UNIVERSIDAD DE SONSONATE

Implantación de Sistemas

Guía 3 –Conexión a Base de datos MySQL

Instructor: Ing. David Gildardo Rajo

Facultad de Ingeniería y Ciencias Naturales

Introducción a las bases de datos – MySql

Haciendo uso de la herramienta de administración de bases de datos para MySQL de tu elección, o la utilizada en clase (phpMyAdmin), crea la base de datos llamada: **poma**

Crear cada una de las tablas mostradas en el **Diagrama 1 Base de Datos**

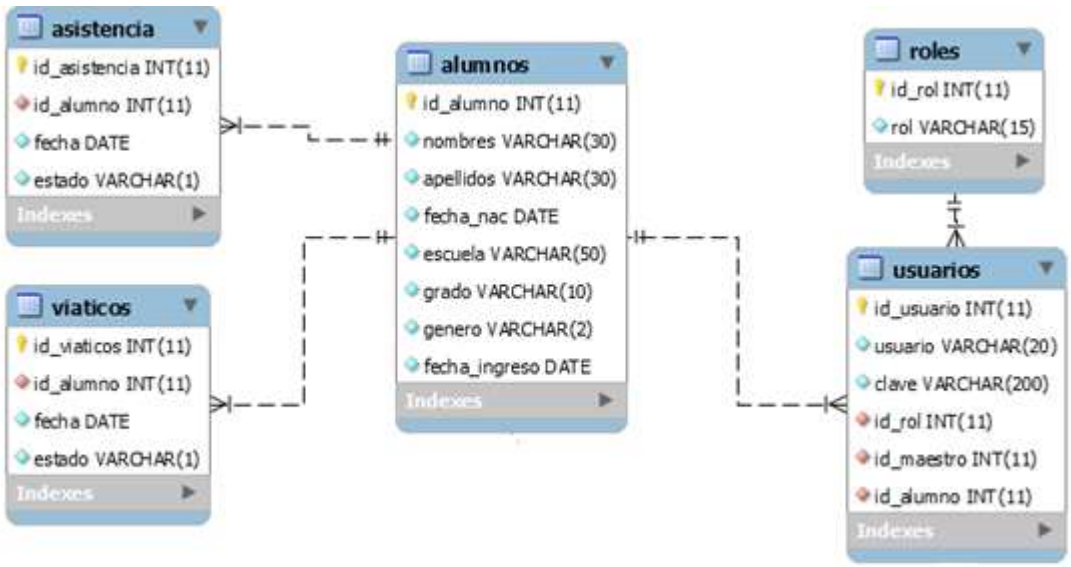
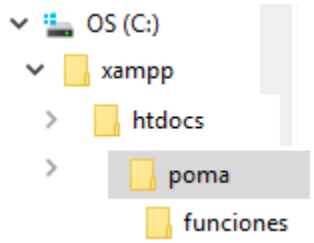


Diagrama 1Base de Datos

Fase II: Preparando conexión

Ahora que ya tenemos la base de datos para la aplicación creada, comenzaremos por preparar la conexión a nuestra base de datos.

Comenzaremos por crear una carpeta llamada **poma** en la que organizaremos los archivos de nuestro proyecto (recuerda esta debe estar dentro de la carpeta C:\xampp\htdocs\).



Dentro de la carpeta poma crearemos otra carpeta llamada **funciones**, en la que dejaremos los archivos necesarios para la conexión a la base de datos. La estructura nos quedará como se muestra.

Una vez lista la carpeta **funciones** crearemos 2 archivos

config_mysql.php

```php
<?php
    //Dirección del Servidor
    define("host","localhost");
    //Usuario con los permisos del servido
    define("user","root");
    //Contraseña del usuario del servidor
    define("pass","");
    //Nombre de la base de datos
    define("db","poma");
?>
```

El usuario con el que se establecerá la conexión al servidor. Por defecto cuando se instala xampp es **root**

La clave para acceder con el usuario establecido, para el usuario root por defecto no tiene clave por lo que se deja en blanco.

Nombre de la base de datos para este caso usaremos la que creamos en la fase I: **poma**

Este primer archivo nos sirve para centralizar las variables necesarias para realizar una conexión a la base de datos, el cual será incluido en las conexiones que realizaremos.

funciones_mysql.php

Funciones_mysql tendrá principalmente la función para realizar la conexión a la base de datos, para ello necesitaremos los datos de conexión que se encuentra en el archivo **config_mysql.php** por lo que se incluirá por medio de la directiva **include** (Un include es un elemento del lenguaje PHP que nos permite incluir una página entera dentro de otra)

Por lo que agregaremos:

include "config_mysql.php";

De esta forma las constantes que definimos en el archivo de configuración, las podremos utilizar como propias en este archivo.

