CONEXIÓN PHP - MYSQL

Para crear una conexión puede ser necesario lo siguiente:

- Base de datos.
- Extensiones de conexión:
    - **MySQLi (MySQL Improved).**
        - Es una extensión de MySQL que incluye nuevas funciones para el acceso a datos de bases creadas con MySLQ.
        - Permite un uso procedimental y también orientado a objetos.
    - **PDO (PHP Data Object).**
        - Extensión que permite la conexión de PHP a bases de datos creadas con distintos sistemas gestores de bases de datos como MySQL, Informix, PostgreSQL y otros.
        - Permite un uso únicamente orientado a objetos.
- Formulario, guardado como HTML o PHP, para introducir datos en una base usando una operación de inserción (INSERT INTO).
- Formulario, guardado como HTML o PHP, para modificar o eliminar datos en una base que implican operaciones de consulta, modificación y/o eliminación (SELECT, UPDATE Y DELETE).
- Formulario, guardado como HTML o PHP, para confirmar datos existentes en la base, lo que implica una consulta para hacer una comparación entre los datos que hay en la base y los que se introduce por un formulario. (SELECT)
- También se puede usar una base de datos sin formulario porque lo que se va a hacer es mostrar los datos en una página web, tras hacer una consulta en la base y enviar los datos a la página. (SELECT).

Extensión o clase MYSQLI.

- Clase de PHP que representa una conexión entre PHP y una base de datos MySQL.
- Incluye diversas propiedades y métodos.
- Algunos **métodos**:
    - **mysqli_connect().**
        - Permite establecer la conexión PHP-MySQL.
        - Puede crearse un archivo sólo con los datos de conexión o incluirlos, por ejemplo, en el archivo de PHP que procese los datos que se enviarán a, o desde la base.
        - También se puede crear una clase genérica que sirva para crear conexiones de bases de datos e incluso establecer consultas.
        - Si se encuentra en un archivo distinto al que procesará los datos, en este último, hay que incluir la llamada al archivo con la conexión usando métodos como include() o include_once().
        - Sintaxis:
            - mysqli_connect (“servidor[:puerto]”, “usuario”,” contraseña”, “base”);

- variable de conexión = mysqli_connect (“servidor[:puerto]”, “usuario”,” contraseña”, “base”);
- Variable de conexión**:**
  - Es una variable que sirve para cargar la conexión, lo que facilita el manejo de otros métodos que necesitan los parámetros de una conexión
- Argumentos o parámetros de la función:
  - Todos los parámetros para incluir en el método pueden ser datos literales entre comillas, variables que contengan los datos, o ambos.
  - Parámetros:
    - **Servidor**.
      - Servidor al que uno se conecta.
      - Puede ser local o remoto.
      - Puede ser necesario incluir el número de puerto para la conexión, si el predeterminado se ha cambiado
      - Puerto predeterminado para MySQL: 3306.
    - **Usuario**
      - Usuario que va a trabajar con la conexión.
      - Si no se han creado usuarios, el usuario por defecto existente para conectarse es *root*.
    - **Contraseña**.
      - Contraseña asociada al usuario que se va a conectar.
      - Si no se ha establecido una contraseña, se incluye una cadena de caracteres de longitud cero (“”).
    - **Base**.
      - Base de datos con la que se establecerá la conexión.

- Ejemplos:
  - **Función con los datos de conexión literales**.
    - $conexion = mysqli_connect (“localhost”, “root”,””, “biblioteca”); // Conexión usando datos literales y cargando después la conexión realizada en la variable $conexión, para usos posteriores.
    - $conexion = mysqli_connect (“localhost:3310”, “root”,””, “ventas”); // Conexión usando datos literales y cargando después la conexión realizada en la variable $conexión, para usos posteriores. También se especifica un puerto distinto al predeterminado.
    - mysqli_connect (“bases.misitio.com”, “root”,”1234”, “tienda”); // Igual que la opción anterior, pero sin carga de

la conexión en una variable y usando un servidor de base de datos remoto (dato que hay que consultar en la cuenta de hosting o alojamiento que tenga uno cada uno).

- **Función con variables que contienen los datos de conexión**.
  - $servidor =”localhost”; // Ejemplo de servidor local o $servidor =”basesSQL.tienda.com”; // Ejemplo de remoto.
  - $usuario =”root”;
  - $contraseña = “1234” o $contraseña =””; // Con y sin contraseña respectivamente.
  - $base = “coches”;
  - $conexion = mysqli_connect (“$servidor”, “$usuario”, ”$contraseña”, “$base”); // Conexión usando variable y cargándola en la variable $conexión para usos posteriores.

