UNIDAD 3

# Trabajar con bases de datos en PHP.

En esta unidad utilizarás esos conocimientos para realizar programas sencillos que utilicen información almacenada en bases de datos. El principal sistema gestor de bases de datos que se utiliza junto al lenguaje PHP es MySQL. A continuación, aprenderás a acceder desde PHP a bases de datos MySQL utilizando las extensiones MySQLi y PDO.

# 1.- Acceso a bases de datos desde PHP.

Una de las aplicaciones más frecuentes de PHP es generar una interfaz 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 para 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.

Veremos 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();

# 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 ni tablas transaccionales**. 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,** y añade además algunas extensiones propias.

En el tema anterior instalamos **Xampp** en **Windows**, el SGBD que viene con **Xampp** es **MariaDB**que es un sistema de gestión de bases de datos derivado de**MySQL**con licencia GPL. A todos los efectos, a lo largo de este tema y del módulo, dará igual utilizar uno u otro, de hecho, **MariaDB** está desarrollado por parte de la comunidad de **MySQL**, antes de que el proyecto fuese adquirido por Oracle y, salvo ligeros cambios, ambos son altamente compatibles.

Guía de MySQL: <https://downloads.mysql.com/docs/refman-5.0-es.pdf>

## 2.1.- Base de datos proyecto

La base de datos **proyecto**. En ella meteremos las tablas del siguiente esquema, fíjate en los campos y las relaciones entre ellas. De igual manera en dicho archivo crearemos el usuario "**gestor**" con contraseña "**secreto**" y le daremos todos los permisos en la base de datos "**proyecto**".

A continuación, se muestra el código SQL para la creación de la base de datos proyecto, junto con sus tablas y los datos. Copia el código a un fichero llamado proyecto.sql para llevar a cabo la creación de la base de datos siguiendo las indicaciones de la sección siguiente.

## 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 una interfaz gráfica 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.
* **mysqladmin**. Es un cliente específico para tareas de administración.
* **mysqlcheck**. Permite correr acciones de mantenimiento de las tablas de la base de datos. Comprueba, repara, optimiza y analiza tablas.
* **mysqldump**. Es una utilidad de cliente para realizar copias de seguridad de las bases de datos.
* **mysqlimport**. Proporciona una línea de comandos para importar los contenidos de un fichero.
* **mysqlshow**. Muestra información sobre bases de datos y tablas.

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 usará "localhost".

Para establecer una conexión al servidor local con la herramienta mysql, podemos hacer:

El ejecutable "mysql" está en "c:\xampp\mysql\bin\" con una consola cmd o powershell abierta en esta ubicación (si no la hemos metido en la ruta (path) de los ejecutable del sistema) basta con **teclear "mysql -u root"** .

Con cualquier otro usuario: **mysql -u usuario -p baseDeDatos**. Por ejemplo: "mysql -u juan -p proyecto".

**mysql**

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

También puedes usar el comando **mysql** para que ejecute todas las sentencias de un archivo de procesamiento por lotes (normalmente con extensión .**sql**).

mysql -u root -p < proyecto.sql

**Debes conocer**

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

**MySQL** **Workbench** es una herramienta genérica con interfaz 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.

**phpMyAdmin** es una aplicación web muy popular para la administración de servidores MySQL. Presenta una interfaz 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.

**Recomendación**

¿Podemos o no almacenar "emojis" en **MySQL**?, la respuesta es si, pero no utilizando el "**utf-8**" que normalmente se usaba.

En 2010 (Con su versión **5.5.3**) **MySQL** agrega una variante a **"utf-8"** llamada "**utf8mb4"**. Con este nuevo tipo de codificación, cada carácter puede ser representado hasta 4 bytes, lo que nos permitirá guardar "emojis" en nuestras tablas algo cada vez más necesario.

mysql>create database mi\_base\_de\_datos CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_unicode\_ci;

## 2.2.2.- phpMyAdmin.

XAMPP incluye la herramienta **phpMyAdmin** con lo que no hace falta ninguna instalación adicional. Para ejecutarlo pichamos en el botón **Admin** asociado a MySQL en el panel de control de XAMPP.

Por otro lado, si el entorno se ha creado instalando servidor MySQL, debes instalarlo de forma individual ya que no viene incluido en la distribución del servidor.

En el caso de Ubuntu lo podemos hacer de dos formas:

* La más sencilla es usar el gestor de paquetes, por ejemplo, tecleando desde un terminal: **sudo apt 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/**".
* La otra es descargarnos directamente el paquete de la web de **phpMyadmin** ([**https://www.phpmyadmin.net/**)](https://www.phpmyadmin.net/)) y descomprimirlo en "**/var/www/html**"  (el directorio raíz de Apache en Ubuntu). Lo más sencillo es que renombremos la carpeta con el nombre **phpmyadmin**.

La interfaz 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 "**proyecto**", 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 esta.
* **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.

