PROCESO DIRECCIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE

1. IDENTIFICACIÓN DE LA GUIA DE APRENIZAJE

- Denominación del Programa de Formación: TÉCNICO EN PROGRAMACION DE SOFTWARE
- Código del Programa de Formación: 228120
- Nombre del Proyecto: CODIFICACION DE APLICACIONES DE SOFTWARE PARA EL SECTOR EDUCATIVO Y EMPRESARIAL DEL DEPARTAMENTO DEL QUINDIO
- Fase del Proyecto: EJECUCION
- Actividad de Proyecto: Codificar y Presentar la Matriz CRUD, teniendo en cuenta el mapa de navegación
- Competencia: 220501007 - CONSTRUIR EL SISTEMA QUE CUMPLA CON LOS REQUISITOS DE LA SOLUCIÓN
- Resultados de Aprendizaje Alcanzar: 22050100704 - CONSTRUIR LA MATRIZ CRUD EN EL LENGUAJE DE PROGRAMACIÓN SELECCIONADO PARA VERIFICAR LA FUNCIONALIDAD DEL SISTEMA DE ACUERDO CON EL DISEÑO ENTREGADO
- Duración de la Guía: 60 horas

2. PRESENTACION

Cordial Saludo Estimados Aprendices, en esta actividad de aprendizaje vamos a ver cómo crear un **CRUD en PHP** (create, read, update, delete), dedicado aquellos que están iniciándose en la programación Web con PHP.

Esta Guía de Aprendizaje es algo muy sencillo, lo que se pretende con ella, es dejar sentado las bases con las que funciona PHP en cuanto al acceso y manipulación de datos, además mostrar de lo sencillo que es programar con este lenguaje.

Para finalizar con éxito el ejemplo, debe tener un editor como puede ser Eclipse-PHP o cualquier otro editor de código para PHP, en mi caso utilizo Sublime Text 2, además necesitas un entorno que te permita desarrollar aplicaciones para PHP, estos pueden ser: AppServ, XAMPP o WampServer que es el que he utilizado en este ejemplo.

En esta guía de formación exploramos la manera en la que puedes usar la base de datos MySQL con PHP. El propósito es brindar conocimientos básicos sobre la conectividad a las bases de datos de MySQL con PHP. Veremos cómo ir desde la configuración de una conexión a una base de datos hasta la ejecución de diferentes tipos de consultas.

INTRODUCCION

Un desencadenador es un tipo especial de procedimientos almacenados que entra en vigor cuando se modifican datos en una tabla especificada utilizando una o más operaciones de modificación de datos: UPDATE (actualización), INSERT (inserción) o DELETE (eliminación). Los desencadenadores pueden consultar otras tablas e incluir instrucciones SQL complejas. Son especialmente útiles para exigir reglas o requisitos complejos. Por ejemplo, se puede controlar que se permita insertar un pedido basándose en el estado de cuenta actual de un cliente.

Los desencadenadores también son útiles para exigir la integridad referencial, que conserva las relaciones definidas entre tablas cuando se agregan, actualizan o eliminan filas en esas tablas. No obstante, la mejor manera de exigir la integridad referencial es definiendo restricciones de clave principal y de clave externa en las

tablas relacionadas. Si se utilizan diagramas de bases de datos, se puede crear relaciones entre tablas para crear automáticamente una restricción de clave externa.

Ventajas de utilizar desencadenadores

Los desencadenadores tienen varias utilidades:

- Los desencadenadores son automáticos: se activan inmediatamente después de que se efectúen modificaciones en los datos de la tabla, como una entrada manual o una acción de la aplicación.
- Pueden realizar cambios en cascada a través de tablas relacionadas de la base de datos.
- Los desencadenadores pueden hacer referencia a columnas de otras tablas. Por ejemplo, un desencadenador puede deshacer actualizaciones que intenten aplicar descuentos (almacenados en la tabla discounts) a libros (almacenados en la tabla titles) con un precio inferior a $10.

*Las tablas **INSERTED** (se genera con las instrucciones INSERT y UPDATE) y **DELETED** (se genera con las instrucciones UPDATE, DELETE) hacen referencia, al interior de un Trigger, a la tabla en mención y/o modificada.*

Creación:
Sintaxis (Simplificada):

CREATE TRIGGER <nombre_trigger> ON <nombre_tabla>
INSTEAD OF |FOR [INSERT,DELETE,UPDATE]

AS

<conjunto_Instrucciones>

Donde:

FOR: Después de la actualización de la (s) tablas

INSTEAD OF: Antes de la actualización de la (s) tablas. 2

Modificación:
Sintaxis:

ALTER TRIGGER <nombre_trigger> ON <nombre_tabla>
INSTEAD OF |FOR [INSERT,DELETE,UPDATE]

AS

<conjunto_instrucciones>

Eliminación:
Sintaxis:

DROP TRIGGER <nombre_desencadenador>

Activar/Desactivar Desencadenador en una tabla:
Sintaxis:

ALTER TABLE <nombre_tabla>

ENABLE|DISABLE <nombre_desencadenador>

Nota:

Para mostrar los desencadenadores de una tabla, se da la siguiente instrucción, desde el analizador de consultas, por ejemplo:

sp_helptrigger conceptos

Para mostrar el texto de un trigger, se da la siguiente instrucción, desde el analizador de consultas, por ejemplo:

sp_helptext nvoconcepto

Ejemplos:

CREATE TRIGGER nuevoconcepto ON
conceptos FOR INSERT

AS

Print “Se adicionó un Concepto”

