MANEJO DE BASES DE DATOS EN PHP

BASE DE DATOS (MYSQL)

Uno de los empleos principales de PHP es el acceso a una base de datos en el servidor. Las operaciones básicas se hacen empleando como lenguaje el SQL.

PHP implementa distintas funciones según la base de datos a emplear. Existen funciones actualmente para acceder a las siguientes servidores de bases de datos:

- MySQL
- Microsoft SQL Server
- Oracle
- PostgreSQL
- mSQL
- SQLlite

https://www.php.net/manual/es/mysqli.quickstart.dual-interface.php

https://cursos.mejorcodigo.net/article/pdo-vs-mysqli-cuales-son-las-diferencias-37

Para crear una base de datos, el Xampp instala también un programa codificado en PHP que nos permite interactuar con el MySQL. Este programa se llama PHPMyAdmin, el cual nos permite crear las bases de datos, tablas, índices, usuarios etc.)

Actividad 01

Los aprendices deberán ingresar a PHPMyAdmin ó HeidiSQL y crearán una base de datos llamada: *universidad*. Luego crearán una tabla llamada *alumno* con la siguiente estructura:

codigo int auto_increment primary key
nombre varchar(50)
mail varchar(70)
codigocurso int

La tabla almacenará datos de alumnos que desarrollarán cursos de programación en PHP, ASP y JSP.

El código del alumno es de tipo numérico (int) y al indicar que es auto_increment se generará automáticamente por el gestor de base de datos.

Los campos nombre y mail son de tipo varchar (podemos almacenar cualquier caracter) y por último el campo codigocurso representa el curso a tomar por el alumno (1=PHP, 2=ASP y 3=JSP) El campo clave de esta tabla es el código de alumno (es decir no podemos tener dos alumnos con el mismo código, no así el nombre del alumno que puede eventualmente repetirse)

En los próximos conceptos comenzaremos a ver como desde PHP podemos comunicarnos con la base de datos "universidad" y acceder a la tabla "alumno" para ejecutar los comandos SQL más comunes como pueden ser: select, insert, delete, update etc.

INSERT (ALTA DE REGISTROS EN UNA TABLA)

Luego de crear una base de datos y sus tablas (Vamos a trabajar con la base de datos ya creada: *universidad*, que contiene la tabla alumno), veremos como agregar registros. Para añadir datos en la tabla empleamos el comando SQL llamado insert. Necesitamos dos páginas para este proceso, una será el formulario de carga de datos y la siguiente será la que efectúe la inserción en la tabla.

Formulario de carga de datos - crear un archivo llamado: form_adicionar_alumno.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Adicionar Alumno</title>
</head>

<body>
    <h1>Alta de Alumnos</h1>
    <form action="procesarDatos/InsertarAlumno.php" method="post">
        Ingrese nombre:
        <input type="text" name="nombre">
        <br> Ingrese mail:
        <input type="text" name="mail">
        <br> Seleccione el curso:
        <select name="codigocurso">
            <option value="1">PHP</option>
            <option value="2">ASP</option>
            <option value="3">JSP</option>
        </select>
        <br>
        <input type="submit" value="Registrar">
    </form>
</body>

</html>
```

Cada opción tiene su respectivo valor (en este caso los números 1, 2 y 3) y los textos a mostrar PHP, ASP y JSP. El dato que se envía a la otra página es el código de curso (esto debido a que definimos la propiedad value).

Crearemos una carpeta llamada clases y guardaremos un archivo llamado Conexion.php que será la clase engargada de la conexión con la bases de datos y tendrá el siguiente código:

```php
<?php
class Conexion
{

    private $host, $user, $pass, $database;

    public function __construct()
    {
        $this->host = "localhost";
        $this->user = "root";
        $this->pass = "";
        $this->database = "universidad";
    }