- **mysqli_set_charset().**
  - Método para especificar el juego de caracteres a usar en la página PHP cuando en ella se carguen los datos provenientes de la base.
  - Sintaxis:
    - mysqli_set_charset (variable conexión, “juego de caracteres”);
  - Ejemplo:
    - mysqli_set_charset ($conexion, “utf-8”);
- **mysqli_query().**
  - Método que permite ejecutar una instrucción LMD como consultas, inserciones, borrados o modificaciones (Select, Insert, Delete o Update).
  - La instrucción SQL suele cargarse como una cadena de caracteres en una variable que, a su vez, servirá como parámetro de la función.
  - Sintaxis:
    - mysqli_query (variable conexión, variable con la instrucción SQL);
  - Ejemplos
    - Consulta:

      $consulta = “SELECT * FROM coches WHERE marca = ‘SEAT’”;
      mysqli_query ($conexion, $consulta); // Usando la variable con la instrucción SQL.
      mysqli_query ($conexion, “SELECT * FROM coches WHERE marca = ‘SEAT’”); // Usando directamente la instrucción SQL como argumento.
    - Inserción:

      $insertar = “INSERT INTO coches (‘100BV’, ‘Seat’,’Ibiza’,’ Diesel’, 15000.00)”;
      mysqli_query ($conexion, $insertar); // Usando la variable con la instrucción SQL.
      mysqli_query ($conexion, “INSERT INTO coches (‘100BV’, ‘Seat’,’Ibiza’,’ Diesel’, 15000.00)”); // Usando directamente la instrucción SQL como argumento.
    - Actualización:

```
$cambiar = "UPDATE coches SET precio = 20000.50 WHERE modelo = 'Toledo'";
mysqli_query ($conexion, $cambiar); // Usando la variable con la instrucción SQL.
mysqli_query ($conexion, "UPDATE coches SET precio = 20000.50 WHERE modelo = 'Toledo'"); // Usando directamente la instrucción SQL como argumento.
```

- Eliminación:

```
$borrar = "DELETE FROM coches WHERE marca = 'Dacia'";
mysqli_query ($conexion, $borrar); // Usando la variable con la instrucción SQL.
mysqli_query ($conexion, "DELETE FROM coches WHERE marca = 'Dacia'"); // Usando directamente la instrucción SQL como argumento.
```

- **mysqli_num_rows().**
  - Permite obtener el número de filas que coinciden con la consulta especificada usando PHP. (También se pueden contar filas usando la función de agregado count(*) de MySQL).
  - Con ello, se puede confirmar si un dato buscado está incluido en una base (aparecen filas), o no (no aparecen filas).
  - Útil para identificar usuarios.
  - El resultado obtenido puede cargarse en una variable y posteriormente ésta ser usada en una condición.
  - Resultados:
    - **Número mayor que 0**:
      - Indica el número de filas afectadas por la operación LMD.
    - **0**:
      - Indica que no ha habido filas afectadas por la operación LMD. No ha habido coincidencias o no se ha ejecutado la consulta.
    - **-1**:
      - Indica que ha habido un error.
  - Puede facilitarse su manejo creando una función miembro dentro de una clase que la incluya.
  - Sintaxis:
    - $variable con el resultado de una consulta = mysqli_query ($variable de conexión, $variable con consulta);
    - mysqli_num_rows($variable con el resultado de una consulta);
    - **Método si se usa en una clase**:

```
public function nombreMetodo(){
        return mysqli_num_rows ($this->variable de resultado);
}
```

  - Ejemplos:

- **Ejemplo 1**:
  - $resultado = mysqli_query($conexion, $consulta); // En la variable *$resultado* se cargan los resultados obtenidos por una consulta.
  - $registro = mysqli_num_rows($resultado); // En la variable *$registro* se carga el número de filas que se han encontrado en la consulta.
- **Ejemplo 2**: (Como función en una clase).

```
public function consultadas () {
return mysqli_num_rows ($this->resultado);
}; // Función creada dentro de la definición de una clase.

Coches-> consultarRegistros ('select * from concesionario'); // Llamada al método consultarRegistros() a través de objeto Coches, que realiza una consulta sobre una determinada conexión. Dicho método deberá estar creado en la clase.

$num_consultadas = Coches->cantidadFilas(); // En la variable $num_consultadas se carga el número de filas que se han obtenido con la consulta. Para ello, se llama al método cantidadFilas() a través el objeto Coches. Este método, incluye en su definición a la función mysqli_num_rows(), que retorna la cantidad de registros coincidentes con la consulta realizada.
```

