Desarrollo de un CRUD usando HTML + Php (mysqli) + MySQL + POO

Nos adentraos en la cuarta entrega de las guías sobre base de datos, php, mysql, en esta ocasión le damos el turno a POO (Programación Orientada a Objetos), en esta sesión tendremos el mismo CRUD, solamente que introduciremos POO, de igual manera no podemos dejar atrás guías las cuales son [1] Como crear una base de datos en MySQL usando PhpMyAdmin, [2] Formularios HTML y procesamiento usando php, [3] - Desarrollo de un CRUD usando HTML + Php (mysqli) + MySQL), ya que esta es continuación de las antes mencionadas.

Asumiendo que ya esas dos guías fueron estudiadas, comprendidas y practicadas procedemos por definir ciertos conceptos que necesitamos tener en claro, el cual es el tema que corresponde a este guía.

Nota: la siguiente estructura es una manera de muchas de desarrollar con POO, dicha estructura es parte tomada de internet, parte tomada de la ayuda de php, parte tomada de experiencia personal, por lo cual NO, es restrictivo su uso tal cual, solo se trata de plasmar una forma sencilla de entender el paradigma de la POO.

Los siguientes conceptos son extraídos de Wikipedia tal cual, los mismos no tienen ninguna variación: https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

Definamos:

POO: La Programación Orientada a Objetos (POO, en español; OOP, según sus siglas en inglés) es un paradigma de programación que parte del concepto de "objetos" como base, los cuales contienen información en forma de campos (a veces también referidos como atributos o propiedades) y código en forma de métodos.

Los objetos son capaces de interactuar y modificar los valores contenidos en sus campos o atributos (estado) a través de sus métodos (comportamiento).

Muchos de los objetos prediseñados de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.

Clase: Una clase es una especie de "plantilla" en la que se definen los atributos y métodos predeterminados de un tipo de objeto. Esta plantilla se crea para poder crear objetos fácilmente. Al método de crear nuevos objetos mediante la lectura y recuperación de los atributos y métodos de una clase se le conoce como instanciación.

Herencia: Por ejemplo, herencia de la clase C a la clase D, es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D. Por lo tanto, puede usar los mismos métodos y variables registrados como "públicos" (public) en C. Los componentes registrados como "privados" (private) también se heredan, pero se mantienen escondidos al programador y solo pueden ser accedidos a través de otros métodos públicos. Para poder acceder a un atributo u operación de una clase en cualquiera de sus subclases pero mantenerla oculta para otras clases es necesario registrar los componentes como "protegidos" (protected), de esta manera serán visibles en C y en D pero no en otras clases.

Objeto: Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

Método: Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

Evento: Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento la reacción que puede desencadenar un objeto; es decir, la acción que genera.

Atributos: Características que tiene la clase.

Mensaje: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

Propiedad o atributo: Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

Estado interno: Es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.

Miembros de un objeto: Atributos, identidad, relaciones y métodos.

Identificación de un objeto: Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes. En comparación con un lenguaje imperativo, una "variable" no es más que un contenedor interno del atributo del objeto o de un estado interno, así como la "función" es un procedimiento interno del método del objeto.

Contenido de la Carpeta guiaCrudPOO

La estructura es la misma aplicada al caso de ejemplo **guiaCrud** (Imagen 1), solo que aparecerá una nueva carpeta llamada Clases y dentro tendremos una clase llamada class_estudiantes.php, y un pequeño cambio en la conexión ya utilizaremos un **extends** de la clase **mysqli** para heredar todos sus atributos y métodos de dicha clase.

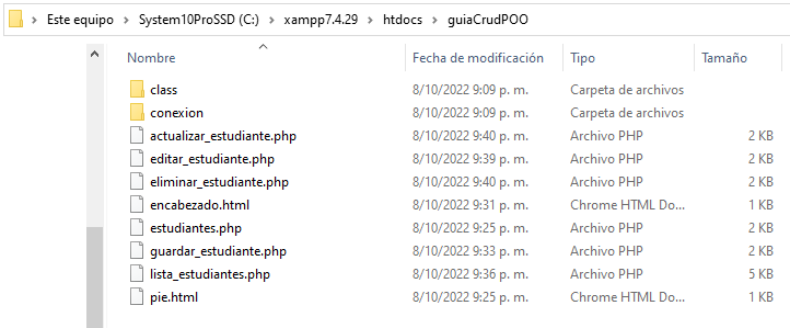


Imagen 1

Para este ejemplo de un CRUD sencillo usando html y php tendremos la siguiente estructura

Conexión: esta carpera contiene el archivo de **config.php**, el cual es una clase llamada **Cconfig** el cual contiene los atributos de conexión y los métodos Getter de dichos atributos, el archivo **basedatos.php** el cual es una clase el cual es un **extends** de la clase **mysqli** de php, aquí tenemos un solo método el cual es el **__construct()** dicho método se ejecuta cuando se hace una instanciación a dicha clase, es decir cuando la misma sea instanciada se realiza la conexión con la base de datos. Ver métodos contenidos dentro de la clase mysqli. (Nuevo)

Css: donde se almacenan los archivos css (No usada).
Imágenes: donde se almacenan la imagen (No usada).

Js: donde se almacenarán los archivos js (javascript).

class: contiene la clase o las clases donde estarán los métodos que hacen uso de la base de datos, leer, agregar, modificar, eliminar, buscar. (Nuevo)

Y los diferentes archivos HTML y php usados.

Nota : La siguiente explicación es una copia fiel y exacta del artículo [3] - Desarrollo de un CRUD usando HTML + Php (mysqli) + MySql), ya que el uso para el usuario final es el mismo el que cambia es el código que se empleo para el desarrollo.

lista_estudiantes.php (Imagen 2): es el primer archivo que se debe ejecutar y contiene la lista que permite visualizar los diferentes registros existentes en la tabla permitiendo manipularlo mostrando las opciones de Agregar, Modificar y Eliminar respectivamente. A nivel de programación se sigue la secuencia de consulta a la tabla con un SELECT, extraer los registros a la memoria del PC y luego son mostrados ordenados en una tabla y se colocan los botones de acción.

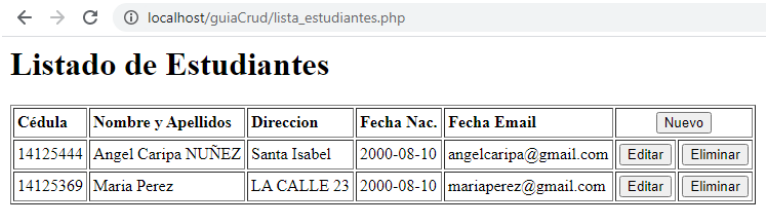


Imagen 2

Agregar Nuevo Registro

Presionemos el botón nuevo y nos llevará al formulario de estudiantes.html, el cual permite capturar los datos para poder registrar un nuevo estudiante que será anexado a la lista, el formulario vacío (Imagen 3), procedemos a editar los datos del mismo (imagen 4) y al presionar el botón de enviar (Imagen 4), mostrara el mensaje de Guardado con Exito y un link para regresar al listado (Imagen 5). A nivel de programación la secuencia seria estudiante.html (Formulario que captura los datos) este envía los datos a un archivo llamado guardar_estudiante.php y este conecta con la base de datos y guarda el registro.

Imagen 3

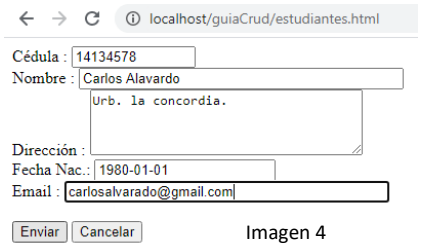


Imagen 4

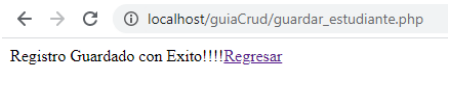


Imagen 5

Cuando se regresa a la lista podemos visualizar el nuevo registro de Carlos Alvarado (imagen 6).

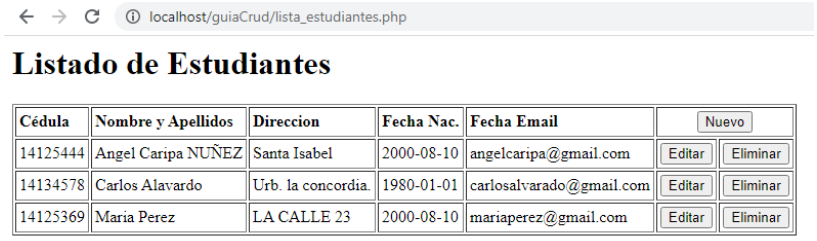


Imagen 6

Editar Registro

Si elegimos editar un Registro seleccionamos el que deseamos haciendo Click en la fila correspondiente al registro, aparecerá un mensaje indicando si se desea editar el registro (Imagen 7, Algo opcional, pero hacemos uso del lenguaje JavaScript embebido en HTML).