    public function connect()
    {
        $conexion = mysqli_connect($this->host, $this->user, $this->pass, $this->database) or
die("Problemas con la conexión");
        return $conexion;
    }
}
```

La función mysqli_connect se conecta a una base de datos de tipo MySql, el primer parámetro es la dirección donde se encuentra el gestor de base de datos (en este caso en el mismo servidor por lo que indicamos esto con "localhost), el segundo parámetro es el nombre de usuario de la base de datos ("root" en nuestro caso, que es el usuario por defecto que crea MySql para el administrador), seguidamente indicamos la clave del usuario root (por defecto al instalar el Xampp se crea con clave vacía) y por último indicamos el nombre de la base de datos a conectarnos (en nuestro ejemplo ya creamos la base de datos llamada: *universidad* que tiene la tabla *alumnos*)

En caso de haber algún error en la llamada a la función la misma retorna false por lo que se ejecuta la instrucción seguida del operador or, en nuestro caso llamamos a la función die que detiene la ejecución del programa y muestra el mensaje por pantalla.

El paso más importante es la codificación del comando SQL insert(debemos llamar a la función mysqli_query pasando como primer parámetro la referencia a la conexion y el segundo parámetro es un string con el comando insert). Para ello crearemos un archivo (dentro de la carpeta *procesarDatos*) llamado: **procesarDatos/InsertarAlumno.php** el cual recibirá los datos del formulario:

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Insertar Alumno</title>
</head>
<body>
<?php
```

```
    // Se incluye la clase Conexion
    require_once('../clases/Conexion.php');
    // Se crea un objeto de la clase Conexion
    $connObj = new Conexion();
    // Se llama al método connect, el cual permitirá abrir una conexión a una base
de datos MySQL
    $conexion = $connObj->connect();

    // Se reciben los datos del formulario
    $nombre = isset($_POST['nombre']) ? $_POST['nombre'] : "";
    $mail = isset($_POST['mail']) ? $_POST['mail'] : "";
    $coidgocurso = isset($_POST['codigocurso']) ? $_POST['codigocurso'] : "";
    // Se arma la sentencia SQL
    $SQL = "INSERT INTO alumno (nombre,mail,codigocurso) VALUES
                        ('$nombre','$mail',$coidgocurso)";

    /* Se ejecuta el método que permite realizar una consulta a la BD.
       Se le envía como parámetros la conexión y la consulta */
    mysqli_query($conexion, $SQL) or die("Problemas en el INSERT" .
mysqli_error($conexion));

    // Se cierra la conexión
    mysqli_close($conexion);

    echo "El alumno fue dado de alta.";
?>
</body>
</html>
```

La sintaxis del comando insert es bastante sencilla, indicamos el nombre de la tabla y los campos de la tabla a cargar. Luego debemos indicar en el mismo orden los valores a cargar en cada campo (dichos valores los rescatamos del formulario anterior).

Los campos de tipo varchar SQL requiere que encerremos entre comillas simples, esto sucede para el nombre y el mail; en cambio, para el codigocurso esto no debe ser así.

En caso que MySql detecte un error, retorna false esta función, por lo que se ejecuta la instrucción posterior al OR, es decir la función **die** que mostrará el error generado por MySql llamando a la función mysqli_error().

Por último cerramos la conexión con la base de datos y mostramos un mensaje indicando que la carga se efectuó en forma correcta. Tener en cuenta que el campo código se generó en forma automática.

Si queremos controlar que el insert se efectuó en forma correcta podemos ingresar al PHPMyAdmin y seleccionar la tabla "alumnos", y en la pestaña "examinar" podremos ver los datos ingresados desde la página que creamos

Actividad 02.

Crear en la base de datos "*universidad*" otra tabla llamada "cursos". La estructura de esta segunda tabla debe ser:

```
codigo int primary key auto_increment
nombrecurso varchar(40)
```

Utilizar el PHPMyAdmin o HeidiSQL para la creación de esta tabla. Implementar las dos páginas necesarias para efectuar el alta de cursos. Un formulario para ingresar el nombre del curso (**form_adicionar_curso.html)** y otra página donde se efectuará el insert (**procesarDatos/InsertarCurso.php)**.

LISTADO (SELECCIÓN DE REGISTROS DE UNA TABLA)

Para recuperar datos desde MySQL debemos emplear el comando select:

```
select codigo,nombre,mail,codigocurso from alumnos
```

Debemos pasar desde PHP un string con este comando para que MySQL lo ejecute y retorne todas las filas de la tabla alumnos. Veremos entonces como recuperar los datos almacenados en la tabla alumnos de la base de datos "*universidad*".