CREATE TRIGGER actconcepto ON conceptos
FOR update

AS

Print “Se modificó un concepto”

CREATE TRIGGER borrarconcepto ON conceptos
FOR delete

AS

Print “Se eliminó un concepto”

CREATE TRIGGER adicconceptos2 ON Conceptos
FOR INSERT

AS

print "se adicionó tambien en conceptos1"

insert into conceptos1 select codigoc,descripcionc from inserted 3

CREATE TRIGGER BorrarConcepto ON Conceptos
FOR DELETE

AS

delete from conceptos1

where conceptos1.codigoc in (select codigoc from deleted)
CREATE TRIGGER actulizaconcepto ON Conceptos

FOR
UPDATE
AS

update conceptos1 set descripcionc = (select descripcionc from inserted)
where conceptos1.codigoc in (select codigoc from inserted)

3. FORMULACION DE LAS ACTIVIDADES DE APRENDIZAJE

ACTIVIDADES DE REFLEXION PARA LA COMPETENCIA: Tiempo Establecido para la actividad 2 horas

Existen operaciones que se deben ejecutar automáticamente luego de una transacción o ejecución de un comando SQL, tales como la deducción de la cantidad de un artículo que se acaba de vender. ¿Podrían un desencadenador o Trigger realizar esta tarea?

ACTIVIDADES DE CONTEXTUALIZACION: Tiempo establecido para la actividad 4 horas.

Tarea de Aprendizaje

1. Qué es un desencadenador?
2. Dónde se almacenan los desencadenadores?
3. Cuál es el objetivo de los desencadenadores?
4. Cuáles son las acciones para que un desencadenador se ejecute?
5. Qué es INSERTED y DELETED en un desencadenador?
6. En qué momento se crea INSERTED y DELETED?
7. Difiera entre procedimientos almacenados y desencadenadores
8. Qué hace SP_HELPTRIGGER?
9. Consulte acerca de los tipos de triggers o desencadenadores existen y cuáles son sus limitaciones.
10. Definir y crear los triggers para su proyecto: Sistema de Control y Administración de Votaciones. Complementar

ACTIVIDADES DE APROPIACION: Tiempos establecido para la actividad 24 horas

Descripción de la Actividad:

Realice la Actividad de Aprendizaje Propuesta

ACCION INTERPRETATIVA:

Actividad de Aprendizaje:

En esta actividad vamos a explorar cómo puedes usar una base de datos **MySQL** para llevar a cabo operaciones **CRUD** (crear, leer, actualizar y borrar, o create, read, update y delete en inglés) con PHP. La cual pretender guiarlos en el trabajo con conectividad de bases de datos en PHP.

Estamos iniciando con PHP, probablemente te hayas dado cuenta de que la conectividad de bases de datos es una característica esencial con la que necesitarás familiarizarte tarde o temprano. En la mayoría de los casos, una base de datos es la columna vertebral de cualquier aplicación web y contiene los datos de la misma. Por lo tanto, como desarrollador de PHP necesitarás saber cómo tratar con las operaciones en la base de datos.

En esta guía de aprendizaje mantendremos las cosas simples y exploraremos cómo usar las funciones básicas de mysqli. En futuros artículos de esta serie, exploraremos un par de formas diferentes para gestionar la conectividad de las bases de datos.

Hoy revisaremos los fundamentos de la conectividad de bases de datos en PHP, y usaremos la base de datos MySQL como nuestro back-end de base de datos. Echemos un vistazo rápido a lo que vamos a cubrir en este artículo:

- Cómo crear una conexión a una base de datos
- Cómo seleccionar una base de datos
- Cómo insertar y actualizar registros
- Cómo obtener registros
- Cómo borrar registros

Vamos a suponer que tienes una instalación funcional de PHP y de MySQL, y que conoces los fundamentos de MySQL. Además, debes asegurarte de que la extensión mysqli esté habilitada en tu instalación de PHP, ya que la usaremos para llevar a cabo operaciones en la base de datos MySQL.

Si no estás seguro de la extensión mysqli, puedes revisar usando la función `phpinfo()`. En la salida de `phpinfo()` puedes verificar si hay una sección titulada **mysqli**. También deberás ver el encabezado **Mysqli Support | enabled** (Soporte de Mysqli | habilitado) si la extensión mysqli está habilitada.

También hay una manera fácil de verificar usando la interfaz CLI. Puedes ejecutar el siguiente comando para mostrar todas las extensiones que estén habilitadas en tu instalación de PHP.

```
1  $php -m
```

Este debe imprimir una lista de extensiones, y si contiene la palabra clave **mysqli** significa que dicha extensión está habilitada.

Ahora que estamos listos, vamos a comenzar a crear una conexión a una base de datos.

Cómo establecer una conexión a una base de datos

En esta sección discutiremos cómo puedes establecer una conexión a una base de datos usando la extensión mysqli.

Existen dos formas en las que puedes usar la extensión mysqli, que son por procedimientos y orientada a objetos, pero usaremos la manera por procedimientos en esta publicación para mantener las cosas simples. Si sientes curiosidad acerca de la sintaxis orientada a objetos, hazme llegar tus preguntas en la sección de comentarios y estaré encantado de responderlas.