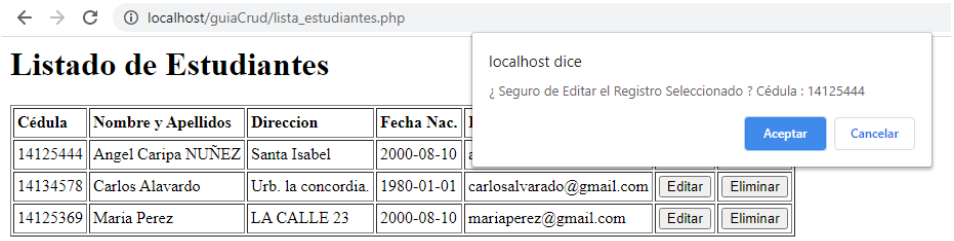


Imagen 7

Presionamos Aceptar y nos permitirá realizar la edición de este. A nivel de programación la secuencia es , al seleccionar el botón de editar se envía la cedula(campo clave) via GET(Observe la URL en la barra de dirección los datos son enviados a través de esta, Imagen 8), se realiza una consulta a la tabla contenida en la base de datos con SELECT se almacenan los datos en la memoria y se muestran en las cajas de textos con la opción de Editar, modificar o cambiar, véase imagen 9 donde se cambian el nombre se borra **NUÑEZ** y se le agregar **la Playa** en dirección, al presionar Enviar en la imagen 9 se llama **actualizar_estudiante.php** este hace la conexión con la base de datos y actualiza el registro y muestra el mensaje (Imagen 10), al presionar el link de Regresar se muestra el listado de estudiantes actualizado (Imagen 11).

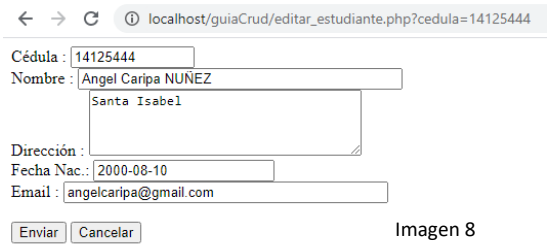


Imagen 8

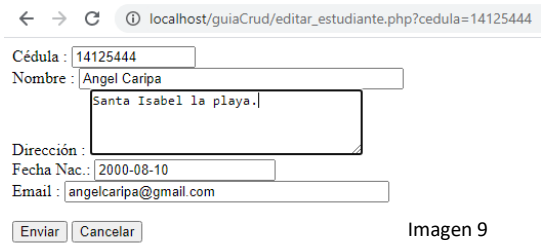


Imagen 9

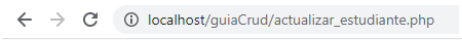


Imagen 10

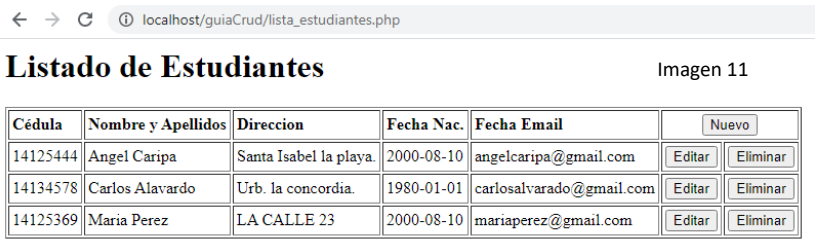


Imagen 11

Editar Registro

Si elegimos eliminar un Registro seleccionamos el que deseamos haciendo Click en la fila correspondiente al registro, aparecerá un mensaje indicando si se desea eliminar el registro (Imagen 12, este si no es opcional ya que voy a eliminar el mismo la idea sería consultar el usuario si realmente está de acuerdo en eliminar el registro, de igual manera hacemos uso de código JavaScript embebido en html (Imagen 12).

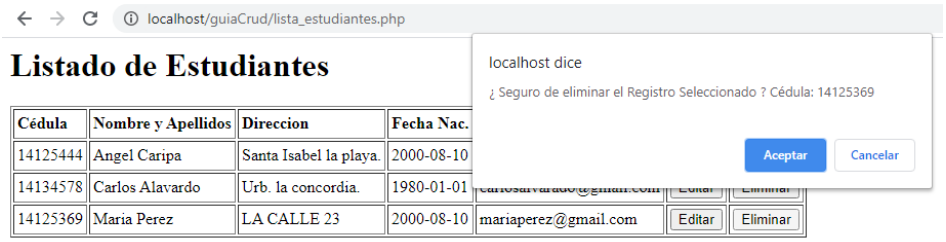


Imagen 12

Presionamos Aceptar y nos permitirá eliminar el registro, nos mostrará el menaje de Registro Eliminado (Imagen 13). A nivel de programación la secuencia es, al seleccionar el botón de eliminar se envía la cedula(campo clave) via GET(Observe la URL en la barra de dirección los datos son enviados a través de esta, Imagen 13), se realiza un DELETE en la tabla correspondiente y se elimina el mismo, al regresar de nuevo al listado vemos que el mismo esta actualizado con 1 registro menos (Imagen 14).

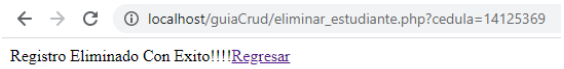


Imagen 13

Imagen 14

Nota: una aclaración importante con respecto a la eliminación de registros, ya que esto es algo delicado ya que una eliminación con DELETE permite eliminar físicamente el registro y el mismo no se puede recuperar, muchos autores o programadores utilizan la eliminación lógica la cual consiste en marcar o ocultar los registros para que los mismos no aparezcan en las consultas y se puedan recuperar en cualquier momento, otros utilizan una tabla o base de datos de histórico, otros utilizan una auditoria o bitácora de eventos ocurridos en los registros para asi tener un seguimiento a los que le sucedió a un registro. Cualquier técnica que utilice es valida siempre y cuando pueda permitir tener un control, integridad y seguridad en los datos.

Esta explicación es a nivel de usuario Final, es decir lo que hace el usuario en las vistas de la aplicación, y por supuestos todos estos procesos inciden en la base de datos actualizando la misma, ahora analicemos el código contenido en cada uno de los archivos de la aplicación.

Nota: la explicación de como el usuario maneja la aplicación se le conoce como manual de Usuario final.

Código de los archivos (Nuevo)

conexión/config.php : (Nuevo)en este archivo se encuentra la clase de **Cconfig** la cual contiene los datos de conexión con el servidor de base de datos, nótese que los atributos son de tipo privado y para poder acceder a ellos se debe crear los métodos Getter(Permiten leer el valor de los atributos) el cual son de tipo publico y son los que están presentes o se pueden acceder cuando se hace un instanciación a dicha clase.


```php
<?php
class Cconfig
{
    /** Atributos de la Clase */
    /** Datos de conexion al servidor de Base de Datos. */
    private $baseDato       = "academia";
    private $servidor       = "localhost";
    private $usuarioBd      = "root";
    private $claveBd        = "";
    private $puerto         = "3306";

    /**
     * Get the value of baseDato
     */
    public function getBaseDato()
    {
        return $this->baseDato;
    }

    /**
     * Get the value of servidor
     */
    public function getServidor()
    {
        return $this->servidor;
    }

    /**
     * Get the value of usuarioBd
     */
    public function getUsuarioBd()
    {
        return $this->usuarioBd;
    }

    /**
     * Get the value of claveBd
     */
    public function getClaveBd()
    {
        return $this->claveBd;
    }

    /**
     * Get the value of tipoBaseDato
     */
    public function getTipoBaseDato()
    {
        return $this->tipoBaseDato;
    }

    /**
     * Get the value of puerto
     */
    public function getPuerto()
    {
        return $this->puerto;
    }
}
```

conexion/basedatos.php : (Nuevo)en este archivo encontramos una clase **CBasedatos** la cual realiza un **extends** a la clase mysqli de php el cual están contenido los métodos para acceder a base de datos de MySQL y MariaDB. Al final de la guía pueden encontrar una Lista de los Métodos con sus parámetros y la descripción de cada uno de ellos (en Ingles).