Para visualizar los registros, crearemos un archivo llamado: **listar_alumnos.php**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Listar Alumnos</title>
</head>
<body>

<?php
// Se incluye la clase Conexion
require_once 'clases/Conexion.php';
// Se crea un objeto de la clase Conexion
$connObj = new Conexion();
// Se llama al método connect, el cual permitirá abrir una conexión a una base de
datos MySQL
$conexion = $connObj->connect();
$SQL = "SELECT codigo,nombre, mail, codigocurso FROM alumno";
```

```php
$registros = mysqli_query($conexion, $SQL) or
die("Problemas en el select:" . mysqli_error($conexion));

// Se recorren todos los registros de la Base de Datos
while ($reg = mysqli_fetch_array($registros)) {
    echo "Codigo:" . $reg['codigo'] . "<br>";
    echo "Nombre:" . $reg['nombre'] . "<br>";
    echo "Mail:" . $reg['mail'] . "<br>";
    echo "Curso:";
    switch ($reg['codigocurso']) {
        case 1:echo "PHP";
            break;
        case 2:echo "ASP";
            break;
        case 3:echo "JSP";
            break;
    }
    echo "<br>";
    echo "<hr>";
}

// Se cierra la conexión
mysqli_close($conexion);
?>
</body>
</html>
```

La primer parte es similar a lo visto hasta ahora, es decir nos conectamos con el servidor de base de datos y seleccionamos la base de datos *universidad*. En caso de haber codificado incorrectamente, el comando SQL select la función mysqli_query retorna false, por lo que se ejecuta el comando siguiente al operador OR, es decir la función **die**.

Si el comando SQL es correcto, en la variable $registros se almacena una referencia a los datos rescatados de la tabla alumnos. Ahora debemos ir mostrando registro a registro los datos extraídos:

```php
while ($reg = mysqli_fetch_array($registros)) {
```

Para rescatar registro a registro los datos obtenidos por el select debemos llamar a la función mysqli_fetch_array. Esta función retorna un vector asociativo con los datos del registro rescatado, o false en caso de no haber más registros. Es decir que si retorna un registro se almacena en el vector $reg y la condición del while se valida como verdadero y pasa a ejecutarse el bloque del while:

El bloque del while muestra el contenido del registro rescatado por la función mysql_fetch_array. Como vemos, para rescatar cada campo accedemos mediante el vector asociativo $reg indicando como subíndice un campo indicado en el select:$reg['codigo']

Cada vez que llamamos a la función mysqli_fetch_array nos retorna el siguiente registro.

Cuando debemos mostrar el curso mediante la instrucción switch, analizamos si tiene un 1,2 ó 3 y procedemos a mostrar el nombre del curso.

Para separar cada alumno en la página HTML disponemos el elemento "<hr>"

Actividad 03.

Realizar un programa que recupere los datos de la tabla cursos de la base de datos *universidad*. Mostrar el código de curso y su nombre.

CONSULTA (SELECCIÓN DE REGISTROS DE UNA TABLA)

El proceso de consulta de datos de una tabla es similar al del listado, la diferencia es que se muestra sólo aquel o aquellos que cumplen la condición por la que buscamos. Haremos un programa que nos permita consultar los datos de un alumno ingresando su mail para su búsqueda. Tengamos en cuenta que no puede haber dos alumnos con el mismo mail, por lo que la consulta nos puede arrojar uno o ningún registro. Debemos codificar un formulario para el ingreso del mail a consultar (**form_buscar_alumno.html**):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Buscar Alumno</title>
</head>
<body>
    <form action="procesarDatos/buscarAlumno.php" method="post">
        Ingrese el mail del alumno a consultar:
        <input type="text" name="mail">
        <br>
        <input type="submit" value="Buscar">
    </form>
</body>
</html>
```

Por otro lado tenemos el archivo "**buscarAlumno.php**" que se encarga de buscar el mail ingresado en el formulario:

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Buscar Alumno</title>
</head>
<body>
<?php
// Se incluye la clase Conexion
require_once '../clases/Conexion.php';
// Se crea un objeto de la clase Conexion
$connObj = new Conexion();
// Se llama al método connect, el cual permitirá abrir una conexión a una base de
datos MySQL
$conexion = $connObj->connect();

$mail = isset($_POST["mail"]) ? $_POST["mail"] : "";
$SQL = "SELECT codigo,nombre, codigocurso FROM alumno WHERE mail='$mail'";