Primero revisemos la sintaxis de la función `mysqli_connect`, que se usa para establecer una conexión con el back-end de MySQL.

```
<?php
$connection_obj = mysqli_connect("{MYSQL_HOSTNAME}", "{MYSQL_USERNAME}",
"{MYSQL_PASSWORD}", "{MYSQL_DATABASE}");
?>
```

La función `mysqli_connect` recibe cuatro argumentos y devuelve el objeto con la conexión si la conexión tuvo éxito. Echemos un vistazo a cada argumento:

- `MYSQL_HOSTNAME` : Este debe ser sustituido con el nombre del host o la dirección IP del servidor MySQL. Si estás trabajando con el servidor de MySQL en tu sistema local, puedes usar `localhost` o `127.0.0.1` .
- `MYSQL_USERNAME` : El nombre de usuario de tu usuario de MySQL.
- `MYSQL_PASSWORD` : La contraseña de tu usuario de MySQL.
- `MYSQL_DATABASE` : La base de datos a la que quieras conectarte.

Después de logar una conexión exitosa, la variable `$connection_obj` contiene el objeto de la conexión. Con este objeto estás listo para ejecutar consultas en la base de datos que fue proporcionada en el argumento `MYSQL_DATABASE` .

Por otro lado, si la conexión no tiene éxito podemos ejecutar una depuración como se muestra en el siguiente fragmento de código:

```
<?php
$connection_obj = mysqli_connect("{MYSQL_HOSTNAME}", "{MYSQL_USERNAME}",
"{MYSQL_PASSWORD}", "{MYSQL_DATABASE}");

if (!$connection_obj) {
    echo "Error No: " . mysqli_connect_errno();
    echo "Error Description: " . mysqli_connect_error();
    exit;
}
?>
```

En la siguiente sección veremos cómo puedes seleccionar una base de datos específica usando el objeto de la conexión. Un **desencadenador** (o Trigger) es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos. **SQL** Server permite crear varios **desencadenadores** para una instrucción específica.

Cómo seleccionar una base de datos

En la sección anterior discutimos cómo establecer la conexión a una base de datos usando la función `mysqli_connec` t. En esta sección veremos cómo seleccionar una base de datos una vez que la conexión a MySQL haya sido exitosa.

Desde luego, como ya hemos visto siempre puedes enviar una base de datos en el cuarto argumento de la propia función `mysqli_connect` . Pero también hay otra manera en la que puedes hacer esto, en caso de que quieras cambiar una base de datos después de que te hayas conectado a MySQL. Puedes usar la función `mysqli_select_db` para seleccionar una base de datos con la que quieras trabajar.

Revisemos el ejemplo discutido en la sección anterior para ver cómo es que puede funcionar.

```
01    <?php
02    $connection_obj = mysqli_connect("{MYSQL_HOSTNAME}", "{MYSQL_USERNAME}", "{MYSQL_PASSWORD}");
03
04    if (!$connection_obj) {
05        echo "Error No: " . mysqli_connect_errno();
06        echo "Error Description: " . mysqli_connect_error();
07        exit;
08    }
09
10    mysqli_select_db($connection_obj, "{MYSQL_DATABASE}");
11    ?>
```

Como puedes ver hemos enviado solamente tres argumentos en la función `mysqli_connect` , y eso debe darnos una conexión exitosa a una base de datos. En vez de enviar una base de datos en el cuarto argumento, seleccionamos la base de datos usando la función `mysqli_select_db` .

La función `mysqli_select_db` recibe dos argumentos: el objeto de conexión y la base de datos a la que quieras conectarte.

Cómo crear (insertar) y actualizar registros

En las secciones anteriores discutimos cómo establecer una conexión con MySQL y seleccionar una base de datos para trabajar con ella. Ahora veremos cómo ejecutar diferentes tipos de consultas en la base de datos seleccionada. Primero veamos cómo insertar y actualizar registros.

Si quieres seguir los ejemplos discutidos en este artículo, necesitarás crear la siguiente tabla de MySQL en tu base de datos. Es la tabla que vamos a usar en todos los ejemplos de ahora en adelante.

```
1     CREATE TABLE `employee` (
2       `id` int(11) NOT NULL AUTO_INCREMENT,
```

```
3       `name` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
4       `email` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
5       `phone` varchar(50) COLLATE utf8mb4_unicode_ci NOT NULL,
6       PRIMARY KEY (`id`)
7     ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Ejecutar el comando anterior en tu base de datos de MySQL debe crear la tabla `employee`.

Cómo crear registros

Para mantener las cosas simples vamos a inicializar los valores que necesitan ser insertados en la tabla employee al inicio de la secuencia de comandos. Sin embargo, en la mayoría de los casos estos datos provienen de la entrada del usuario en las variables `$_POST` enviadas usando un formulario.