C:\xampp7.4.29\htdocs\guiaCrudPOO\conexion\basedatos.php - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?

config.php basedatos.php

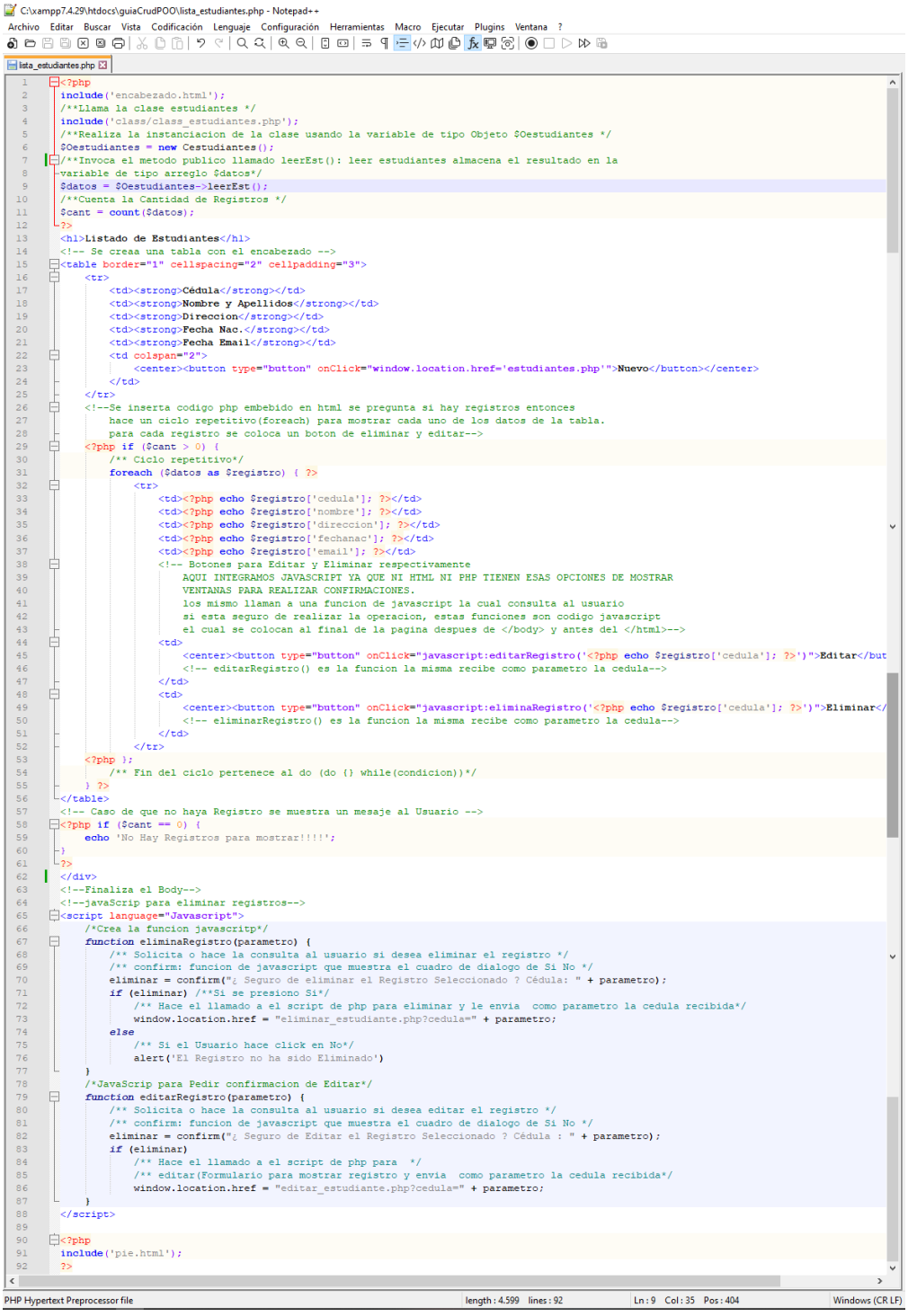
```php
<?php
/**Se requiere la utilizacion de la clase Cconfig contenida en el archivo config.php*/
require_once('config.php');

class CBaseDato extends mysqli
{
    private $dbh;
    public function __construct()
    {
        try {
            /**Instanciacion a la Clase config donde se encuentran todos los parametros de la misma*/
            $Oconfig   = new CConfig();
            $this->dbh = parent::__construct($Oconfig->getServidor(),
            $Oconfig->getUsuarioBd(), $Oconfig->getClaveBd(), $Oconfig->getBaseDato());
            if (mysqli_connect_error()) {
                die('Error de Conexión (' . mysqli_connect_errno() . ') '
                        . mysqli_connect_error());
            }
        } catch (Exception $e) {
            echo "Error : ".$e->getCode().":".$e->getMessage();
        }
    }

    /* cerrarConexion(): permite cerrar la conexion con el servidor de Base de Datos */
    public function cerrarConexion()
    {
        $this->conn->close();
        $this->dbh=null;
    }
}
```

PHP Hypertext Preprocessor file | length : 1.051 lines : 30 | Ln : 30 Col : 2 Pos :

Lista_estudiantes.php(Nuevo)

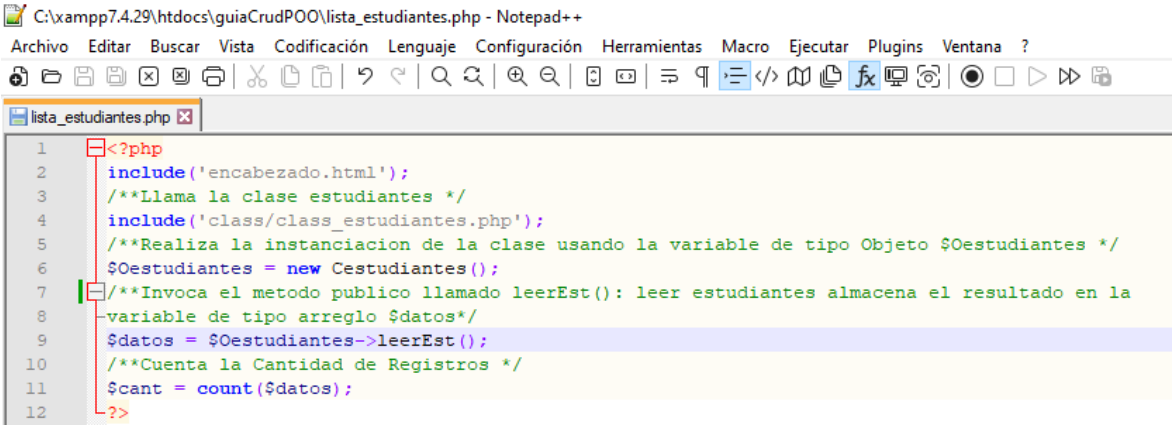


```php
<?php
include('encabezado.html');
/**Llama la clase estudiantes */
include('class/class_estudiantes.php');
/**Realiza la instanciacion de la clase usando la variable de tipo Objeto $Oestudiantes */
$Oestudiantes = new Cestudiantes();
/**Invoca el metodo publico llamado leerEst(): leer estudiantes almacena el resultado en la
variable de tipo arreglo $datos*/
$datos = $Oestudiantes->leerEst();
/**Cuenta la Cantidad de Registros */
$cant = count($datos);
?>
<h1>Listado de Estudiantes</h1>
<!-- Se creaa una tabla con el encabezado -->
<table border="1" cellspacing="2" cellpadding="3">
    <tr>
        <td><strong>Cédula</strong></td>
        <td><strong>Nombre y Apellidos</strong></td>
        <td><strong>Direccion</strong></td>
        <td><strong>Fecha Nac.</strong></td>
        <td><strong>Fecha Email</strong></td>
        <td colspan="2">
            <center><button type="button" onClick="window.location.href='estudiantes.php'">Nuevo</button></center>
        </td>
    </tr>
    <!--Se inserta codigo php embebido en html se pregunta si hay registros entonces
        hace un ciclo repetitivo(foreach) para mostrar cada uno de los datos de la tabla.
        para cada registro se coloca un boton de eliminar y editar-->
    <?php if ($cant > 0) {
        /** Ciclo repetitivo*/
        foreach ($datos as $registro) { ?>
            <tr>
                <td><?php echo $registro['cedula']; ?></td>
                <td><?php echo $registro['nombre']; ?></td>
                <td><?php echo $registro['direccion']; ?></td>
                <td><?php echo $registro['fechanac']; ?></td>
                <td><?php echo $registro['email']; ?></td>
                <!-- Botones para Editar y Eliminar respectivamente
                    AQUI INTEGRAMOS JAVASCRIPT YA QUE NI HTML NI PHP TIENEN ESAS OPCIONES DE MOSTRAR
                    VENTANAS PARA REALIZAR CONFIRMACIONES.
                    los mismo llaman a una funcion de javascript la cual consulta al usuario
                    si esta seguro de realizar la operacion, estas funciones son codigo javascript
                    el cual se colocan al final de la pagina despues de </body> y antes del </html>-->
                <td>
                    <center><button type="button" onClick="javascript:editarRegistro('<?php echo $registro['cedula']; ?>')">Editar</but
                    <!-- editarRegistro() es la funcion la misma recibe como parametro la cedula-->
                </td>
                <td>
                    <center><button type="button" onClick="javascript:eliminaRegistro('<?php echo $registro['cedula']; ?>')">Eliminar</
                    <!-- eliminarRegistro() es la funcion la misma recibe como parametro la cedula-->
                </td>
            </tr>
    <?php };
        /** Fin del ciclo pertenece al do (do {} while(condicion))*/
    } ?>
</table>
<!-- Caso de que no haya Registro se muestra un mesaje al Usuario -->
<?php if ($cant == 0) {
    echo 'No Hay Registros para mostrar!!!!';
}
?>
</div>
<!--Finaliza el Body-->
<!--javaScrip para eliminar registros-->
<script language="Javascript">
    /*Crea la funcion javascritp*/
    function eliminaRegistro(parametro) {
        /** Solicita o hace la consulta al usuario si desea eliminar el registro */
        /** confirm: funcion de javascript que muestra el cuadro de dialogo de Si No */
        eliminar = confirm("¿ Seguro de eliminar el Registro Seleccionado ? Cédula: " + parametro);
        if (eliminar) /**Si se presiono Si*/
            /** Hace el llamado a el script de php para eliminar y le envia  como parametro la cedula recibida*/
            window.location.href = "eliminar_estudiante.php?cedula=" + parametro;
        else
            /** Si el Usuario hace click en No*/
            alert('El Registro no ha sido Eliminado')
    }
    /*JavaScrip para Pedir confirmacion de Editar*/
    function editarRegistro(parametro) {
        /** Solicita o hace la consulta al usuario si desea editar el registro */
        /** confirm: funcion de javascript que muestra el cuadro de dialogo de Si No */
        eliminar = confirm("¿ Seguro de Editar el Registro Seleccionado ? Cédula : " + parametro);
        if (eliminar)
            /** Hace el llamado a el script de php para  */
            /** editar(Formulario para mostrar registro y envia  como parametro la cedula recibida*/
            window.location.href = "editar_estudiante.php?cedula=" + parametro;
    }
</script>

<?php
include('pie.html');
?>
```

