devuelve un array asociativo.

Es similar a utilizar el parámetro "`PDO::FETCH_BOTH`"

3.2.4.- Consultas preparadas.

Al igual que con MySQLi, también utilizando PDO podemos preparar consultas parametrizadas en el servidor para ejecutarlas de forma repetida. El procedimiento es similar e incluso los métodos a ejecutar tienen prácticamente los mismos nombres.

Para preparar la consulta en el servidor MySQL, deberás utilizar el método `prepare` de la clase PDO. Este método devuelve un objeto de la clase `PDOStatement`. Los parámetros se pueden marcar utilizando signos de interrogación como en el caso anterior.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$consulta = $dwes->prepare('INSERT INTO familia (cod, nombre) VALUES (?, ?)');
```

O también utilizando parámetros con nombre, precediéndolos por el símbolo de dos puntos.

```
$dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
$consulta = $dwes->prepare('INSERT INTO familia (cod, nombre) VALUES (:cod, :nombre)');
```

Antes de ejecutar la consulta hay que asignar un valor a los parámetros utilizando el método `bindParam` de la clase `PDOStatement`. Si utilizas signos de interrogación para marcar los parámetros, el procedimiento es equivalente al método `bindColumn` que acabamos de ver.

```
$cod_producto = "TABLET";
$nombre producto = "Tablet PC";
$consulta->bindParam(1, $cod_producto);
$consulta->bindParam(2, $nombre_producto);
```

Si utilizas parámetros con nombre, debes indicar ese nombre en la llamada a `bindParam`.

```
$consulta->bindParam(":cod", $cod_producto);
$consulta->bindParam(":nombre", $nombre_producto);
```

Tal y como sucedía con la extensión MySQLi, cuando uses `bindParam` para asignar los parámetros de una consulta preparada, deberás usar siempre variables como en el ejemplo anterior.

Una vez preparada la consulta y enlazados los parámetros con sus valores, se ejecuta la consulta utilizando el método `execute`.

```
$consulta->execute();
```

Alternativamente, es posible asignar los valores de los parámetros en el momento de ejecutar la consulta, utilizando un array (asociativo o con claves numéricas dependiendo de la forma en que hayas indicado los parámetros) en la llamada a `execute`.