- **mysqli_affected_rows().**
  - Permite obtener el número de filas que han sido afectadas después de un cambio en una tabla (eliminación, inserción o modificación).
  - El resultado obtenido puede cargarse en una variable y posteriormente ésta ser usada en una condición.
  - Puede facilitarse su manejo creando una función miembro dentro de una clase que la incluya.
  - Sintaxis:
    - $variable con el resultado de una operación LMD = mysqli_query ($variable de conexión, $variable con consulta LMD); // En la variable para el resultado se cargan los resultados obtenidos por la modificación.
      $variable_filas_afectadas = mysqli_affected_rows ($variable con el resultado de una operación LMD); // En la variable se carga el número de filas que han sufrido cambios con una operación LMD.
    - **Método si se usa en una clase**:

```
public function nombreMetodo(){
        return mysqli_affected_rows ($this->variable de conexión);
```

}

- Ejemplos:
  - **Ejemplo 1**:
    - mysqli_query ($conexion, $cambios); // Se ejecuta la instrucción LMD incluida en la variable *$cambios*.
    - $afectadas = mysqli_affected_rows($conexion); // En la variable *$afectadas* se carga el número de filas que han sufrido cambios con una operación LMD.
  - **Ejemplo 2**: (Como función en una clase).

```
public function afectadas () {
return mysqli_affected_rows ($this->conexion);
}; // Función creada dentro de la definición de una clase.
```

```
coches->insertarRegistros('insert into coches values ("123ER","SEat","Ibiza2",14000.00)');
```

// Llamada al método insertarRegistro(), que realiza la operación LMD de insertar un registro sobre una determinada conexión. Dicho método deberá estar creado en la clase. Puede haer otros métodos para eliminar o actualizar registros, o uno común para todas estas operaciones.

```
$afectadas = Clientes->cantidadInsertadas();
```

// En la variable *$afectadas* se carga el número de filas que han sufrido cambios o modificaciones al ejecutar el método cantidadInsertadas() a través del objeto Coches. El valor de *$afectadas* se enviará a una web o se guardará en otro lugar. El método cantidadInsertadas() incluye la función mysqli_affected_rows(), por lo que en realidad retorna la cantidad de las filas afectadas por la operación LMD que se haya utilizado.

- **mysqli_close()**
  - Sirve para cerrar una conexión a una base de datos.
  - Sintaxis:
    - mysqli_close (variable de conexión);
    - La variable de conexión será aquella en la que se ha cargado la función mysqli_connect().
  - Ejemplo:
    - mysqli_close ($conexion);
- **mysqli_fetch_assoc().**
  - Obtiene una fila de resultados como un array asociativo.
  - Tras una consulta, si se obtienen varias filas o registros, se puede cargar cada una de ellas en una variable usando este método dentro de un bucle while o do…while. Así, dentro del bucle se van recorriendo todas las filas de resultados.

- Posteriormente se puede acceder a cada dato guardado con el correspondiente valor asociativo.
- Tras ejecutar una consulta con mysqli_query() y cargarla en una variable, un resultado se puede luego cargar en otra variable con fetch_assoc(), que se convertirá en un array asociativo.
- Sintaxis:
  - mysqli_fetch_assoc (variable con una fila cargada)
- Ejemplo:
  - mysqli_fetch_assoc ($resultado);

Conexión usando una clase reutilizable.

- Se puede crear una clase con todas las características necesarias para establecer una conexión de base datos para todos los proyectos o aplicaciones que lo necesiten.
- Dicha clase contendrá las propiedades, métodos o funciones que se necesiten y que al menos serán las siguientes:
  - **Propiedades o variables.**
    - Una variable para incluir los datos del servidor (IP, URL, localhost, etc.).
    - Una variable para el identificador del usuario.
    - Una variable para la contraseña.
    - Una variable para el nombre de la base de datos.
    - Una variable para cargar el método mysqli_connect() con todos los datos anteriores.
  - **Métodos o funciones.**
    - Un constructor para cargar los datos en las propiedades cuando se cree el objeto basado en la clase.
    - Si no se añade un constructor, se incluye uno vacío por defecto, por lo que habrá que asignar los valores de las propiedades a través del objeto creado. (objeto->propiedad).
    - Otra función para establecer la conexión usando el método mysqli_connect().
  - **Métodos adicionales**:
    - Si se quiere que la clase realice también consultas y devuelva los registros que coincidan con los criterios especificados en ellas, se pueden incluir las siguientes funciones:
      - Una función para usar el método de consulta mysqli_query().
      - Una función para comprobar si la consulta ha producido resultados y, si es así, extraer los registros en un array usando el método mysql_fetch_array().
- Sintaxis:

  class nombre de la clase {

```
[alcance] $variable1; // Propiedad de la clase usando variables.
[alcance]  $variable2;
[alcance]  $variableN;
function __construct($parámetro1, $parámetro2, …, $parámetroN)
{
        $this.variable1 = $parametro1;
        $this->variable2 = $parametro2;
        $this->variableN = $parametroN;
}

[alcance]  function nombre método 1([$parámetro 1,
$parametro2,…,parámetro N])
{
        Instrucciones;
        [return;]
} //Métodos que indican cual es el comportamiento de los objetos.

[alcance]  function nombre método 2([$parámetro 1,
$parametro2,…,parámetro N])
{
        Instrucciones;
        [return;]
}
[alcance]  function nombre método N([$parámetro 1,
$parametro2,…,parámetro N])
{
        Instrucciones;
        [return;]
}
// Método para cerrar la conexión.
[alcance]  function nombre método N()
{
        Método close();
}
}
```