Para el código de **listado_estudiantes.php** el cual permite mostrar una tabla con los datos de los estudiantes, hay que hacer énfasis en dos partes, las primeras líneas de código y la forma como se hace el ciclo repetitivo. Este archivo usa lo que se llama código php embebido en HTML. Aquí se conservan parte del código de guiaCrud.

Primeras líneas del archivo.



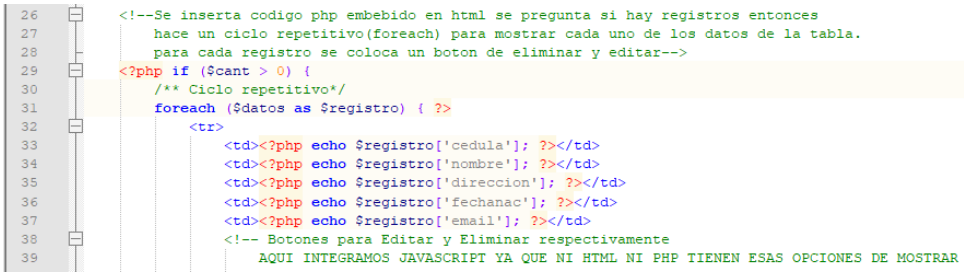
En la línea numero 2: comienza por un include, y hace referencia a un archivo **encabezado.html**, el cual el objetivo es establecer un encabezado para toda la aplicación es decir este archivo debe ser el mismo para todas las pantallas entonces para ahorra trabajo se crear un archivo separado y al modificarse este se modifica en todos los archivos donde se utiliza, así como se tiene el encabezado al principio del archivo se usa uno al final el cual es el **pie.html**.

En la línea numero 4: hacemos el include a la clase estudiantes(class_estudiantes.php) es aquí donde están todos los métodos que usaremos en el CRUD (leerEst(), agregarEst(), buscarEst(), modificarEst(), eliminarEst()).

En la línea numero 6: hacemos la instanciación de la clase es decir definimos una variable de tipo Objeto, la cual me va a permitir acceder a todos los métodos públicos contenidos en la clase **Cestudiantes**.

En la línea numero 9: le asignamos el resultado del método leerEst() a la variable $datos(la cual es un arreglo asociativo que contine todos los registros devueltos por la consulta SELECT). Al final de class_estudiante.php puede ver un ejemplo de los arreglos asociativos y su forma de usar.

En la línea Numero 11: tenemos una variable **$cant**, la cual almacena el tamaño del arreglo $datos, si el mismo tiene 0 elementos es porque no se devolvió ningún registro y tiene un valor mayor a 0 es porque encontró registros.



En la línea numero 29: realizamos la verificación de la variable $cant si la misma es mayor a 0, es porque el método leerEst(), devolvió una cantidad de registros y los mismos se pueden listar o mostrar.

En la línea numero 31: realizamos el ciclo repetitivo, en este caso utilizamos un foreach(), el cual permite recorrer arreglos asociativos.

En la línea numero 33,34,35,36,37: se muestran los valores de los campos, para ello se utiliza la variable $registro[‘campo’], dicha variable esta contenida en el foreach() y es la que contiene los campos de cada registro.

```
53          <?php };
54              /** Fin del ciclo pertenece al do (do {} while(condicion))*/
55          } ?>
56      </table>
57      <!-- Caso de que no haya Registro se muestra un mesaje al Usuario -->
58      <?php if ($cant == 0) {
59          echo 'No Hay Registros para mostrar!!!!';
60      }
61      ?>
62      </div>
63      <!--Finaliza el Body-->
```

En la línea numero 53: en esta línea tenemos el cierre del foreach().

En la línea numero 58: en esta línea realizamos la verificación nuevamente de la variable $cant si la misma es isual a 0, se emite un mensaje indicando la no existencia de registros, también se puede usar **else** para indicar la no existencia de registros.

estudiantes.php (Formulario d Captura de Datos): (Nuevo)ahora modificado por **estudiantes.php**, para poder incluir encabezado.html y pie.html, véase líneas 1 y 23 respectivamente.

C:\xampp7.4.29\htdocs\guiaCrudPOO\estudiantes.php - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?

lista_estudiantes.php | estudiantes.php

```
1   <?php include('encabezado.html'); ?>
2   <!--Inico del Formulario, importante Metodo(mthod, POST o GET) y Acccion(action): donde se enviara el fo
3   <form id="form1" name="form1" method="POST" action="guardar_estudiante.php">
4       <label>Cédula : </label>
5       <input type="text" name="cedula" id="cedula" size="12" maxlength="12" required />
6       <br>
7       <label>Nombre :</label>
8       <input type="text" name="nombre" id="nombre" size="40" maxlength="40" required />
9       <br>
10      <label>Dirección :</label>
11      <textarea name="direccion" id="direccion" cols="35" rows="4" required></textarea>
12      <br>
13      <label>Fecha Nac.:</label>
14      <input type="date" name="fechanac" id="fechanac" size="20" maxlength="10" required />
15      <br>
16      <label>Email :</label>
17      <input type="email" name="email" id="email" size="40" maxlength="40" required />
18      <br><br>
19      <input name="button" type="submit" id="buttonenviar" value="Enviar" />
20      <input name="button" type="reset" id="buttoncancelar" value="Cancelar" />
21  </form>
22  <!--Finaliza el Formulario-->
23  <?php include('pie.html'); ?>
```

guardar_estidiante.php (Guarda los datos editados en el formulario anterior). (Nuevo)

C:\xampp7.4.29\htdocs\guiaCrudPOO\guardar_estudiante.php - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?