```
01  <?php
02  $connection_obj = mysqli_connect("{MYSQL_HOSTNAME}", "{MYSQL_USERNAME}", "{MYSQL_PASSWORD}", "{MYSQL_DATABASE}");
03
04  if (!$connection_obj) {
05      echo "Error No: " . mysqli_connect_errno();
06      echo "Error Description: " . mysqli_connect_error();
07      exit;
08 }
09
10  // initialize variables for the insert query
11  $name = 'John Smith';
12  $email = 'john.smith@yahoo.com';
13 $phone = '541-754-1234';
14
15  // prepare the insert query
16  $query = "INSERT INTO employee(`name`,`email`, `phone`)
17  VALUES ('". mysqli_real_escape_string($connection_obj, $name) ."','". mysqli_real_escape_string($connection_obj, $email) ."','". mysqli_real_escape_string($connection_obj, $phone) ."')" ;
18
19  // run the insert query
20  mysqli_query($connection_obj, $query);
21
22  // close the db connection
23  mysqli_close($connection_obj);
24 ?>
```

En la secuencia de comandos anterior primero hemos inicializado las variables que vamos a usar en la consulta insert. A continuación hemos preparado la consulta insert y la hemos asignado a la variable `$query`.

Es importante destacar que hemos usado la función `mysqli_real_escape_string` para escapar los valores de tipo cadena que vamos a usar en la consulta insert. Debes usar esta función cuando estés trabajando con valores de tipo cadena enviados a través de variables `$_POST`. Esto garantiza que tus usuarios no inyecten código malicioso en tus consultas a la base de datos.

Finalmente, ejecutamos la consulta insert usando la función `mysqli_query`. La función `mysqli_query` se usa para ejecutar una consulta en la base de datos activa. Para terminar, usamos la función `mysqli_close` para cerrar la conexión activa.

Cómo actualizar registros

En la sección anterior discutimos cómo puedes insertar nuevos registros en la tabla de MySQL. En esta sección veremos cómo puedes actualizar un registro que ya existe en la tabla.

Voy a asumir que ya existe un registro en la tabla employee, y que el id de ese registro tiene el valor 1. En el siguiente ejemplo vamos a actualizar un registro que tiene un valor de 1 en la columna id.

```
01      <?php
02      $connection_obj = mysqli_connect("{MYSQL_HOSTNAME}", "{MYSQL_USERNAME}", "{MYSQL_PASSWORD}", "{MYSQL_DATABASE}");
03
04      if (!$connection_obj) {
05          echo "Error No: " . mysqli_connect_errno();
06          echo "Error Description: " . mysqli_connect_error();
07          exit;
08      }
09
10      // initialize variables for the insert query
11      $id = 1;
12      $phone = '333-555-4444';
13
14      // prepare the insert query
15      $query = "UPDATE employee SET `phone` = '". mysqli_real_escape_string($connection_obj, $phone) ."' WHERE `id` = '". (int) $id ."'";
16
17      // run the insert query
18      mysqli_query($connection_obj, $query);
19
20      // close the db connection
21      mysqli_close($connection_obj);
22      ?>
```

Como puedes ver en el ejemplo anterior, queremos actualizar la columna `phone` de la tabla `employee` en donde `id` es igual a 1. Esto es muy similar a la consulta insert, pero estamos usando `UPDATE` en vez de `INSERT`.

Cómo obtener registros

Hasta ahora hemos discutido cómo insertar y actualizar registros en la tabla de MySQL. En esta sección veremos cómo obtener (recuperar) registros de la tabla de MySQL.

Echa un vistazo al siguiente ejemplo:

```
01      <?php
02      $connection_obj = mysqli_connect("{MYSQL_HOSTNAME}", "{MYSQL_USERNAME}", "{MYSQL_PASSWORD}", "{MYSQL_DATABASE}");
03
04      if (!$connection_obj) {
05          echo "Error No: " . mysqli_connect_errno();
06          echo "Error Description: " . mysqli_connect_error();
07          exit;
08      }
09
10      // prepare the select query
11      $query = "SELECT * FROM employee";
12
13      // execute the select query
14      $result = mysqli_query($connection_obj, $query) or die(mysqli_error($connection_obj));
15
16      // run the select query
17      while ($row = mysqli_fetch_array($result, MYSQLI_BOTH)) {
```

```
18          echo "ID:" . $row['id'] . "<br/>";
19          echo "Name:" . $row['name'] . "<br/>";
20          echo "Phone:" . $row['phone'] . "<br/>";
21          echo "Email:" . $row['email'] . "<br/>";
22          echo "<br/>";
23      }
24
25      // close the db connection
26      mysqli_close($connection_obj);
27      ?>
```

Lo primero que debes hacer para recuperar registros de una base de datos es ejecutar la consulta select usando la función `mysqli_query`. Después de haber ejecutado con éxito la consulta select, la función `mysqli_query` devuelve el objeto `mysqli result`, y eso es lo que hemos almacenado arriba en la variable `$result`.

A continuación recorremos el conjunto de resultados usando la función `mysqli_fetch_array` en un bucle while. La función `mysqli_fetch_array` obtiene una sola fila a la vez del conjunto de mysqli result.

La función `mysqli_fetch_array` recibe dos argumentos: el objeto de los resultados y el tipo de los resultados. El tipo indica qué tipo de arreglo será devuelto por la función `mysqli_fetch_array`. Puede ser numérico, asociativo o ambos. Es un argumento opcional, y puedes enviar uno de estos tres valores: `MYSQLI_ASSOC`, `MYSQLI_NUM` o `MYSQLI_BOTH`. `MYSQLI_ASSOC` indica que quieres acceder a las columnas por sus nombres, y `MYSQLI_NUM` indica que quieres acceder a las columnas por sus números.

Si envías el valor `MYSQLI_NUM` en el segundo argumento de la función , puedes `$row[0]`

mysqli_fetch_array acceder a las columnas como , $row[1] y así sucesivamente. En nuestro caso hemos enviado el valor MYSQLI_BOTH, de manera que podemos acceder a las columnas de las dos formas. Si quieres acceder solamente con valores asociativos, en vez de esto puedes usar la función mysqli_fetch_assoc.

En la siguiente y última sección veremos cómo borrar registros de una base de datos.

Cómo borrar registros

En esta sección veremos cómo borrar registros de una base de datos.
Revisa el siguiente ejemplo.