```
$parametros = array(":cod" => "TABLET", ":nombre" => "Tablet PC");
$consulta->execute($parametros);
```

Puedes consultar la información sobre la utilización en PDO de consultas preparadas y la clase PDOStatement en el manual de PHP.

http://es.php.net/manual/es/class.pdostatement.php

Modifica el ejercicio sobre consultas preparadas que realizaste con la extensión MySQLi, el que modificaba el número de unidades de un producto en las distintas tiendas, para que utilice ahora la extensión PDO.

Como puedes comprobar, para obtener la solución se puede aprovechar la mayoría del código existente en el ejercicio anterior.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Consultas preparadas con PDO -->
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title>Ejercicio Tema 3: Consultas preparadas en PDO</title>
      <link href="dwes.css" rel="stylesheet" type="text/css">
   </head>
   <body>
      <?php
         if (isset($ POST['producto'])) $producto = $ POST['producto'];
         $dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
         // Comprobamos si tenemos que actualizar los valores
         if (isset($_POST['actualiz'])) {
            // Preparamos la consulta
            $tienda = $ POST['tienda'];
            $unidades = $_POST['unidades'];
            $sql = "UPDATE stock SET unidades=:unidades WHERE tienda=:tienda";
            $sql .= " AND producto='$producto'";
            $consulta=$dwes->prepare($sql);
            // La ejecutamos dentro de un bucle, tantas veces como tiendas haya
            for($i=0;$i<count($tienda);$i++) {
               $consulta->bindParam(":unidades", $unidades[$i]);
               $consulta->bindParam(":tienda", $tienda[$i]);
               $consulta->execute();
            }
            $mensaje = "Se han actualizado los datos.";
         }
      ?>
      <div id="encabezado">
         <h1>Ejercicio: Consultas preparadas en PDO</h1>
         <form id="form_seleccion" action="." method="post">
            <span>Producto: </span>
            <select name="producto">
            <?php
               // Rellenamos el desplegable con los datos de todos los productos
               $sql = "SELECT cod, nombre_corto FROM producto";
               $resultado = $dwes->query($sql);
               if($resultado) {
                  $row = $resultado->fetch();
                  while ($row != null) {
                     echo "<option value='${row['cod']}'";
                     // Si se recibió un código de producto lo seleccionamos
                     //  en el desplegable usando selected='true'
                     if (isset($producto) && $producto == $row['cod'])
                        echo " selected='true'";
                     echo ">${row['nombre_corto']}</option>";
                     $row = $resultado->fetch();
                  }
               }
            ?>
            </select>
```

```
            <input type="submit" value="Mostrar stock" name="enviar"/>
         </form>
      </div>
      <div id="contenido">
         <h2>Stock del producto en las tiendas:</h2>
         <?php
            // Si se recibió un código de producto y no se produjo ningún error
            //  mostramos el stock de ese producto en las distintas tiendas
            if (isset($producto)) {
               // Ahora necesitamos también el código de tienda
               $sql = <<<SQL
                  SELECT tienda.cod, tienda.nombre, stock.unidades
                  FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
                  WHERE stock.producto='$producto'
SQL;
               $resultado = $dwes->query($sql);
               if($resultado) {
                  // Creamos un formulario con los valores obtenidos
                  echo '<form id="form_actualiz" action="." method="post">';
                  $row = $resultado->fetch();
                  while ($row != null) {
                     // Metemos ocultos el código de producto y los de las tiendas
                     echo "<input type='hidden' name='producto' value='$producto'/>";
                     echo "<input type='hidden' name='tienda[]' value='".$row['cod']."'/>";
                     echo "<p>Tienda ${row['nombre']}: ";
                     // El número de unidades ahora va en un cuadro de texto
                     echo "<input type='text' name='unidades[]' size='4' ";
                     echo "value='".$row['unidades']."'/> unidades.</p>";
                     $row = $resultado->fetch();
                  }
                  echo "<input type='submit' value='Actualizar' name='actualiz'/>";
                  echo "</form>";
               }
            }
         ?>
      </div>
      <div id="pie">
         <?php
            echo $mensaje;
            unset($dwes);
         ?>
      </div>
   </body>
</html>
```

4.- Errores y manejo de excepciones.

A buen seguro que, conforme has ido resolviendo ejercicios o simplemente probando código, te has encontrado con errores de programación. Algunos son reconocidos por el entorno de desarrollo (NetBeans), y puedes corregirlos antes de ejecutar. Otros aparecen en el navegador en forma de mensaje de error al ejecutar el guión.

PHP define una clasificación de los errores que se pueden producir en la ejecución de un programa y ofrece métodos para ajustar el tratamiento de los mismos. Para hacer referencia a cada uno de los niveles de error, PHP define una serie de constantes. Cada nivel se identifica por una constante. Por ejemplo, la constante `E_NOTICE` hace referencia a avisos que pueden indicar un error al ejecutar el guión, y la constante `E_ERROR` engloba errores fatales que provocan que se interrumpa forzosamente la ejecución.

La lista completa de constantes la puedes consultar en el manual de PHP, donde también se describe el tipo de errores que representa.

http://es.php.net/manual/es/errorfunc.constants.php

La configuración inicial de cómo se va a tratar cada error según su nivel se realiza en `php.ini` el fichero de configuración de PHP. Entre los principales parámetros que puedes ajustar están:

- ✓ `error_reporting`. Indica qué tipos de errores se notificarán. Su valor se forma utilizando los operadores a nivel de bit para combinar las constantes anteriores. Su valor predeterminado es `E_ALL & ~E_NOTICE` que indica que se notifiquen todos los errores (`E_ALL`) salvo los avisos en tiempo de ejecución (`E_NOTICE`).
- ✓ `display_errors`. En su valor por defecto (`On`), hace que los mensajes se envíen a la salida estándar (y por lo tanto se muestren en el navegador). Se debe desactivar (`Off`) en los servidores que no se usan para desarrollo sino para producción.