lista_estudiantes.php | estudiantes.php | guardar_estudiante.php

```php
<?php
/**Proceso de formulario*/
/* Recojer los datos enviados del formulario
      Sintaxis : $variable=POST['nombre_objeto'];
Donde $variable : es cualquier nombre de variable este nombre debe guardar cierta relacion con el valor a almacenar.
         $_POST : Arreglo que contiene los valores enviados, depende del metodo seleccionado en el Formulario, puede
                  ser POST este envia los datos oculto al usuario.
          $_GET : Arreglo que contiene los valores enviados, envia los datos en la url del navgador se ve la url de
                  la siguiente manera :
                  http://localhost/guiaCrud/proceso_form_base.php?cedula=14123456&nombre=Pedro+Perez&direccion=La+concordia
                  &fechanac=01%2F01%2F2000&email=pedroperez35%40gmail.com&button=Enviar
'nombre_objeto' : es el nombre especificado en la propiedad name del objeto cuando se define el objeto.
    Conslusión : si se envia con el Metodo POST se debe recibir con el metodo POST[], si se envia con GET[] se debe recibir
usnado GET[], existe una variante de recibir con REQUEST['nombre_objeto'] es cual no importa si el evio fue hecho con
GET o POST.
*/

/**Recoger los datos enviados del Formulario*/
$cedula    = $_REQUEST['cedula'];
$nombre    = $_REQUEST['nombre'];
$direccion = $_REQUEST['direccion'];
$fechanac  = $_REQUEST['fechanac'];
$email     = $_REQUEST['email'];
/**Incluir la clase estudiantes donde estan los metodos a usar en este caso agergarEst()*/
include('class/class_estudiantes.php');
/**Se realiza la instanciacion a la clase Cestudiantes para poder tener acceso a los metodos
 * publicos alli existentes
 * */
$Oestudiantes = new Cestudiantes();
/**Este metodo agergarEst(), rquiere de los datos enviados por el formulario es decir
 * se necesita asignarle a los atributos de la clase los datos del registro a guardar, los
 * cuales serian cedula, nombre, direccion, fechanac,email pero como estos son privados
 * se deben utilizar los metos setter creados para cada uno los cuales se encargan de
 * asignarle a dichos atributos los valores capturados del formulario
 * */
$Oestudiantes->setCedula($cedula);
$Oestudiantes->setNombre($nombre);
$Oestudiantes->setDireccion($direccion);
$Oestudiantes->setFechanac($fechanac);
$Oestudiantes->setEmail($email);
/**Una vez asignados los valores a dichos atributos llamamos al metodo para que,
 * agregue el registro
 * */
$Oestudiantes->agregarEst();
echo "<a href='lista_estudiantes.php'>Regresar</a>"
/** Fin del proceso */
?>
```

PHP Hypertext Preprocessor file | length : 2.523 lines : 47 | Ln : 4 Col : 35 Pos : 116 | Windows (CR LF)

Para el caso de guardarestudiantes.php, anteriormente teniamos la conexión a la base de datos, la instruccion SQL que insertaba el registro (Manera convencional de insertar un registros), aquí cambiamos un poco la cosa y comenzamos a pornerle un poco de profesionalismo al codigo, o lo que llamamos las mejores practicas de desarrollo.

Y vemos en el archivo el contneido de las siguientes lineas.

En la linea numero 19,20,21,22,23, la captura de los datos que provienen del formulario, aquí un pequeño cambio usamos $_REQUEST[], recordemos que de esta forma no nos intereza si viene de $_POST[], $_GET[].

En la linea número 25: tenemos la inclusion de la clase, antes de poder instanciar debemos de incluirla.

En la linea número 29: tenemos la instanciacion a la clase cestudiantes, eto para poder usar los metodos publicos contenidos en ella.

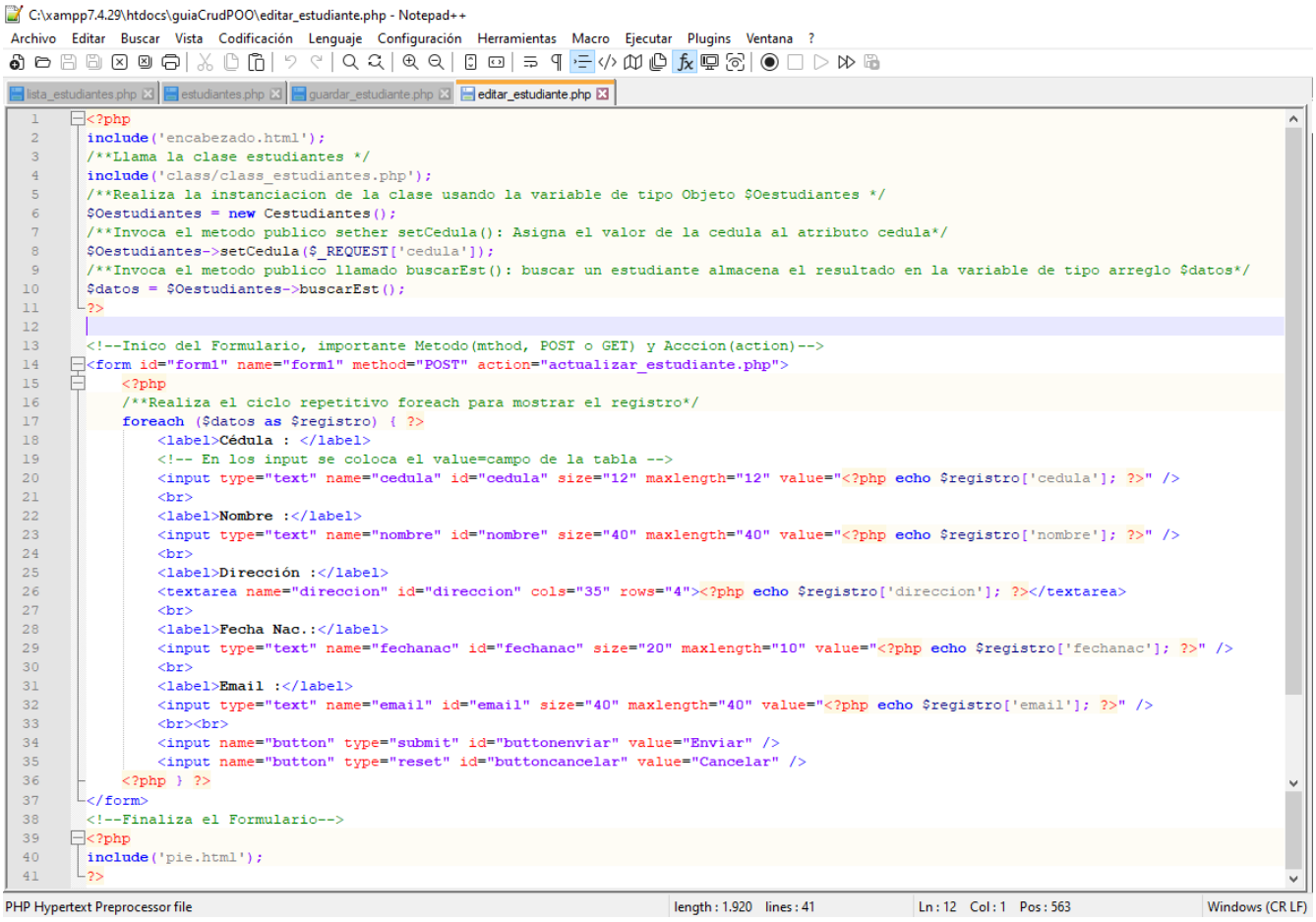
En la linea número 36 hasta 40, tenemos el seteo de las variables o atributos de la clase estudiantes.

En la liena número 44, tenemos la ejecucion del metodo agregarEst(), el cual toma el valor asignado a los atributos y los mismos son eviados a la base de datos usando una instrucción SQL(INSERT). Vease la class_estudiante.php

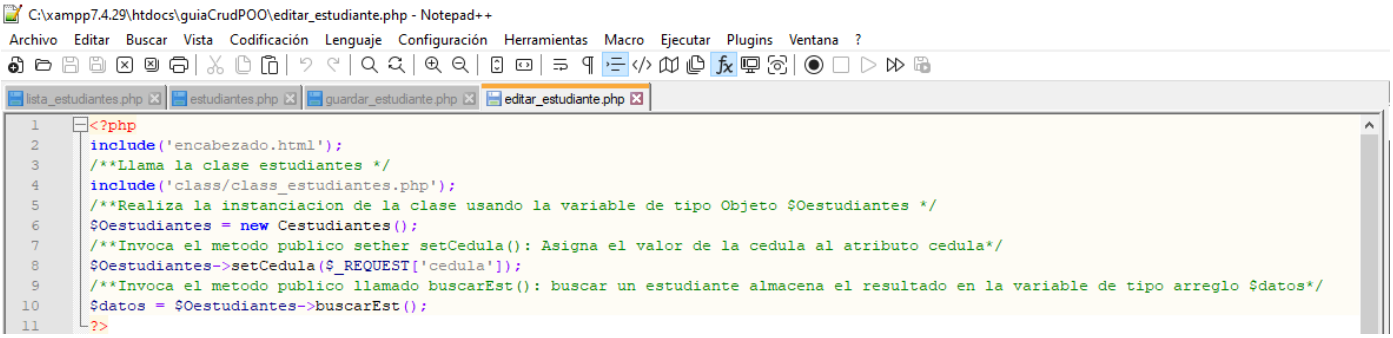
Para complementera este codigo vease la class_estudiantes.php especificamente el metodo **agregarEst().**

Nota : para esta fecha del año 2022, hablando de las mejores practicas de desarrollo no se se debe usar instrucciones SQL para el acceso a las base de datos eso ya no se permite por convenciones, seguridad entre otras, se deben utilizar ORM(del inglés Object Relational Mapping) o Mapeo Objeto-Relacional, el cual no es más que una pieza de software que nos permite interactuar con nuestra base de datos sin la necesidad de conocer SQL (El lenguaje de consultas). Todo esto utilizando el paradigma de programación orientada a objetos, esto trae seguridad para los sistemas desarrollados, sin embargo personalmente sigo utiulizando instruciones SQL y conexiones estandares a las base de datos, solo es aplicaciones complejas hago uso de estos ORM o framework de desarrollo como Laravel, Synfony entre otros. Algunos ORM Eloquent,Doctrine,Propel,ReadBean.

editar_estudiante.php (Muestra el registro en un formulario para ser editado). (Nuevo)



Recordemos el formulario de editar estudiantes comenzaba con una SQL del tipo SELECT, el cual permitía buscar el registro a modificar en la tabla para luego mostrarlo en el formulario y se da la opción de cambiar los valores por otro para luego actualizar el mismo. Este también sufrió un cambio el cual se nombra a continuación.