$registros = mysqli_query($conexion, $SQL) or
die("Problemas en el select:" . mysqli_error($conexion));

if ($reg = mysqli_fetch_array($registros)) {
    echo "Nombre:" . $reg['nombre'] . "<br>";
    echo "Curso:";
    switch ($reg['codigocurso']) {
        case 1:echo "PHP";
            break;
        case 2:echo "ASP";
            break;
        case 3:echo "JSP";
            break;
    }
} else {
    echo "No existe un alumno con ese mail.";
}

mysqli_close($conexion);
?>
</body>
</html>
```

Lo más importante está en el comando select:

```php
$SQL = "SELECT codigo,nombre, codigocurso FROM alumno WHERE mail='$mail'";

$registros = mysqli_query($conexion, $SQL) or
die("Problemas en el select:" . mysqli_error($conexion));
```

Acá es donde con la clausula where seleccionamos sólo el registro que cumple con la condición que el mail sea igual al que ingresamos. Como sólo puede haber un registro que cumpla la condición, llamamos a la función mysqli_fetch_array en un if y no una estructura repetitiva como el listado:

```
if ($reg = mysqli_fetch_array($registros))
```

En caso de retornar un vector asociativo la condición del if se verifica como verdadera y pasa a mostrar los datos, en caso de retornar false se ejecuta el else.

Actividad 04.

Realiar un programa que permita ingresar el nombre de un alumno en un formulario, luego mostrar los datos del mismo (tener en cuenta que puede haber más de un alumno con el mismo nombre)

DELETE (BAJA DE UN REGISTRO EN UNA TABLA)

El objetivo de este punto es el borrado de un registro de una tabla. Para ello, implementaremos un algoritmo que solicite ingresar el mail de un alumno y posteriormente efectúe su borrado. Para eliminar filas en una tabla debemos utilizar el comando SQL delete. La primer página es idéntica a la consulta, ya que debemos implementar un formulario que solicite la carga del mail del alumno (**form_eliminar_alumno.html**):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Eliminar Alumno</title>
</head>
<body>
    <form action="procesarDatos/eliminarAlumno.php" method="post">
        Ingrese el mail del alumno a borrar:
        <input type="text" name="mail">
        <br>
        <input type="submit" value="buscar">
    </form>
</body>
</html>
```

Por otro lado tenemos el archivo "**eliminarAlumno.php**" que se encarga de buscar el mail ingresado en el formulario y en caso que exista se procede a borrarlo:

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Elminar alumno</title>
</head>
<body>
<?php
// Se incluye la clase Conexion
require_once '../clases/Conexion.php';
// Se crea un objeto de la clase Conexion
$connObj = new Conexion();
// Se llama al método connect, el cual permitirá abrir una conexión a una base de
datos MySQL
$conexion = $connObj->connect();

$mail = isset($_POST["mail"]) ? $_POST["mail"] : "";
$SQL = "SELECT codigo FROM alumno WHERE mail='$mail'";

$registros = mysqli_query($conexion, $SQL) or
die("Problemas en el select:" . mysqli_error($conexion));
if ($reg = mysqli_fetch_array($registros)) {
    mysqli_query($conexion, "DELETE FROM alumno WHERE mail='$mail'") or
    die("Problemas en el select:" . mysqli_error($conexion));
    echo "Se efectuó el borrado del alumno con dicho mail.";
} else {
    echo "No existe un alumno con ese mail.";
}
mysqli_close($conexion);
?>
</body>
</html>
```

En esta segunda página efectuamos dos llamadas a la función mysqli_query, una para consultar si existe el mail ingresado y otra para efectuar el borrado del registro respectivo. Si no existe el mail ingresado mostramos un mensaje de tal situación.

Actividad 05.

Realizar un programa que permita ingresar el nombre de un curso por teclado y posteriormente efectúe el borrado de dicho registro en la tabla cursos. Mostrar un mensaje si no existe el curso.

DELETE (BAJA DE TODOS LOS REGISTROS DE UNA TABLA)

Vimos en el concepto anterior que podemos borrar una o más filas de una tabla indicando en el where del comando SQL delete la condición que debe cumplir la fila para ser borrada. Para borrar todos los registros de una tabla debemos llamar al comando delete de SQL sin disponer la cláusula where.

Es importante tener en cuenta que la ausencia de la cláusula where en el comando delete eliminará todas las filas de la tabla.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Eliminar Alumnos</title>
</head>
<body>
<?php
// Se incluye la clase Conexion
require_once 'clases/Conexion.php';
// Se crea un objeto de la clase Conexion
$connObj = new Conexion();
// Se llama al método connect, el cual permitirá abrir una conexión a una base de
datos MySQL
$conexion = $connObj->connect();

mysqli_query($conexion, "DELETE FROM alumno") or die("Problemas en el select:" .
mysqli_error($conexion));
echo "Se efectuó el borrado de todos los alumnos.";
mysqli_close($conexion);
?>
</body>
</html>
```

Actividad 06.