Existen otros parámetros que podemos utilizar en php.ini para ajustar el comportamiento de PHP cuando se produce un error.

http://es.php.net/manual/es/errorfunc.configuration.php

Desde código, puedes usar la función `error_reporting` con las constantes anteriores para establecer el nivel de notificación en un momento determinado. Por ejemplo, si en algún lugar de tu código figura una división en la que exista la posibilidad de que el divisor sea cero, cuando esto ocurra obtendrás un mensaje de error en el navegador. Para evitarlo, puedes desactivar la notificación de errores de nivel `E_WARNING` antes de la división y restaurarla a su valor normal a continuación:

```
error_reporting(E_ALL & ~E_NOTICE & ~E_WARNING);
$resultado = $dividendo / $divisor;
error_reporting(E_ALL & ~E_NOTICE);
```

Al usar la función **error_reporting** solo controlas qué tipo de errores va a notificar PHP. A veces puede ser suficiente, pero para obtener más control sobre el proceso existe también la posibilidad de reemplazar la gestión de los mismos por la que tú definas. Es decir, puedes programar una función para que sea la que ejecuta PHP cada vez que se produce un error. El nombre de esa función se indica utilizando **set_error_handler** y debe tener como mínimo dos parámetros obligatorios (el nivel del error y el mensaje descriptivo) y hasta otros tres opcionales con información adicional sobre el error (el nombre del fichero en que se produce, el número de línea, y un volcado del estado de las variables en ese momento).

```
set_error_handler("miGestorDeErrores");
$resultado = $dividendo / $divisor;
restore_error_handler();

function miGestorDeErrores($nivel, $mensaje)
{
    switch($nivel) {
        case E WARNING:
            echo "Error de tipo WARNING: $mensaje.<br />";
            break;
        default:
            echo "Error de tipo no especificado: $mensaje.<br />";
    }
}
```

La función **restore_error_handler** restaura el manejador de errores original de PHP (más concretamente, el que se estaba usando antes de la llamada a **set_error_handler**).

4.1.- Excepciones.

A partir de la versión 5 se introdujo en PHP un modelo de excepciones similar al existente en otros lenguajes de programación:

- ✓ El código susceptible de producir algún error se introduce en un bloque **try**.
- ✓ Cuando se produce algún error, se lanza una excepción utilizando la instrucción **throw**.
- ✓ Después del bloque `try` debe haber como mínimo un bloque `catch` encargado de procesar el error.
- ✓ Si una vez acabado el bloque `try` no se ha lanzado ninguna excepción, se continúa con la ejecución en la línea siguiente al bloque o bloques `catch`.

Por ejemplo, para lanzar una excepción cuando se produce una división por cero podrías hacer:

```
try {
    if ($divisor == 0)
        throw new Exception("División por cero.");
    $resultado = $dividendo / $divisor;
}
catch (Exception $e) {
    echo "Se ha producido el siguiente error: ".$e->getMessage();
}
```

PHP ofrece una clase base `Exception` para utilizar como manejador de excepciones. Para lanzar una excepción no es necesario indicar ningún parámetro, aunque de forma opcional se puede pasar un mensaje de error (como en el ejemplo anterior) y también un código de error.

Entre los métodos que puedes usar con los objetos de la clase **Exception** están:

- ✓ **getMessage**. Devuelve el mensaje, en caso de que se haya puesto alguno.
- ✓ **getCode**. Devuelve el código de error si existe.

Las funciones internas de PHP y muchas extensiones como MySQLi usan el sistema de errores visto anteriormente. Solo las extensiones más modernas orientadas a objetos, como es el caso de PDO, utilizan este modelo de excepciones. En este caso, lo más común es que la extensión defina sus propios manejadores de errores heredando de la clase **Exception** (veremos cómo utilizar la herencia en una unidad posterior).

