PROGRAMACIÓN WEB EN ENTORNO SERVIDOR UD-13

BBDD EN PHP

ÍNDICE

[VARIABLES DE SERVIDOR 3](#_Toc147867406)

[A. $\_SERVER[] 3](#_Toc147867407)

[VALIDACIÓN DE DATOS 4](#_Toc147867408)

[TEXTO IMPRIMIBLE 4](#_Toc147867409)

[INTRODUCCIÓN DE <br /> 4](#_Toc147867410)

[FORMATEO SALIDA CON PRINTF 5](#_Toc147867411)

[FORMATEO SALIDA CON NUMBER\_FORMAT 5](#_Toc147867412)

[FUNCIONES Y CONSTANTES MATEMÁTICAS 6](#_Toc147867413)

# DISEÑO

Es fundamental realizar un buen diseño de la BD para que nuestro programa funcione correctamente.

Lo normal es representar los objetos como una entidad en el modelo ER, de forma que los atributos del objeto sean los atributos de la entidad.

Los datos que nos llegan a través de un formulario deberían ser comprobados antes de acceder a la BD. Es muy frecuente eliminar los espacios en blanco al principio y al final del valor pasado trim($\_GET[“nombre”]).

Hay que tener cuidados con los ataques por inyección SQL.

# CONEXIÓN: mysqli

Como en Java, PHP permite la conexión con multitud de SGBD. Nosotros utilizaremos MySQL, porque es gratuito. Para MySQL, podemos utilizar la biblioteca mysqli (BD relacional) o la extensión de abstracción de BD PDO (varias BD y OO).

Vamos a ver como funciona mysqli y luego pasaremos a PDO.

mysqli\_connect("SERVIDOR", "USUARIO", "CONTRASEÑA", "BASE DE DATOS");

new mysqli ("SERVIDOR", "USUARIO", "CONTRASEÑA", "BASE DE DATOS");

Con este comando creamos la conexión con nuestra BD de manera procedimental u orientada a objetos.

Para ejecutar una consulta usaremos el siguiente comando:

$array = mysqli\_query($strConexion, $strConsulta);

Que nos devuelve un array con los valores de la consulta.

Texto

Descripción generada automáticamente

Se pueden utilizar ? Dentro de la consulta, para meter datos en una consulta preconstruida. Con esto se evitan los ataques por inyección sql.

Captura de pantalla de computadora

Descripción generada automáticamente

# CONEXIÓN: PDO

PDO es el sistema más utilizado hoy en día, dado que permite la conexión con múltiples BD y el uso de orientación a objetos.

Para crear una conexión usaremos el comando

new PDO(“BD:host=ip;dbname=NombreBD", “usuario", “password");

new PDO("mysql:host=localhost;dbname=java","root","");

Con prepare, cargaré la sentencia y me devolverá el cursor.

$cursor = $conexion->prepare($strConsulta);

Puedo establecer el modo en que el cursor me devolverá los registros en arrays, utilizando el comando setFetchMode e indicando la constante que nos interese:

* PDO::FETCH\_ASSOC: array por referencia nombre\_atributo=>valor.
* PDO::FETCH\_BOTH: array por referencia y posicional.
* PDO::FETCH\_CLASS: crea un objeto de la clase indicada llamando al constructor. Los atributos de la clase y de la tabla se tienen que llamar igual.
* PDO::FETCH\_INTO: igual que el anterior, pero se le pasa un objeto ya creado que se actualiza.
* PDO::FETCH\_OBJ: se crea un objeto anónimo sin llamar al constructor de ninguna clase. El objeto tendrá atributos que se llamarán igual que los nombres dados por la tabla.
* PDO::FETCH\_PROPS\_LATE: se usa con PDO::FETCH\_CLASS, se llama al constructor antes de asignar las propiedades.

Texto

Descripción generada automáticamente

## INSERCIÓN, MODIFICACIÓN Y BORRADO: PDO

Además de consultas, se pueden crear tablas, modificar, insertar y borrar valores.

## CONEXIÓN: PDO

Se pueden pasar parámetros en la consulta SQL con bindParam(":parametro", $variable); que asociará la variable al valor para poder usar el parámetro en múltiples ocasiones. Se puede usar también bindValue, pero no se asocia a la variable.

## fetchAll: PDO

Si voy a pasar varias veces por los mismos datos, me puede interesar utilizar el comando fetchAll(), que devuelve un array con bidimensional con todos los registros;

## FETCH\_OBJ: PDO

Pero si lo que queremos es leer datos con forma de objeto utilizando PDO::FETCH\_OBJ, debemos crear un objeto con propiedades públicas con el mismo nombre que las columnas de la tabla que vayamos a consultar.

## FETCH\_CLASS: PDO

Otra manera de utilizar objetos es indicar en que clase los queremos dejar con PDO::FETCH\_CLASS. Tiene que estar creada la clase con los getters.

## FETCH\_PROPS\_LATE: PDO

Para inicializar los datos llamando al constructor de la clase, usaremos PDO::FETCH\_PROPS\_LATE, hay que tener el constructor en la clase y combinarlo con PDO::FETCH\_CLASS.

## FETCH\_PROPS\_LATE: PDO

Puedo usar también PDO::FETCH\_INTO, pero no tiene mucho sentido porque los atributos de la clase tienen que ser públicos