```
01      <?php
02      $connection_obj = mysqli_connect("{MYSQL_HOSTNAME}", "{MYSQL_USERNAME}", "{MYSQL_PASSWORD}", "{MYSQL_DATABASE}");
03
04      if (!$connection_obj) {
05          echo "Error No: " . mysqli_connect_errno();
06          echo "Error Description: " . mysqli_connect_error();
07          exit;
08      }
09
10      // initialize variables for the delete query
11      $id = 1;
12
13      // prepare the insert query
14      $query = "DELETE FROM employee WHERE `id` = '". (int) $id."'";
15
16      // run the delete query
17      mysqli_query($connection_obj, $query);
18
19      // close the db connection
20      mysqli_close($connection_obj);
21      ?>
```

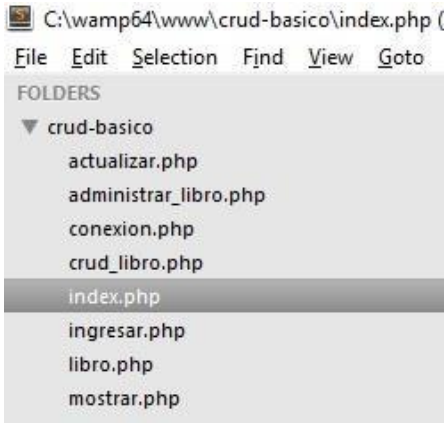
Como puedes ver, funciona básicamente de la misma manera en la que lo hacen las consultas insert y update. La función mysqli_query ejecuta la consulta DELETE, y esta eventualmente elimina el registro de la tabla employee.

ACTIVIDADES DE TRANSFERENCIA: Tiempo establecido para la actividad 30 horas

Una vez desarrollada la siguiente actividad de formación deberá transferir los conocimientos y habilidades adquiridos durante todo el proceso formativo, correspondiente al resultado de aprendizaje. CONSTRUIR LA MATRIZ CRUD EN EL LENGUAJE DE PROGRAMACIÓN SELECCIONADO PARA VERIFICAR LA FUNCIONALIDAD DEL SISTEMA DE ACUERDO CON EL DISEÑO ENTREGADO. Por medio de la siguiente actividad de formación.

QUÉ VAMOS HACER?: Vamos hacer un ejemplo muy sencillo, que nos va permitir ingresar, editar, leer y eliminar los datos de una tabla llamada libro y aunque es una sola tabla con tan solo 4 campos, te da la pauta suficiente para hacerlo con más tablas que tengan más campos.

ESTRUCTURA DEL PROYECTO?: La estructura queda como se ve en la imagen, con 8 archivos que los describo de forma general a continuación:



- ✓ **index.php:** Es un archivo que contiene html y lo que muestra son las opciones **Ingresar y Ver**, las mismas que direccionan a las páginas ingresar.php y mostrar.php respectivamente.
- ✓ **conexion.php:** Este archivo contiene el código para la conexión con MySQL.
- ✓ **libro.php:** Contiene la clase Libro, con sus respectivas propiedades y métodos getters y setters.
- ✓ **crud_libro.php:** Este archivo tiene los métodos crud para la tabla libros.
- ✓ **administrar_libro.php:** Funciona como un controlador que redirecciona dependiendo la petición hecha desde la vista.

- ✓ Finalmente se encuentran los archivos ingresar.php, mostrar.php y actualizar.php cuyas funciones son fácilmente reconocibles por los nombres que llevan.

CÓMO EMPEZAMOS?
Para empezar debes crear una carpeta llamada **crud-basico** dentro de la siguiente ruta C:\wamp64\www.

Una vez creada, la vas abrir con el editor que utilices, en mi caso con Sublime Text (File->Open Folder).

CREANDO LA BASE DE DATOS
Antes de continuar con la codificación, vamos a crear una base de datos en MySQL llamada **biblioteca** con una tabla llamada **libro**, para esto debes utilizar el siguiente script.

```sql
CREATE DATABASE IF NOT EXISTS biblioteca;

USE biblioteca;
CREATE TABLE `libros` (
  `id` int(11) NOT NULL,
  `nombre` varchar(50) COLLATE utf8mb4_spanish2_ci NOT NULL,
  `autor` varchar(50) COLLATE utf8mb4_spanish2_ci NOT NULL,
  `anio_edicion` varchar(50) COLLATE utf8mb4_spanish2_ci NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_spanish2_ci;

ALTER TABLE `libros`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `libros`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1;
```

ARCHIVO DE CONEXIÓN A LA BASE DE DATOS

De regreso a Sublime Text dentro de esta carpeta (**crud-basico**) creas el primero de 8 archivos, el primer archivo que creas es el de conexión a la base de datos, al mismo que lo vas a llamar conexion.php, y que contendrá el siguiente código.

```php
<?php
    class  Db{
        private static $conexion=NULL;
        private function __construct (){}

        public static function conectar(){
            $pdo_options[PDO::ATTR_ERRMODE]=PDO::ERRMODE_EXCEPTION;
            self::$conexion= new PDO('mysql:host=localhost;dbname=biblioteca','root','',$pdo_options
            return self::$conexion;
        }
    }
?>
```

Lo que creamos es una clase para conectarnos a la base de datos **biblioteca**, la conexión se la realiza a través de la interfaz PDO, la misma que proporciona un conjunto de funciones que nos permiten acceder a las tablas de una base de datos de una manera muy sencilla.
El constructor de la clase PDO recibe 3 parámetros: el origen de la base de datos, el usuario, la clave y un último parámetro que permite manejar alguna excepción que pueda ocurrir, ya sea debido a que el nombre de la base de datos, clave o usuario no fueran los correctos .
Se crea una función estática con la finalidad de que se pueda acceder sin la necesidad de crear un objeto.

CREANDO LA VISTA PRINCIPAL

Lo que vamos hacer ahora es crear la vista principal de nuestra pequeña aplicación, este archivo se va llamar **index.php** puesto que es lo primero que se va mostrar al usuario.