Utilizando la clase ErrorException es posible traducir los errores a excepciones.
http://es.php.net/manual/es/class.errorexception.php

Concretamente, la clase **PDO** permite definir la fórmula que usará cuando se produzca un error, utilizando el atributo `PDO::ATTR_ERRMODE`. Las posibilidades son:

- ✓ `PDO::ERRMODE_SILENT`. No se hace nada cuando ocurre un error. Es el comportamiento por defecto.
- ✓ `PDO::ERRMODE_WARNING`. Genera un error de tipo `E_WARNING` cuando se produce un error.
- ✓ `PDO::ERRMODE_EXCEPTION`. Cuando se produce un error lanza una excepción utilizando el manejador propio `PDOException`.

Es decir, que si quieres utilizar excepciones con la extensión **PDO**, debes configurar la conexión haciendo:

```
$dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Por ejemplo, el siguiente código:

```
$dwes = new PDO("mysql:host=localhost; dbname=dwes", "dwes", "abc123.");
$dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
try {
    $sql = "SELECT * FROM stox";
    $result = $dwes->query($sql);
    …
}
catch (PDOException $p) {
    echo "Error ".$p->getMessage()."<br />";
}
```

Captura la excepción que lanza PDO debido a que la tabla no existe. El bloque `catch` muestra el siguiente mensaje:

```
Error SQLSTATE[42S02]: Base table or view not found: 1146 Table 'dwes.stox' doesn't exist
```

Agrega control de excepciones para controlar los posibles errores de conexión que se puedan producir en el último ejercicio, mostrando en la parte inferior de la pantalla los errores que se produzcan.
Revisa en la solución propuesta los cambios realizados para utilizar excepciones en el establecimiento de la conexión

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "
http://www.w3.org/TR/html4/loose.dtd">
<!-- Desarrollo Web en Entorno Servidor -->
<!-- Tema 3 : Trabajar con bases de datos en PHP -->
<!-- Ejemplo: Utilización de excepciones en PDO -->
<html>
   <head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <title>Ejercicio Tema 3: Excepciones en PDO</title>
      <link href="dwes.css" rel="stylesheet" type="text/css">
   </head>
   <body>
      <?php
         if (isset($ POST['producto'])) $producto = $ POST['producto'];
         try {
            $dwes = new PDO("mysql:host=localhost;dbname=dwes", "dwes", "abc123.");
            $dwes->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
         }
         catch (PDOException $e) {
            $error = $e->getCode();
            $mensaje = $e->getMessage();
         }
         if (!isset($error)) {
            // Comprobamos si tenemos que actualizar los valores
            if (isset($_POST['actualiz'])) {
               // Preparamos la consulta
               $tienda = $_POST['tienda'];
```

```
                $unidades = $_POST['unidades'];
                $sql = "UPDATE stock SET unidades=:unidades WHERE tienda=:tienda";
                $sql .= " AND producto='$producto'";
                $consulta=$dwes->prepare($sql);
                // La ejecutamos dentro de un bucle, tantas veces como tiendas haya
                for($i=0;$i<count($tienda);$i++) {
                   $consulta->bindParam(":unidades", $unidades[$i]);
                   $consulta->bindParam(":tienda", $tienda[$i]);
                   $consulta->execute();
                }
                $mensaje = "Se han actualizado los datos.";
             }
          }
       ?>
       <div id="encabezado">
          <h1>Ejercicio: Utilización de excepciones en PDO</h1>
          <form id="form seleccion" action="." method="post">
             <span>Producto: </span>
             <select name="producto">
                <?php
                   if (!isset($error)) {
                      // Rellenamos el desplegable con los datos de todos los productos
                      $sql = "SELECT cod, nombre corto FROM producto";
                      $resultado = $dwes->query($sql);
                      if($resultado) {
                         $row = $resultado->fetch();
                         while ($row != null) {
                            echo "<option value='${row['cod']}'";
                            // Si se recibió un código de producto lo seleccionamos
                            //  en el desplegable usando selected='true'
                            if (isset($producto) && $producto == $row['cod'])
                               echo " selected='true'";
                            echo ">${row['nombre corto']}</option>";
                            $row = $resultado->fetch();
                         }
                      }
                   }
                ?>
             </select>
             <input type="submit" value="Mostrar stock" name="enviar"/>
          </form>
       </div>
       <div id="contenido">
          <h2>Stock del producto en las tiendas:</h2>
          <?php
             // Si se recibió un código de producto y no se produjo ningún error
             //  mostramos el stock de ese producto en las distintas tiendas
             if (!isset($error) && isset($producto)) {
                // Ahora necesitamos también el código de tienda
                $sql = <<<SQL
                   SELECT tienda.cod, tienda.nombre, stock.unidades
                   FROM tienda INNER JOIN stock ON tienda.cod=stock.tienda
                   WHERE stock.producto='$producto'
SQL;
                $resultado = $dwes->query($sql);
                if($resultado) {
                   // Creamos un formulario con los valores obtenidos
                   echo '<form id="form_actualiz" action="." method="post">';
                   $row = $resultado->fetch();
                   while ($row != null) {
                      // Metemos ocultos el código de producto y los de las tiendas
                      echo "<input type='hidden' name='producto' value='$producto'/>";
                      echo "<input type='hidden' name='tienda[]' value='".$row['cod']."'/>";
                      echo "<p>Tienda ${row['nombre']}: ";
                      // El número de unidades ahora va en un cuadro de texto
                      echo "<input type='text' name='unidades[]' size='4' ";
                      echo "value='".$row['unidades']."'/> unidades.</p>";
                      $row = $resultado->fetch();
                   }
                   echo "<input type='submit' value='Actualizar' name='actualiz'/>";
                   echo "</form>";
                }
             }
          ?>
       </div>
       <div id="pie">
          <?php
```

```
            if (isset($error)) echo "<p>Se ha producido un error! $mensaje</p>";
            else {
               echo $mensaje;
               unset($dwes);
            }
         ?>
      </div>
   </body>
</html>
```