En la línea numero 2: como en todos los archivos se muestra el include del encabezado.html.

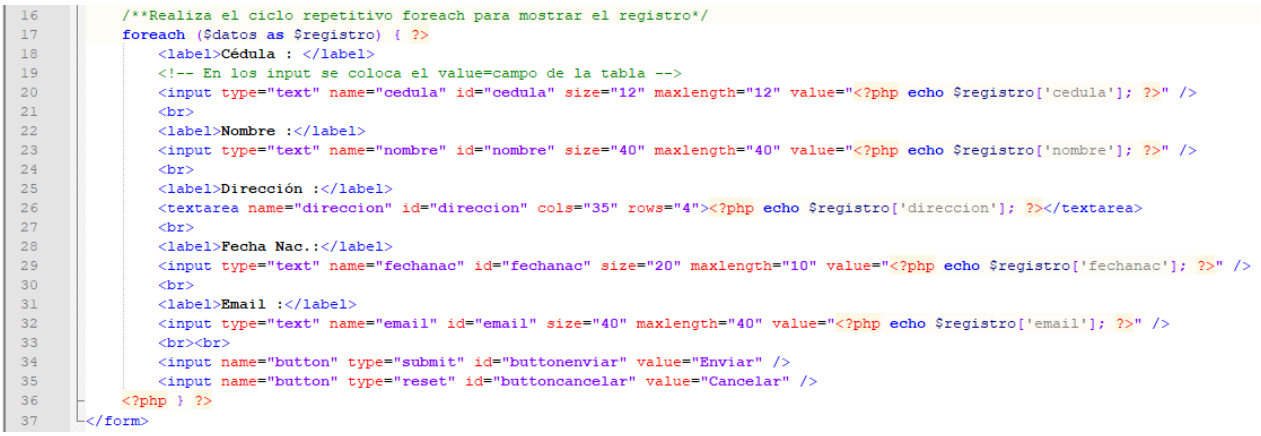
En la línea numero 4: se hace el include a la clase **class_estudiantes.php**, recordemos que para poder instanciar dicha clase la misma se debe incluir como parte del código antes de su uso.

En la línea numero 6: realizamos la instanciación a la clase Cestudiantes, para poder utilizar sus métodos de tipo público.

En la línea numero 8: el método **buscarEst()** requiere el valor del campo clave a buscar, es por ello que solo se setea el atributo cedula.

En la línea numero 10: se realizar el llamado o la ejecución al método **buscarEst()**, el cual se encargara de devolver un arreglo asociativo y almacenarlo en la variable $datos.

Aunque siempre la consulta o el método **buscarEst(),** devolverá un solo registro poder hacer un **foreach()** para extraer el valor del único registro que esta en el arreglo y posteriormente mostrarlo en los value de los input, recordemos nuevamente que aquí usamos PHP embebido en HTML.



actualizar_estudiante.php (Actualiza el registro editado en el formulario anterior). (Nuevo)

C:\xampp7.4.29\htdocs\guiaCrudPOO\actualizar_estudiante.php - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?

lista_estudiantes.php | estudiantes.php | guardar_estudiante.php | editar_estudiante.php | actualizar_estudiante.php

```php
<?php
/** Proceso de formulario */

/*Recojer los datos enviados del formulario
      Sintaxis : $variable=POST['nombre_objeto'];
Donde $variable : es cualquier nombre de variable este nombre debe guardar cierta relacion con el valor a almacenar.
        $_POST : Arreglo que contiene los valores enviados, depende del metodo seleccionado en el Formulario, puede
                 ser POST este envia los datos oculto al usuario.
         $_GET : Arreglo que contiene los valores enviados, envia los datos en la url del navgador se ve la url de
                 la siguiente manera :
                 http://localhost/guia/proceso_form_base.php?cedula=14123456&nombre=Pedro+Perez&direccion=La+concordia
                 &fechanac=01%2F01%2F2000&email=pedroperez35%40gmail.com&button=Enviar
'nombre_objeto' : es el nombre especificado en la propiedad name del objeto cuando se define el objeto.
     Conslusión : si se envia con el Metodo POST se debe recibir con el metodo POST[], si se envia con GET[] se debe recibir
usnado GET[], , existe una variante de recibir con REQUEST['nombre_objeto'] es cual no importa si el evio fue hecho con
GET o POST */

/** Recoger los datos enviados del Formulario */
$cedula    = $_POST['cedula'];
$nombre    = $_POST['nombre'];
$direccion = $_POST['direccion'];
$fechanac  = $_POST['fechanac'];
$email     = $_POST['email'];
/**Incluir la clase estudiantes donde estan los metodos a usar en este caso modificarEst()*/
include('class/class_estudiantes.php');
/**Se realiza la instanciacion a la clase Cestudiantes para poder tener acceso a los metodos
 * publicos alli existentes
 * */
$Oestudiantes = new Cestudiantes();
/**Este metodo modificarEst(), rquiere de los datos enviados por el formulario es decir
 * se necesita asignarle a los atributos de la clase los datos del registro a guardar, los
 * cuales serian cedula, nombre, direccion, fechanac,email pero como estos son privados
 * se deben utilizar los metodos setter creados para cada uno los cuales se encargan de
 * asignarle a dichos atributos los valores capturados del formulario
 * */
$Oestudiantes->setCedula($cedula);
$Oestudiantes->setNombre($nombre);
$Oestudiantes->setDireccion($direccion);
$Oestudiantes->setFechanac($fechanac);
$Oestudiantes->setEmail($email);
/**Una vez asignados los valores a dichos atributos llamamos al metodo para que,
 * modifique el registro
 * */
$Oestudiantes->modificarEst();
echo "<a href='lista_estudiantes.php'>Regresar</a>"
/** Fin del proceso */
?>
```

PHP Hypertext Preprocessor file | length : 2.519 lines : 47 | Ln : 1 Col : 1 Pos : 1

Para el caso de actualizarestudiantes.php, anteriormente teniamos la conexión a la base de datos, la instruccion SQL que actualizaba el registro (Manera convencional de actualizar registros), aquí cambiamos igualmente que en guardarestudiantes.php y comenzamos a pornerle un poco de profesionalismo al codigo, o lo que llamamos las mejores practicas de desarrollo.

Y vemos en el archivo el contenido de las siguientes lineas.

En la línea numero 19,20,21,22,23, la captura de los datos que provienen del formulario, segimos usando $_POST[], igual manera podemos usar $_REQUEST[].

En la línea numero 25: tenemos la inclusion de la clase, antes de poder instanciar debemos de incluirla.