```html
<html>
    <head>
        <title>Administrar Libro</title>
    </head>
    <body>
        <header>
            Bienvenido Administrar Libro
        </header>
        <table border=1>
            <tr>
                <td><a href="ingresar.php">Ingresar</a></td>
            </tr>
            <tr>
                <td><a href="mostrar.php">Ver</a></td>
            </tr>
        </table>
        <footer>
            Administrar Libro 2017
        </footer>
    </body>
</html>
```

Lo más importante de este archivo son el direccionamiento a las páginas **ingresar.php** que permite crear un nuevo libro y **mostrar.php**, que muestra todos los libros guardados.

FORMULARIO PARA INGRESAR UN LIBRO

El siguiente archivo que vamos a crear, permite ingresar los datos del libro y crear la petición para guardar los datos en la base, creamos el archivo con el nombre de **ingresar.php.**

```
<html>
<head>
    <title> Ingresar Libro</title>
</head>
<header>
Ingresa los datos del Libro
</header>
<form action='administrar_libro.php' method='post'>
    <table>
        <tr>
            <td>Nombre libro:</td>
            <td> <input type='text' name='nombre'></td>
        </tr>
        <tr>
            <td>Autor:</td>
            <td><input type='text' name='autor' ></td>
        </tr>
        <tr>
            <td>Fecha Edición:</td>
            <td><input type='text' name='edicion' ></td>
        </tr>
        <input type='hidden' name='insertar' value='insertar'>
    </table>
    <input type='submit' value='Guardar'>
    <a href="index.php">Volver</a>
</form>

</html>
```

Para que una petición sea enviada y procesada en el servidor debe estar dentro de un formulario html, además de eso se puede utilizar dos métodos (**method**) para el envío: **POST** o **GET**. Sin entrar mucho a detalle la diferencia entre estos dos métodos es que a través de GET los parámetros (por ejemplo, un id, nombres, apellidos etc.) que se envíen al servidor viajan a través de la URL, mientras que con POST no lo hacen.

Así mismo se debe poner la página en la que será procesada la petición (action) que para este caso es el archivo **administrar_libro.php** que lo crearemos más adelante.

Algo interesante es el elemento <input type=’hidden’ name=’insertar’ value=’insertar’>, se lo pone como (hidden) oculto puesto que se lo utiliza como una bandera, para determinar si debe ingresar al código de ingresar en el archivo **administrar_libro.php**

LA CLASE LIBRO

El siguiente archivo **libro.php** contiene la clase **Libro** que permite mapear los atributos a la tabla libros.

```php
<?php
    class Libro{
        private $id;
        private $nombre;
        private $autor;
        private $anio_edicion;

        function __construct(){}

        public function getNombre(){
        return $this->nombre;
        }

        public function setNombre($nombre){
            $this->nombre = $nombre;
        }

        public function getAutor(){
            return $this->autor;
        }

        public function setAutor($autor){
            $this->autor = $autor;
        }

        public function getAnio_edicion(){
        return $this->anio_edicion;
        }

        public function setAnio_edicion($anio_edicion){
            $this->anio_edicion = $anio_edicion;
        }
        public function getId(){
            return $this->id;
        }

        public function setId($id){
            $this->id = $id;
        }
    }
?>
```

LA CLASE CON LAS FUNCIONES CRUD

En el archivo **crud_libro.php** vamos a codificar las funciones CRUD para la tabla libros.

```php
<?php
// incluye la clase Db
require_once('conexion.php');

    class CrudLibro{
        // constructor de la clase
        public function __construct(){}

        // método para insertar, recibe como parámetro un objeto de tipo libro
        public function insertar($libro){
            $db=Db::conectar();
            $insert=$db->prepare('INSERT INTO libros values(NULL,:nombre,:autor,:anio_edicion)');
            $insert->bindValue('nombre',$libro->getNombre());
            $insert->bindValue('autor',$libro->getAutor());
            $insert->bindValue('anio_edicion',$libro->getAnio_edicion());
            $insert->execute();

        }

        // método para mostrar todos los libros
        public function mostrar(){
            $db=Db::conectar();
            $listaLibros=[];
            $select=$db->query('SELECT * FROM libros');

            foreach($select->fetchAll() as $libro){
                $myLibro= new Libro();
                $myLibro->setId($libro['id']);
                $myLibro->setNombre($libro['nombre']);
                $myLibro->setAutor($libro['autor']);
                $myLibro->setAnio_edicion($libro['anio_edicion']);
                $listaLibros[]=$myLibro;
            }
            return $listaLibros;
        }

        // método para eliminar un libro, recibe como parámetro el id del libro
        public function eliminar($id){
            $db=Db::conectar();
            $eliminar=$db->prepare('DELETE FROM libros WHERE ID=:id');
            $eliminar->bindValue('id',$id);
            $eliminar->execute();
        }

        // método para buscar un libro, recibe como parámetro el id del libro
        public function obtenerLibro($id){
            $db=Db::conectar();
            $select=$db->prepare('SELECT * FROM libros WHERE ID=:id');
            $select->bindValue('id',$id);
            $select->execute();
            $libro=$select->fetch();
            $myLibro= new Libro();
            $myLibro->setId($libro['id']);
            $myLibro->setNombre($libro['nombre']);
            $myLibro->setAutor($libro['autor']);
            $myLibro->setAnio_edicion($libro['anio_edicion']);
            return $myLibro;
        }