A continuación, crearemos una función para realizar la conexión a la base de dato y poder reutilizarla las veces que sean necesario.

Para realizar una conexión a MySql lo haremos por medio de la clase **mysqli,** esta clase se le agregará 4 parámetros, el primero de la dirección donde nos conectaremos, el segundo el usuario, el tercero la clave, y el cuarto será la base de datos con la que vamos a trabajar (estos datos fueron asignados en el archivo anterior)

Si la conexión se realiza con éxito retornaremos la conexión en la variable **$dbconn**.

```
function conectar(){
                $dbconn = new mysqli(host,user,pass,db);

                return $dbconn;
}
```

Si quisiéramos verificar en caso de error en la conexión agregaremos las siguientes líneas antes de **return $dbconn;**

```
if ($dbconn->connect_errno) {
echo "Fallo al conectar a MySQL: (" . $dbconn->connect_errno . ") " . $dbconn->connect_error;
}
```

El código completo quedaría de la siguiente forma:

```php
<?php
    include "config_mysql.php";
    // Conectarse a la base de datos
    function conectar(){
        $dbconn = new mysqli(host,user,pass,db);
        if ($dbconn->connect_errno) {
            echo "Fallo al conectar a MySQL: (" . $dbconn->
                connect_errno . ") " . $dbconn->connect_error;
        }
        return $dbconn;
    }
    //$conexion = conectar();
    //echo $conexion->host_info;
?>
```

Si quieres verificar que lo anterior funciona quítales el comentario a las 2 últimas líneas y pruébalo en el navegador con la dirección **localhost/poma/funciones** y selecciona el archivo **funciones_mysql.php**

Si la conexión se realizó con éxito deberá aparecer el texto en el navegador: localhost via TCP/IP

Fase III: Preparando consultas a una tabla

Teniendo lista la conexión a la base de datos, la implementaremos para realizar las consultas a una tabla.

La tabla con la comenzaremos será la tabla rol, crearemos un archivo llamado **roles_op.php** dentro de la carpeta funciones.

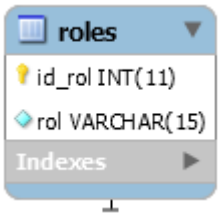
En este archivo agregaremos las funciones con las operaciones básicas que se realizan a las tablas, los llamados CRUD (Create, Read, Update, Delete) que hacen referencia a las acciones de crear, leer, modificar y eliminar datos de una tabla.

Para realizar las consultas de las tablas es necesario el conectarnos a la base de datos por lo que agregaremos a nuestro archivo el archivo de conexión, el que llamamos **funciones_mysql.php**

```
<?php
include "funciones_mysql.php";

?>
```

Tomando en cuenta la estructura de la tabla **roles**: tiene 2 campos, 1 **el id_roles** (llave principal, de tipo int, y con la propiedad Auto Increment -AI) y el 2 el campo **rol** (tipo varchar de longitud 15)



Insertar (CRUD)

Comenzaremos por crear la función que se encargará de insertar datos que enviemos a la tabla.

Dentro de archivo roles_op.php creamos la función llamada **insertar**.

Para realizar las operaciones a las tablas seguiremos 5 pasos:

1. Establecer Conexión
2. Definir sentencia SQL
3. Preparar la sentencia
4. Enlazar parámetros
5. Ejecutar la sentencia

Establecer Conexión (1)

Para establecer la conexión implementamos la función conectar que definimos en el archivo de funciones y la almacenamos en una variable.

```
$conexion = conectar();
```

Definir sentencia SQL (2)

Las sentencias SQL para insertar datos a una tabla en MySQL es:

```
INSERT INTO nombre_de_Tabla(campos) VALUES (valores);
```

Campos: lista de campos de la tabla a insertar separados por comas.

Valores: los valores que insertaremos separados por coma, en el orden en que se dejaron los campos. Los valores para ser agregados se le dejará con un cierre de signo de interrogación.

```
$sql = "INSERT INTO roles(rol) VALUES(?)";
```

En el caso de la table roles la lista de los campos será solo el campo rol, id_rol no se inserta puesto que al crear la tabla fue creada con este campo en auto_increment el cual incrementa de forma automática por lo que no hay necesidad de enviarle ese campo para ser insertado.