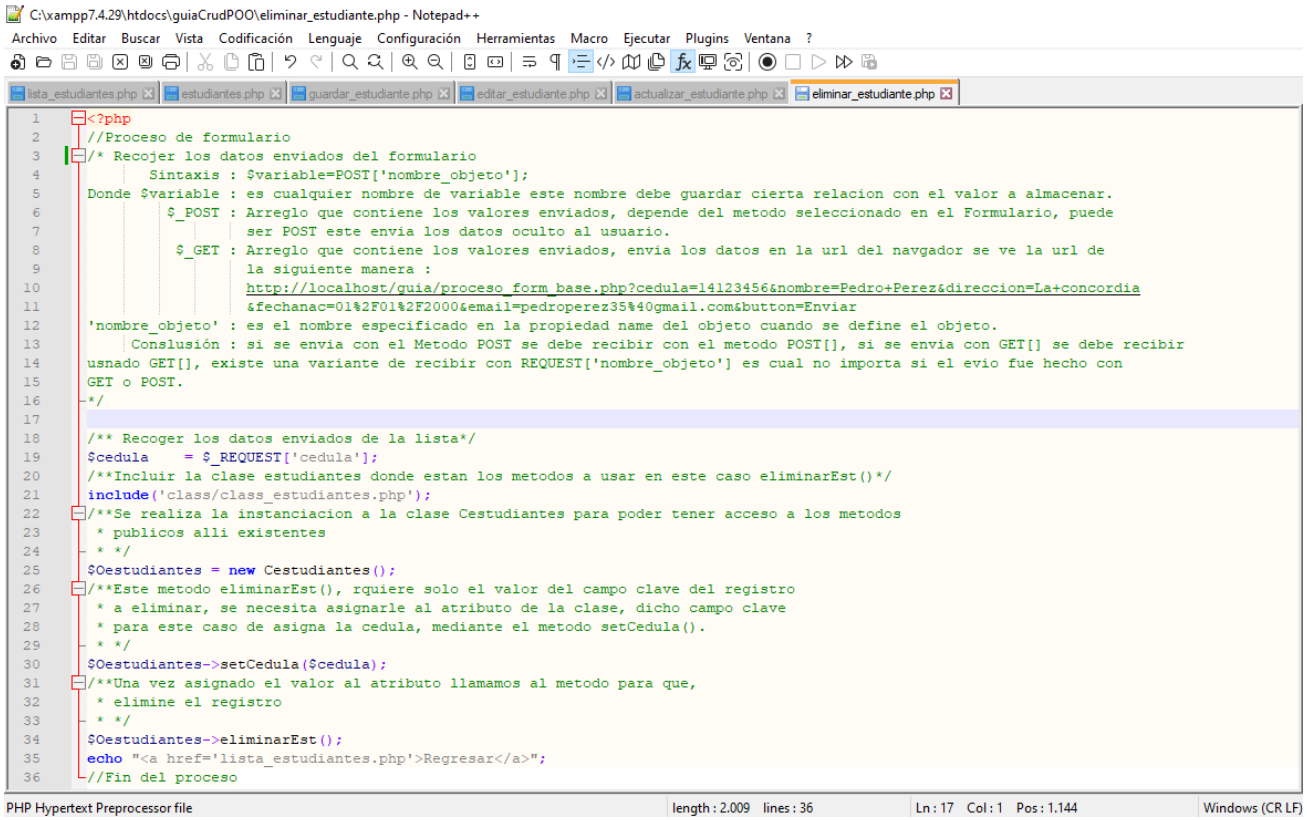
En la línea numero 29: tenemos la instanciacion a la clase cestudiantes, esto para poder usar los metodos publicos contenidos en ella.

En la línea numero 36 hasta 40, tenemos el seteo de las variables o atributos de la clase estudiantes.

En la líena numero 44, tenemos la ejecucion del metodo agregarEst(), el cual toma el valor asignado a los atributos y los mismos son eviados a la base de datos usando una instrucción SQL.

Para complementera este codigo vease la class_estudiantes.php especificamente el metodo **modificarEst().**

eliminar_registro.php (Elimina el registro seleccionado de la lista). (Nuevo)



En la línea numero 19: se hace el capture de la cedula, el cual es el valor que se requiere para eliminar.

En la línea numero 21: se hace el include a la clase **class_estudiantes.php**, recordemos que para poder instanciar dicha clase la misma se debe incluir como parte del código antes de su uso.

En la línea numero 25: realizamos la instanciación a la clase Cestudiantes, para poder utilizar sus métodos de tipo público.

En la línea numero 34: el método **eliminarEst()** requiere el valor del campo clave a buscar, es por ello que solo se setea el atributo cedula.

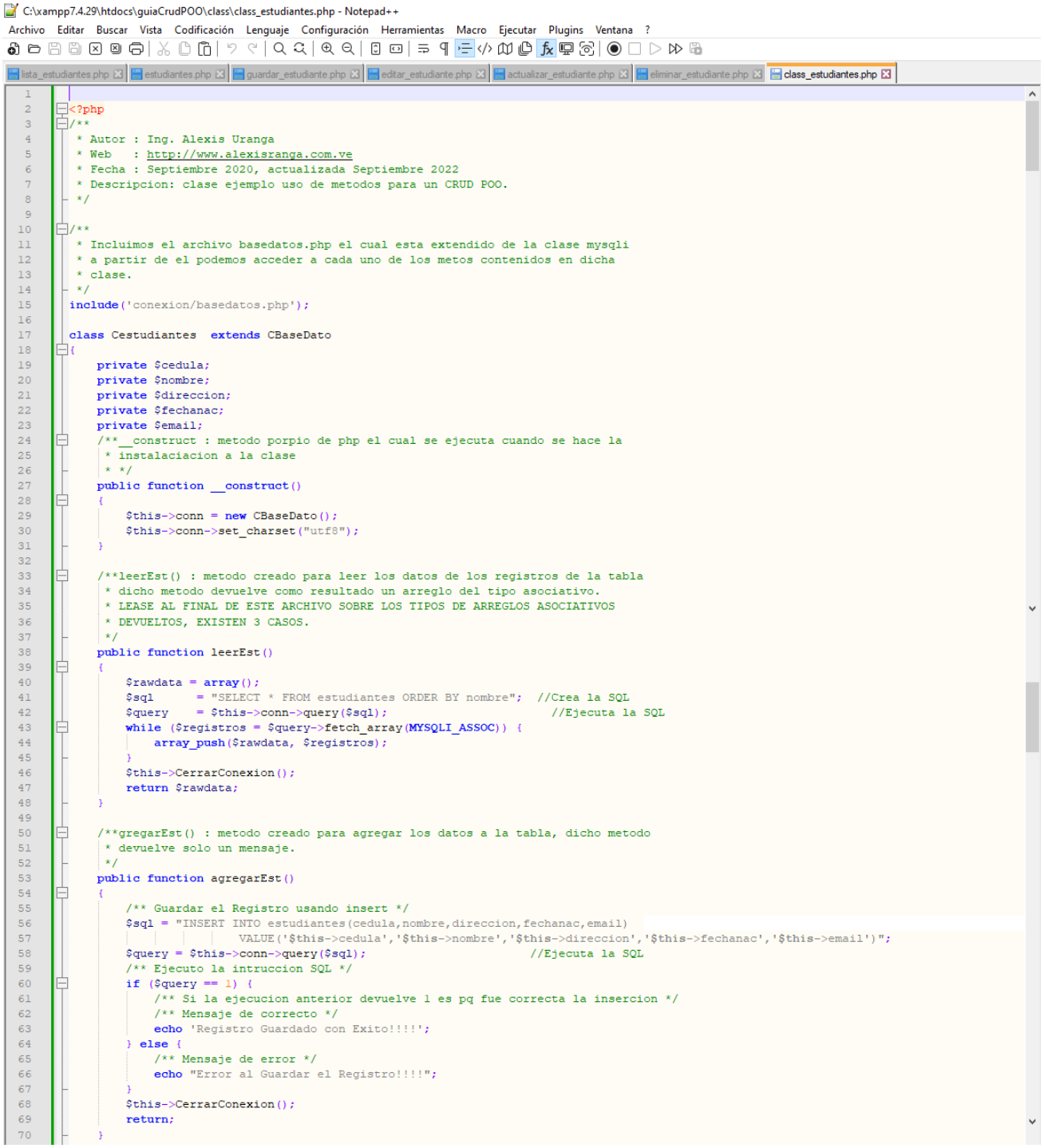
En la línea numero 34: se realizar el llamado o la ejecución al método **eliminarEst()**, el cual se encargará de eliminar el registro.

Todos los formularios utilizan como base la clase **class_estudiantes.php** ya que en ella es donde se encuentra toda la lógica del negocio es decir los métodos de leer, agregar, buscar, modificar, eliminar o lo que es lo mismo el CRUD, en dicha clase podemos encontrar las instrucciones SQL que realizan cada operación, esta es una parte de lo que llamamos las buenas practicas de desarrollo, sin embargo existen otras mas avanzadas como lo son hacer el uso de MVC o hacer uso de framework especializados para el acceso a datos.

Como lo dije al principio es una manera de hacer uso de la **Programación Orientada a Objetos** en php, existen muchas maneras de hacer el uso de las clases todo depende de la perspectiva del desarrollador.

Próxima guía la de MVC-POO, en esta próxima guía se corrige un problema de seguridad dado por el código utilizado en este ejemplo el cual es el uso de sentencias preparadas (cuando se crea la instrucción SQL), para asignar los valores de los campos que se van a guardar en las tablas, este se usa para evitar la inyección de SQL.