        // método para actualizar un libro, recibe como parámetro el libro
        public function actualizar($libro){
            $db=Db::conectar();
            $actualizar=$db->prepare("UPDATE libros SET nombre=:nombre, autor=:autor,anio_edicion=:a
            $actualizar->bindValue('id',$libro->getId());
            $actualizar->bindValue('nombre',$libro->getNombre());
            $actualizar->bindValue('autor',$libro->getAutor());
            $actualizar->bindValue('anio',$libro->getAnio_edicion());
            $actualizar->execute();
        }
    }
?>
```

EL ARCHIVO QUE HARÁ DE CONTROLADOR

Ahora creamos el archivo **administrar_libro.php** este archivo hará las funciones de un controlador, ya que procesará de acuerdo a la petición que le sea enviado desde la vista.

```php
<?php
//incluye la clase Libro y CrudLibro
require_once('crud_libro.php');
require_once('libro.php');

$crud= new CrudLibro();
$libro= new Libro();

    // si el elemento insertar no viene nulo llama al crud e inserta un libro
    if (isset($_POST['insertar'])) {
        $libro->setNombre($_POST['nombre']);
        $libro->setAutor($_POST['autor']);
        $libro->setAnio_edicion($_POST['edicion']);
        //llama a la función insertar definida en el crud
        $crud->insertar($libro);
        header('Location: index.php');
    // si el elemento de la vista con nombre actualizar no viene nulo, llama al crud y actualiza el
    }elseif(isset($_POST['actualizar'])){
        $libro->setId($_POST['id']);
        $libro->setNombre($_POST['nombre']);
        $libro->setAutor($_POST['autor']);
        $libro->setAnio_edicion($_POST['edicion']);
        $crud->actualizar($libro);
        header('Location: index.php');
    // si la variable accion enviada por GET es == 'e' llama al crud y elimina un libro
    }elseif ($_GET['accion']=='e') {
        $crud->eliminar($_GET['id']);
        header('Location: index.php');
    // si la variable accion enviada por GET es == 'a', envía a la página actualizar.php
    }elseif($_GET['accion']=='a'){
        header('Location: actualizar.php');
    }
?>
```

LA VISTA MOSTRAR LIBROS

Llamado **mostrar.php**, lo que hace es mostrar todos los libros guardados.

```php
<?php
//incluye la clase Libro y CrudLibro
require_once('crud_libro.php');
require_once('libro.php');
$crud=new CrudLibro();
$libro= new Libro();
//obtiene todos los libros con el método mostrar de la clase crud
$listaLibros=$crud->mostrar();
?>

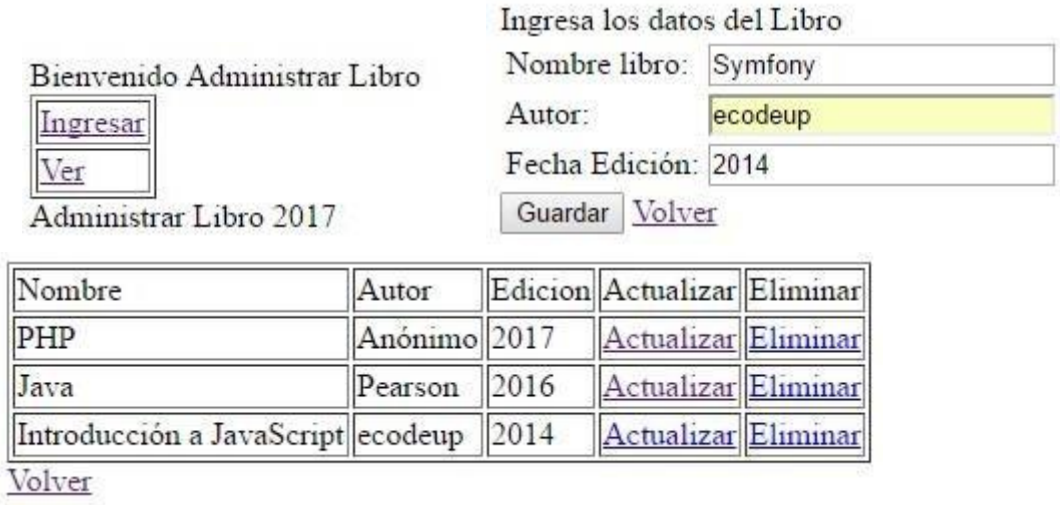
<html>
<head>
    <title>Mostrar Libros</title>
</head>
<body>
    <table border=1>
        <head>
            <td>Nombre</td>
            <td>Autor</td>
            <td>Edicion</td>
            <td>Actualizar</td>
            <td>Eliminar</td>
        </head>
        <body>
            <?php foreach ($listaLibros as $libro) {?>
            <tr>
                <td><?php echo $libro->getNombre() ?></td>
                <td><?php echo $libro->getAutor() ?></td>
                <td><?php echo $libro->getAnio_edicion()?> </td>
                <td><a href="actualizar.php?id=<?php echo $libro->getId()?>&accion=a">Actualizar</a>
                <td><a href="administrar_libro.php?id=<?php echo $libro->getId()?>&accion=e">Elimina
            </tr>
            <?php }?>
        </body>
    </table>
    <a href="index.php">Volver</a>
</body>
</html>
```

LA VISTA PARA ACTUALIZAR UN LIBRO

El archivo **actualizar.php** permite modificar los datos de un libro y posteriormente guardarlos.