Preparar la sentencia (3)

El preparar la sentencia se realiza haciendo uso de un método que pertenece a la conexión, llamado prepare, la cual el resultado lo almacenaremos en una variable.

```
$stm = $conexion->prepare($sql);
```

Enlazar parámetros (4)

La acción de enlazar parámetros se realiza cuando a la sentencia le dejamos signos de cierre de interrogación los cuales son valores condicionales cuando se prepara una sentencia, para enlazar los parámetros el primer parámetro de tipo String (irá entre comillas) y representará los tipos de variables que serán enviados dependiendo de la cantidad de parámetros que se dejaron en la consulta (?), por ejemplo si en la consulta se definen 4 parámetros los primeros 3 son **varchar** y el último un **int** la cadena será “sssi”, seguido y separado por comas las variables que tendrán los valores para enviar a la consulta.

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

En el caso de roles que solo es un campo que enviar, quedará:

```
$stm->bind_param("s",$rol);
```

La variable $rol debe tener el valor que será enviado a la base de datos por lo que se captura su valor por medio de un $_REQUEST el cual será enviados desde el formulario.

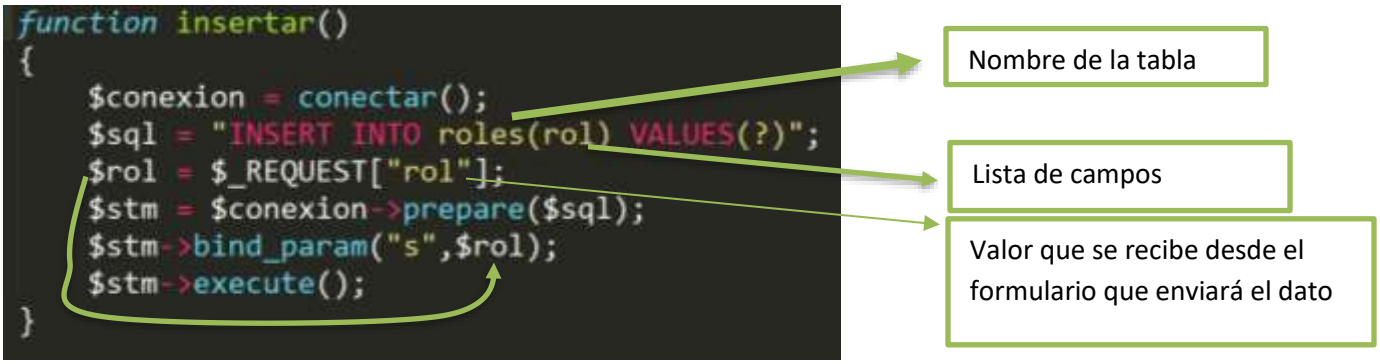
```
$rol = $_REQUEST["rol"];
```

Ejecutar la sentencia (5)

Hasta este punto no se ha realizado consulta a la base de datos, solo se ha preparado todo para ser enviada, para ejecutar la sentencia utilizaremos la siguiente sentencia.

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

El código completo quedará como se muestra a continuación:



Para usar la función insertar recibiremos la variable **op** que se encargará de verificar qué operación se realizará con la tabla (insertar, modificar, eliminar) El siguiente código lo agregarás antes o después de la función insertar, **no** adentro.

Para verificar su funcionamiento escribiremos en el navegador:

localhost/poma/funciones/roles_op.php?op?=insertar&rol=Admin

Si todo está bien deberá estar el rol “Admin” dentro de la tabla roles, compruébalo desde phpMyAdmin. (cada vez que recargues la página, guardará en la base de datos el valor, si no quieres datos duplicados solo pruébalo 1 vez si te ha funcionado)

Leer Datos (C**R**UD)

Continuaremos con la función que se encargará de mostrar los datos que tengamos en la tabla.

Dentro de archivo roles_op.php creamos la función llamada **mostrar**.