Efectuar el borrado de todos los registros de la tabla cursos.

UPDATE (MODIFICACIÓN DE UN REGISTRO DE UNA TABLA)

De las actividades con una tabla esta es la más larga. Vamos a resolverlo implementando tres páginas, la primera un formulario de consulta del mail de un alumno, la segunda otro formulario que nos permita cargar su mail modificado y la última registrará el cambio en la tabla.

El formulario de consulta del mail del alumno es similar a problemas anteriores **(form_editar_alumno.html)**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Editar Alumno</title>
</head>
<body>
    <form action="procesarDatos/editarAlumno.php" method="post">
        Ingrese el mail del alumno a consultar:
        <input type="text" name="mail">
        <br>
        <input type="submit" value="buscar">
    </form>
</body>
</html>
```

La segunda página tendrá el siguiente código (**editarAlumno.php)**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Actualizar Alumno</title>
</head>
<body>
<?php
    // Se incluye la clase Conexion
    require_once '../clases/Conexion.php';
    // Se crea un objeto de la clase Conexion
    $connObj = new Conexion();
    // Se llama al método connect, el cual permitirá abrir una conexión a una base
de datos MySQL
    $conexion = $connObj->connect();

    $mail = isset($_POST["mail"]) ? $_POST["mail"] : "";
    $SQL = "SELECT * FROM alumno WHERE mail='$mail'";
```

```
    $registros = mysqli_query($conexion, $SQL) or die("Problemas en el select:" .
mysqli_error($conexion));
    if ($reg = mysqli_fetch_array($registros)) {
?>
    <form action="actualizarAlumno.php" method="post">
        Ingrese nuevo mail:
        <input type="text" name="mailnuevo" value="<?php echo $mail; ?>">
        <br>
        <input type="hidden" name="mailviejo" value="<?php echo $mail; ?>">
        <input type="submit" value="Actualizar">
    </form>
<?php
    } else {
        echo "No existe alumno con dicho mail";
    }
?>
</body>
</html>
```

Lo primero que podemos observar es que si el if se verifica verdadero se ejecuta un bloque que contiene código HTML:

```
if ($reg = mysqli_fetch_array($registros)) {
?>
    <form action="actualizarAlumno.php" method="post">
        Ingrese nuevo mail:
        <input type="text" name="mailnuevo" value="<?php echo $mail; ?>">
        <br>
        <input type="hidden" name="mailviejo" value="<?php echo $mail; ?>">
        <input type="submit" value="Actualizar">
    </form>
<?php
    }
```

Es decir que podemos disponer bloques de PHP dispersos dentro de la página. Otro concepto importante es como enviar el mail del primer formulario a la tercer página, esto se logra con los controles de tipo "hidden", este tipo de control no se visualiza en el formulario pero se envía al presionar el botón submit.

Si queremos que el control text se cargue con el mail ingresado en el formulario anterior debemos cargar la propiedad value con dicho valor:

```
<input type="text" name="mailnuevo" value="<?php echo $mail; ?>">
```

Por último el archivo **actualizarAlumno.php** es el que efectúa la modificación de la tabla propiamente dicha. Con el mail ingresado en la página **form_editar_alumno.html**, el mail modificado en **editarAlumno.php** se efectúa el update.

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Actualizar Alumno</title>
</head>
<body>
<?php
    // Se incluye la clase Conexion
    require_once '../clases/Conexion.php';
    // Se crea un objeto de la clase Conexion
    $connObj = new Conexion();
    // Se llama al método connect, el cual permitirá abrir una conexión a una base
de datos MySQL
    $conexion = $connObj->connect();