¿Cuántos bloques "`catch`" se han de utilizar después de un bloque "`try`"?

- ○ Uno.
- ◉ **Uno o más.**

se ejecutará el bloque "`catch`" cuyo manejador coincida que se ha lanzado mediante "`throw`". Esto sólo tiene sentido si se utilizan distintos manejadores extendiendo el base que incluye PHP, "`Exception`".

Intenta resolver la siguiente actividad relativa a las extensiones para manejo bases de datos MySQL y al control de errores.

Extensión mysqli	
Si utilizas mysqli para establecer una conexión, ¿qué método usaras para utilizar otra base de datos?	select_db
¿Cuál es la forma procedimental para establecer conexiones utilizando mysqli?	msqli_result
En mysqli, ¿de qué clase es el objeto que devuelve una llamada al método query?	mysqli_connect
En InnoDB por defecto cada consulta se incluye dentro de su propia transacción. ¿Cuál es el nombre de la función de mysqli que permite cambiar este comportamiento?	mysqli_autocommit
¿Qué clase debes utilizar si quieres ejecutar consultas preparadas en mysqli?	mysqli_stmt
En mysqli, ¿cuál debe ser el valor del parámetro del método fetch_array para que devuelva un array asociativo?	MYSQLI_ASSOC

PDO	
¿Qué valor debemos pasarle al método fetch para que devuelva solamente un array numérico?	PDO::FETCH_NUM
¿Qué método se utiliza en PDO para obtener información del estado de la conexión?	getAttribute
¿Qué método se utiliza en PDO para indicar que se quiere comenzar una nueva transacción?	beginTransaction
¿Qué método se utiliza en PDO para ejecutar una consulta de tipo DELETE?	exec

Errores y manejo de excepciones	
Instrucción que debe seguir a un bloque try.	catch
Nombre de la función de PHP que permite establecer un manejador de errores propio.	set_error_handler
Clase base para manejar excepciones en PHP.	exception
Parámetro de php.ini que indica qué errores se notificarán.	error_reporting