Para realizar las operaciones a las tablas seguiremos de nuevo los 5 pasos, con la diferencia que en esta operación recibiremos datos desde la tabla por lo que debemos enlazar los resultados devueltos y leerlos por lo que se agregaran 2 pasos más:

1. Establecer Conexión
2. Definir sentencia SQL
3. Preparar la sentencia
4. Enlazar parámetros
5. **Enlazar resultado**
6. Ejecutar la sentencia
7. **Mostrar resultados**

(1)

```
$conexion = conectar();
```

(2)

```
$sql = "SELECT id_rol, rol FROM roles";
```

(3)

```
$stm = $conexion->prepare($sql);
```

(4)
En el caso de la table roles no condicionaremos la búsqueda de datos por lo que no enlazaremos parámetros.

(5) **Enlazar resultado**

```
$stm->bind_result($id_roles,$roles);
```

Los datos que devuelve la consulta del paso (2) se enlazan con las variables aquí definida, por lo que debe coincidir la cantidad de variables con la lista de los campos.

(6)

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

(7) **Mostrar resultados**

Para mostrar el resultado de la table, se lee cada fila de datos que devuelve la consulta, haciendo uso de $stm->fetch() Como no sabemos cuántas filas son devueltas esta sentencia se utiliza dentro de un **while**, la estructura dependerá de cómo se desea mostrar los datos en nuestro caso lo mostraremos dentro de una tabla:

```
$html="<table border=1>";
while($stm->fetch())
{
                $html = $html . "<tr><td>". $id . "</td><td><input id='rol' value=" .  $rol . "></td> <td><button value='".$id."'>Modificar</button></td> <td><button value='".$id."'>Eliminar</button> </td> </tr>";

}
$html = $html . "</table>";
echo $html;
```

El código completo de la función mostrar queda de la siguiente manera:

```php
function mostrar()
{
    $conexion = conectar();
    $sql = "SELECT id_rol, rol FROM roles";
    $stm = $conexion->prepare($sql);
    $stm->bind_result($id_roles,$roles);
    $stm->execute();
    $html="<table border=1>";
    while($stm->fetch())
    {
        $html = $html . "<tr><td>". $id . "</td><td><input id='rol'
            value=" . $rol . "></td> <td><button value='".$id."'>
            Modificar</button></td> <td><button value='".$id."'>
            Eliminar</button> </td> </tr>";

    }
    $html = $html . "</table>";
    echo $html;
}
```

Para verificar su funcionamiento, primero debemos llamar a la función mostrar, agregando antes o después de la función, la siguiente línea:

mostrar();

Ahora escribiremos en el navegador:

localhost/poma/funciones/roles_op.php

Si todo está bien verás en el navegador la lista de los roles que se encuentra en la tabla con los bonotes para modificar y eliminar esos datos.

Si funcionó debes **comentar** o **eliminar** la línea:

mostrar();

Modificar (CRUD)

Crearemos la función que se encargará de modificar datos de la tabla.

Dentro de archivo roles_op.php creamos la función llamada **modificar**.

Para realizar las operaciones a las tablas seguiremos los 5 pasos.

(1)

```
$conexion = conectar();
```

(2)

```
sql = "UPDATE roles SET rol = ? WHERE id_rol=?";
```

(3)

```
$stm = $conexion->prepare($sql);
```

(4)

```
$id = $_REQUEST["id"];
$rol = $_REQUEST["rol"];
$stm->bind_param("si",$rol,$idrol);
```

(5)

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

Para usar la función modificar recibiremos la variable **op** que se encargará de verificar qué operación se realizará con la tabla (insertar, modificar, eliminar) agregaremos la segunda condición de op para que pueda modificar los datos.

```
if(isset($_REQUEST["op"])){
    if($_REQUEST["op"]=="insertar"){
        insertar();
        //mostrar();
        echo "<a href='roles.php'>Ir a roles</a> ";
    }
    if($_REQUEST[ "op"]=="modificar"){
        modificar();
        //mostrar();
        echo "<a href='roles.php'>Ir a roles</a> ";
    }
}
```

Solo se agrega este bloque dentro del código que ya tenemos.

Para verificar su funcionamiento escribiremos en el navegador:

localhost/poma/funciones/roles_op.php?op?=insertar&rol=Administrador&id=1

Si todo está bien deberá modificar el rol “Admin” que tiene id 1 por “Administrador” dentro de la tabla roles, compruébalo desde phpMyAdmin. (cada vez que recargues la página, modificará en la base de datos el valor)

EJERCICIOS.

1. Continuar los CRUD para las tablas restantes tomando en cuenta la relación que existe con las tablas.
2. Crear un formulario para la inserción y modificación de los datos.
3. Agregar en la tabla la opción de eliminar datos.