    $mailnuevo = isset($_POST["mailnuevo"]) ? $_POST["mailnuevo"] : "";
    $mailviejo = isset($_POST["mailviejo"]) ? $_POST["mailviejo"] : "";
    $SQL = "UPDATE alumno SET mail='$mailnuevo' WHERE mail='$mailviejo'";
    mysqli_query($conexion, $SQL) or die("Problemas en el select:" .
mysqli_error($conexion));
    echo "El mail fue modificado con exito";
?>
</body>
</html>
```

Tengamos en cuenta que el segundo formulario nos envía dos datos: $_POST[mailnuevo] y $_POST[mailviejo].

Actividad 07.

- Permitir que en la edición de un alúmno se permita modificar también el nombre del alumno
- Efectuar la modificación del nombre del curso de la tabla "cursos". Para la búsqueda ingresar el código de curso.

INSERT (Y CONSULTA DE OTRA TABLA)

En un primer paso vimos como efectuar el alta en la tabla alumnos, ahora vamos a ver como resolver el problema del alta de un alumno seleccionando el curso de la tabla "cursos".

Es decir, el formulario de carga de datos no es HTML puro ya que debemos cargar el control "select" con los datos de la tabla cursos.

Para ello crearemos un nuevo archivo llamado: **form_adicionar_alumno_curso.php** con el siguiente código:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Insertar Alumno</title>
</head>
<body>
    <form action="procesarDatos/insertarAlumno.php" method="post">
        Ingrese nombre:
        <input type="text" name="nombre"><br>
        Ingrese mail:
        <input type="text" name="mail"><br>
        Seleccione el curso:
        <select name="codigocurso">
            <?php
            // Se incluye la clase Conexion
            require_once '../clases/Conexion.php';
            // Se crea un objeto de la clase Conexion
            $connObj = new Conexion();
            // Se llama al método connect, el cual permitirá abrir una conexión a
una base de datos MySQL
            $conexion = $connObj->connect();

            $registros = mysqli_query($conexion, "SELECT codigo,nombrecurso FROM
cursos") or
            die("Problemas en el select:" . mysqli_error($conexion));
            while ($reg = mysqli_fetch_array($registros)) {
                echo "<option
value='".$reg['codigo']."'>".$reg['nombrecurso']."</option>";
            }
            ?>
        </select>
        <br>
        <input type="submit" value="Registrar">
    </form>
</body>
</html>
```

El algoritmo es similar a cuando trabajamos con una tabla, pero el control "select" lo cargamos con los datos de la tabla "cursos":

```
while ($reg = mysqli_fetch_array($registros)) {
   echo "<option value='".$reg['codigo']."'>".$reg['nombrecurso']."</option>";
}
```

Dentro del while generamos todas las opciones que contiene el "select" imprimiendo el campo nombrecurso y asociando el campo codigo a la propiedad value(que es en definitiva el código que necesitamos rescatar en la otra página)

La página que efectúa el insert es exactamente la misma que vimos anteriormente (**procesarDatos/insertarAlumno.php**), por lo tanto no hay que crear una nueva ni modificar la existente:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
<?php
    // Se incluye la clase Conexion
    require_once('../clases/Conexion.php');
    // Se crea un objeto de la clase Conexion
    $connObj = new Conexion();
    // Se llama al método connect, el cual permitirá abrir una conexión a una base
de datos MySQL
    $conexion = $connObj->connect();

    // Se reciben los datos del formulario
    $nombre = isset($_POST['nombre']) ? $_POST['nombre'] : "";
    $mail = isset($_POST['mail']) ? $_POST['mail'] : "";
    $coidgocurso = isset($_POST['codigocurso']) ? $_POST['codigocurso'] : "";
    // Se arma la sentencia SQL
    $SQL = "INSERT INTO alumno (nombre,mail,codigocurso) VALUES
                        ('$nombre','$mail',$coidgocurso)";

    /* Se ejecuta el método que permite realizar una consulta a la BD.
       Se le envía como parámetros la conexión y la consulta */
    mysqli_query($conexion, $SQL) or die("Problemas en el INSERT" .
mysqli_error($conexion));

    // Se cierra la conexión
    mysqli_close($conexion);

    echo "El alumno fue dado de alta.";
?>
</body>
</html>
```

Actividad 08.

Realizar el alta de la tabla alumnos empleando controles de tipo "radio" para la selección del curso.

Actividad 09.

Crear un menú principal desde el cual se pueda ir a todas las funcionalidades del sistema. Este menú se deberán incluir en un único archivo en incluirlo donde se requiera con un require_once

Actividad 10.

Los aprendices deberán realizar los formularios correspondientes al CRUD de 3 tablas que se deban administrar del proyecto formativo y gestionar información de las mismas