Para crear la base de datos "**proyecto**" y el usuario "gestor" selecciona la pestaña Importar y pincha en el botón Seleccionar archivo, selecciona el archivo donde reside el esquema y los datos de la base de datos y finalmente pincha el botón Continuar en la parte inferior de la página.

Web phpmyadmin: <https://docs.phpmyadmin.net/en/latest/>

## 2.2.1.- 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 que teclees a partir de ese instante se ejecutarán sobre la base de datos "dwes".

**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"** y para ver la versión instalada **"version"**, puedes ver los comandos disponibles usando **"mysqladmin --help"**:

mysqladmin --help

sudo mysqladmin status

sudo mysqladmin version

Fíjate que los comandos anteriores no necesitan especificar usuario porque son solamente de información. Si quisiésemos, por ejemplo, crear una base de datos, borrarla, reiniciar el servidor... con **mysqladmin,** deberíamos hacerlo con **sudo** y especificando el usuario **root**. Por ejemplo, para crear una base de datos de nombre **miBase**:

sudo mysqladmin -u root create miBase

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

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 controlador disponible para realizar esta función era **libmysql**, que no estaba optimizado para ser utilizado desde PHP. A partir de la versión **5.3**, 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 una 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:

* new mysqli();
* mysqli\_connect();

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.

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

Entre las mejoras que aporta respecto a la antigua extensión **mysql**, figuran:

* Interfaz orientada a objetos.
* Soporte para transacciones.
* Soporte para consultas preparadas.
* Mejores opciones de depuración y seguridad.

Las opciones de configuración 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"**. <https://www.php.net/manual/es/mysqli.configuration.php>

## 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 "**proyecto**" puedes hacer:

$conProyecto = new mysqli('localhost', 'gestor', 'secreto', 'proyecto');

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

$conProyecto = new mysqli();

$conProyecto->connect('localhost', 'gestor', 'secreto', 'proyecto');

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 0 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 "**proyecto**" y finaliza la ejecución si se produce algún error:

$conProyecto = new mysqli('localhost', 'gestor', 'secreto', 'proyecto');

if ($conProyecto->connect\_error) {

die('Error de Conexión (' . $conProyecto->connect\_errno . ') ' . $conProyecto->connect\_error);

//die() detiene la ejecución del script php

}

Por otro lado, es preferible utilizar **excepciones** para realizar el control de errores en PHP tal y como se hace por defecto a partir de la versión de PHP 8.1. Si estoy trabajando con una versión anterior de PHP y queremos activar la generación de excepciones durante el uso de la librería deberemos establecerlo mediante los **flags** de informe de errores **MYSQLI\_REPORT\_ERROR | MYSQLI\_REPORT\_STRICT**

Si quisiéramos realizar control de errores mediante el tratamiento de excepciones podríamos codificarlo de la siguiente manera:

$controlador = new mysqli\_driver();

$controlador->report\_mode = MYSQLI\_REPORT\_ERROR | MYSQLI\_REPORT\_STRICT;

try {

$conProyecto = new mysqli('localhost', 'gestor', 'secreto', 'proyecto');

} catch (mysqli\_sql\_exception $e) {

die ("Error en la conexión a la base de datos: " . $e->getMessage());

}

*Observa que, 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.* [*https://www.php.net/manual/es/language.operators.errorcontrol.php*](https://www.php.net/manual/es/language.operators.errorcontrol.php)

Si 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 ddbb, lógicamente el usuario con el que hemos iniciado la conexión debe tener permisos en la nueva.

$conProyecto->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.

$conProyecto->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**).

Al 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 = $conProyecto->query('SELECT producto, unidades FROM stocks', 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****.* [*https://www.php.net/manual/es/mysqli.real-query.php*](https://www.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->free();

## 3.1.3.- Transacciones.

Si necesitas utilizar transacciones deberás asegurarte de que estén soportadas por el motor de almacenamiento que gestiona tus tablas en **MySQL**. Si utilizas InnoDB, por defecto cada consulta individual se incluye dentro de su propia transacción. Puedes gestionar este comportamiento con el método **autocommit** (función **mysqli\_autocommit**).

$conProyecto->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.

...

$conProyecto->query('DELETE FROM stocks WHERE tienda=2'); // Inicia una transacción

$conProyecto->query('UPDATE stocks SET unidades=3 WHERE producto=24');

. . .

$conProyecto->commit(); // Confirma los cambios

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

## 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 una 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.

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

1. **MYSQLI\_NUM**. Devuelve un array con claves numéricas.
2. **MYSQLI\_ASSOC**. Devuelve un array asociativo.
3. **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 devolverán **null** cuando no haya más registros en el conjunto de resultados.

Clase mysqli\_result: <https://www.php.net/manual/es/class.mysqli-result.php>

Bootstrap: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>

## 3.1.5.- Consultas preparadas.