- Ejemplo:
    - Ver ejemplo digital.

Utilización de la clase para conectarse a una base de datos.

- En el archivo en el que se realizará la conexión y las posteriores operaciones sobre una base de datos se necesitan las siguientes acciones:
    - Carga del archivo que contiene la clase usando los métodos:
        - include ();
        - include_once ();

        - requiere (); // Preferido para no continuar el script si se produce un error.
        - requiere_once (); // Preferido para no continuar el script si se produce un error.
    - Carga de datos de conexión:
        - **Opción 1**:
            - Definir e inicializar las variables que recogerán los datos de la conexión puntual que se quiera hacer en un momento dado.
        - **Opción 2**:
            - Incluir estos datos directamente en el método constructor de la clase creada para la conexión a la base de datos.
        - Independientemente de la opción elegida, los datos serán los siguientes:
            - Servidor al que nos conectaremos.
            - Usuario que realizará la conexión.
            - Su contraseña si la hay, si no, se incluye una cadena de longitud cero.
            - La base a la que se quiere conectar.
    - Creación de un objeto de conexión a base de datos:
        - Crear un objeto de conexión usando, en el constructor, los datos de conexión directos o las variables anteriores como parámetros para inicializar dicho objeto.
    - Operaciones de manipulación de datos:
        - Realizar las operaciones de consulta, inserción, eliminación o actualización de datos que se necesiten.
    - Cerrar la conexión a base de datos:
        - Llamar al método que cierra la conexión a través del objeto creado.

Confirmación de operaciones LMD.

- Si se consultan, borran, insertan o modifican registros, el resultado de dichas operaciones se puede visualizar en la propia base de datos (actualizando las tablas) o enviando el contenido a una página web.
- Dentro de una página web se puede visualizar el contenido dentro de cualquier etiqueta HTML: párrafos, listas tablas, etc.
- La página web HTML o PHP que se cargue tras la operación de modificación de una base de datos puede servir para ver los resultados de la modificación, confirmar o no si las operaciones se han realizado correctamente, dar acceso a un sitio web, etc.
- Para cargar una página web se pueden utilizar muchas funciones de manejo de archivos como include(), require() o header(), entre otras.

header().

- Tiene múltiples usos, como generar y enviar una cabecera HTTP, pero uno de las más comunes es redirigir automáticamente a otra página.
- header() debe usarse antes de mostrar nada en la página web. No puede haber otras etiquetas HTML que generen, ni contenido, ni espacios en blanco, es decir, no puede utilizarse una vez que se haya generado contenido HTML.
- Incluso si se generan etiquetas HTML o líneas en blanco desde archivos externos usando include() o require() u otras funciones de archivo, o desde PHP, header() debe llamarse antes que ellos.
- Sintaxis:
    - header ("Location:ruta de acceso/archivo.extensión");
        - La ruta de acceso puede ser una dirección absoluta o una relativa, tanto local como remota.
        - El archivo será la página web que se quiere cargar.
- Ejemplos:
    - <? php header("Location:http://www.misitio.com/"); ?> // Ruta absoluta, redirección a las páginas por defecto *index.html, index.php*, etc.
    - <? php header("Location:http://www.misitio.com/nueva.html"); ?> // Ruta absoluta, redirección a la página *nueva.html*.
    - <? php header("Location:index.html"); ?> // Ruta relativa, redirección a la página *index.html*, ubicada en la carpeta raíz del sitio.
    - <? php header("Location:paginas/resultados.html"); ?> // Ruta relativa, redirección a la página *resultados.html*, ubicada en la carpeta *páginas* del sitio.
- **Redireccionar con header() tras paso de tiempo.**
    - Se puede usar la función header() para redirigir a otra página transcurrido un tiempo determinado.
    - Sintaxis:
        - header ("Refresh: tiempo en segundos, URL= ruta de acceso/archivo.extensión");
    - Ejemplo:

```
header("Refresh: 10; URL=pagina2.php");
echo "<p>Redireccionando…</p>"; // Recargar la página después de 10 segundos y redireccionar a "pagina2.php"
```