```php
<?php
//incluye la clase Libro y CrudLibro
    require_once('crud_libro.php');
    require_once('libro.php');
    $crud= new CrudLibro();
    $libro=new Libro();
    //busca el libro utilizando el id, que es enviado por GET desde la vista mostrar.php
    $libro=$crud->obtenerLibro($_GET['id']);
?>
<html>
<head>
    <title>Actualizar Libro</title>
</head>
<body>
    <form action='administrar_libro.php' method='post'>
    <table>
        <tr>
            <input type='hidden' name='id' value='<?php echo $libro->getId()?>'>
            <td>Nombre libro:</td>
            <td> <input type='text' name='nombre' value='<?php echo $libro->getNombre()?>'></td>
        </tr>
        <tr>
            <td>Autor:</td>
            <td><input type='text' name='autor' value='<?php echo $libro->getAutor()?>' ></td>
        </tr>
        <tr>
            <td>Fecha Edición:</td>
            <td><input type='text' name='edicion' value='<?php echo $libro->getAnio_edicion() ?>'></
        </tr>
        <input type='hidden' name='actualizar' value'actualizar'>
    </table>
    <input type='submit' value='Guardar'>
    <a href="index.php">Volver</a>
</form>
</body>
</html>
```

RESULTADO FINAL



Como se puede ver es algo sencillo, pero que muestra a través de un **CRUD en PHP** lo fácil y rápido que es desarrollar aplicaciones con PHP

4. ACTIVIDADES DE EVALUACIÓN

Evidencias de Aprendizaje	Criterios de Evaluación	Técnicas e Instrumentos de Evaluación
Resultado de aprendizaje: 22050100704 - CONSTRUIR LA MATRIZ CRUD EN EL LENGUAJE DE PROGRAMACIÓN SELECCIONADO PARA VERIFICAR LA FUNCIONALIDAD DEL SISTEMA DE ACUERDO CON EL DISEÑO ENTREGADO		
Evidencias de Conocimiento: Describir informe técnico de diseño, características, estructura de la Matriz CRUD del Proyecto Formativo	Respuestas a preguntas abiertas o de selección múltiple con base en conocimientos de principio	Formulación de Preguntas
Evidencias de Desempeño: Diseñar la Matriz CRUD del Proyecto Formativo	Construir la Matriz CRUD del Proyecto Formativo	Lista de chequeo
Evidencias de Producto: Crear la Matriz CRUD del Proyecto Formativo	Documento con la Matriz CRUD del Proyecto Formativo	Cuestionario

REGLAMENTO DEL APRENDIZ DEL SERVICIO NACIONAL DE APRENDIZAJE - SENA

CAPÍTULO III. DEBERES Y PROHIBICIONES DEL APRENDIZ SENA. Artículo 8. Deberes del aprendiz SENA.

Cumplir con todas las actividades formativas propias de su proceso de aprendizaje según modalidad y del plan de mejoramiento, definidas durante su etapa lectiva y productiva; presentando puntualmente según el cronograma del Programa, las actividades y evidencias de aprendizaje establecidas para la ruta de aprendizaje.



5. GLOSARIO DE TERMINOS

Base de Datos (Archivo): Conjunto de datos almacenados de manera organizada y sistemática, que pertenecen a un mismo contexto y se relacionan entre sí, los cuales son registrados o recolectados para su posterior uso.

Datos: Es un valor o referente que recibe el computador por diferentes medios, los datos representan la información que el programador manipula en la construcción de una solución o en el desarrollo de un algoritmo

Archivo: Conjuntos de registros.

Registros: Conjuntos de campos.

Campo: Un campo, definido en una consulta, que muestra el resultado de una expresión en lugar de mostrar los datos almacenados. El valor se actualiza cada vez que cambia un valor de la expresión.

CRUD: Responde a las siglas en inglés (Create, Read, Update and Delete), Crear, Leer, Actualizar y Eliminar. Se refiere a las operaciones que se pueden realizar en un almacén de datos.

En la terminología de SQL se refieren a Insertar (INSERT), Seleccionar (SELECT), Actualizar (UPDATE) y Eliminar (DELETE) operaciones, respectivamente.

Desencadenador (o Trigger): Es una clase especial de procedimiento almacenado que se ejecuta automáticamente cuando se produce un evento en el servidor de bases de datos. SQL Server permite crear varios desencadenadores para una instrucción específica.

SQL (por sus siglas en inglés Structured Query Language); en español lenguaje de consulta estructurada): Es un lenguaje de dominio específico utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales.1 Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas.

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor): Es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

HTML: Es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto.

Conexión a Base de Datos: Es un archivo de configuración donde se especifica los detalles físicos de una base de datos como por ejemplo el tipo de base de datos y la versión, y los parámetros que permiten una conexión

6. REFERENTES BILBIOGRAFICOS

Cómo comenzar con operaciones CRUD en bases de datos MySQL con PHP

https://code.tutsplus.com/es/tutorials/how-to-work-with-mysql-in-php--cms-32222

CRUD EN PHP PARA INICIARSE EN LA PROGRAMACIÓN WEB

https://www.ecodeup.com/crud-en-php-para-principiantes/

7. CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	Equipo Ingenieros de Sistemas – Programa Articulación	Instructor	Comercio y Turismo	17/04/2020

8. CONTROL DE CAMBIOS

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					