Contenido de la clase class_estudiantes.php (Nuevo)




```php
<?php
/**
 * Autor : Ing. Alexis Uranga
 * Web   : http://www.alexisranga.com.ve
 * Fecha : Septiembre 2020, actualizada Septiembre 2022
 * Descripcion: clase ejemplo uso de metodos para un CRUD POO.
 */

/**
 * Incluimos el archivo basedatos.php el cual esta extendido de la clase mysqli
 * a partir de el podemos acceder a cada uno de los metos contenidos en dicha
 * clase.
 */
include('conexion/basedatos.php');

class Cestudiantes  extends CBaseDato
{
    private $cedula;
    private $nombre;
    private $direccion;
    private $fechanac;
    private $email;
    /**__construct : metodo porpio de php el cual se ejecuta cuando se hace la
     * instalaciacion a la clase
     * */
    public function __construct()
    {
        $this->conn = new CBaseDato();
        $this->conn->set_charset("utf8");
    }

    /**leerEst() : metodo creado para leer los datos de los registros de la tabla
     * dicho metodo devuelve como resultado un arreglo del tipo asociativo.
     * LEASE AL FINAL DE ESTE ARCHIVO SOBRE LOS TIPOS DE ARREGLOS ASOCIATIVOS
     * DEVUELTOS, EXISTEN 3 CASOS.
     */
    public function leerEst()
    {
        $rawdata = array();
        $sql     = "SELECT * FROM estudiantes ORDER BY nombre";  //Crea la SQL
        $query   = $this->conn->query($sql);                     //Ejecuta la SQL
        while ($registros = $query->fetch_array(MYSQLI_ASSOC)) {
            array_push($rawdata, $registros);
        }
        $this->CerrarConexion();
        return $rawdata;
    }

    /**gregarEst() : metodo creado para agregar los datos a la tabla, dicho metodo
     * devuelve solo un mensaje.
     */
    public function agregarEst()
    {
        /** Guardar el Registro usando insert */
        $sql = "INSERT INTO estudiantes(cedula,nombre,direccion,fechanac,email)
                        VALUE('$this->cedula','$this->nombre','$this->direccion','$this->fechanac','$this->email')";
        $query = $this->conn->query($sql);                      //Ejecuta la SQL
        /** Ejecuto la intruccion SQL */
        if ($query == 1) {
            /** Si la ejecucion anterior devuelve 1 es pq fue correcta la insercion */
            /** Mensaje de correcto */
            echo 'Registro Guardado con Exito!!!!';
        } else {
            /** Mensaje de error */
            echo "Error al Guardar el Registro!!!!";
        }
        $this->CerrarConexion();
        return;
    }
```

```php
    /**buscarEst()  metodo creado para realizar la busqueda de un registro en particular. */
    public function buscarEst()
    {
        $rawdata = array();
        /** Crea la SQL  */
        $sql      = "SELECT * FROM estudiantes WHERE cedula=" . $this->cedula;
        /** Ejecuta la SQL */
        $query    = $this->conn->query($sql);
        /**Recorre el resultado de la SQL y lo almaceno en un arreglo */
        while ($registros = $query->fetch_array(MYSQLI_ASSOC)) {
            array_push($rawdata, $registros);
        }
        /**Cierra la conexion con el servidor */
        $this->CerrarConexion();
        /**Devuelve el arreglo */
        return $rawdata;
    }

    /**modificarEst() : metodo creado permite modificar un registro en particular */
    public function modificarEst()
    {
        /** Proceso: acceso a base de datos para guardar el Registro */
        /** Actualizar el Registro usando UPDATE */
        $sql = "UPDATE estudiantes SET nombre='$this->nombre',direccion='$this->direccion',fechanac='$this->fechanac',email='$this->ema
        /** Ejecuto la intruccion SQL */
        $query = $this->conn->query($sql);
        /** Si la ejecucion anterior devuelve 1 es pq fue correcta la insercion */
        if ($query == 1) {
            /** Mensaje de correcto */
            echo 'Registro Actualizado con Exito!!!!';
        } else {
            /** Mensaje de error */
            echo "Error al Guardar el Registro!!!!";
        }
        /**Cierra la conexion con el servidor */
        $this->CerrarConexion();
        return;
    }

    /**eliminarEst() : metodo creado para permitir eliminar un registro en particular */
    public function eliminarEst()
    {
        /** Proceso: acceso a base de datos para Eliminar*/
        /** Eliminar el Registro Usando DELETE */
        $sql = "DELETE FROM estudiantes WHERE cedula=$this->cedula";
        /* Ejecuto la intruccion SQL */
        $query = $this->conn->query($sql);
        /* Si la ejecucion anterior devuelve 1 es pq fue correcta la insercion   */
        if ($query == 1) {
            /** Mensaje de correcto */
            echo 'Registro Eliminado Con Exito!!!!';
        } else {
            /** Mensaje de error */
            echo "Error al Eliminar el Registro!!!!";
        }
        /**Cierra la conexion con el servidor */
        $this->CerrarConexion();
        return;
    }


    /**
     * Set the value of cedula
     *
     * @return  self
     */
    public function setCedula($cedula)
    {
        $this->cedula = $cedula;

        return $this;
    }

    /**
     * Set the value of nombre
     *
     * @return  self
     */
    public function setNombre($nombre)
    {
        $this->nombre = $nombre;

        return $this;
    }
```

```php
    /**
     * Set the value of nombre
     *
     * @return  self
     */
    public function setNombre($nombre)
    {
        $this->nombre = $nombre;

        return $this;
    }

    /**
     * Set the value of direccion
     *
     * @return  self
     */
    public function setDireccion($direccion)
    {
        $this->direccion = $direccion;

        return $this;
    }

    /**
     * Set the value of fechanac
     *
     * @return  self
     */
    public function setFechanac($fechanac)
    {
        $this->fechanac = $fechanac;

        return $this;
    }

    /**
     * Set the value of email
     *
     * @return  self
     */
    public function setEmail($email)
    {
        $this->email = $email;

        return $this;
    }
}

/**
 * Ejemplo de un arreglo asociativo del tipo campo=>valor, devuelto cuando se usa
```

PHP Hypertext Preprocessor file | length : 8.444 lines : 250 | Ln : 1 Col : 1 Pos : 1 | Windows (CR LF)

```php
/**
 * Ejemplo de un arreglo asociativo del tipo campo=>valor, devuelto cuando se usa
 * MYSQLI_ASSOC como parametro en fetch_array(MYSQLI_ASSOC)
 *
 * Para estos casos se usa la variable asociada en el foreach() y el nombre del
 * campo entre corchetes Ejemplo : $registros['cedula'], hace referencia al
 * campo cedula.
 * array(1) {
 *      [0]=> array(5) {
 *                  ["cedula"]=> string(8) "14125444"
 *                  ["nombre"]=> string(12) "Angel Caripa"
 *                  ["direccion"]=> string(12) "Santa Isabel"
 *                  ["fechanac"]=> string(10) "2000-08-10"
 *                  ["email"]=> string(21) "angelcaripa@gmail.com"
 *                      }
 *          }
 *
 * Ejemplo de un arreglo asociativo del tipo indice=>valor, devuelto cuando se usa
 * MYSQLI_NUM como parametro en fetch_array(MYSQLI_NUM).
 *
 * Para estos casos se usa la variable asociada en el foreach() y el indice del
 * campo entre corchetes Ejemplo : $registros[0], hace referencia al
 * campo cedula.
 *
 * array(1) {
 *      [0]=> array(5) {
 *                  [0]=> string(8) "14125444"
 *                  [1]=> string(12) "Angel Caripa"
 *                  [2]=> string(12) "Santa Isabel"
 *                  [3]=> string(10) "2000-08-10"
 *                  [4]=> string(21) "angelcaripa@gmail.com"
 *                     }
 *         }
 *
 * Existe un tercer metodo el cual es cuando se usa MYSQLI_BOOT, el cual
 * el arreglo devuelto contine un registro con nombre del campo
 * y otro registro con el indice del campo y es indiferente a momento de
 * leer el mismo sde puede acceder por el indice o el nombre del campo.
 *
 * array(1) {
 *      [0]=> array(10) {
 *                  [0]=> string(8) "14125444"
 *                  ["cedula"]=> string(8) "14125444"
 *                  [1]=> string(12) "Angel Caripa"
 *                  ["nombre"]=> string(12) "Angel Caripa"
 *                  [2]=> string(12) "Santa Isabel"
 *                  ["direccion"]=> string(12) "Santa Isabel"
 *                  [3]=> string(10) "2000-08-10"
 *                  ["fechanac"]=> string(10) "2000-08-10"
 *                  [4]=> string(21) "angelcaripa@gmail.com"
 *                  ["email"]=> string(21) "angelcaripa@gmail.com"
 *                      }
 *          }
 *
 * Es un arreglo mas grande ya que tiene un duplicada de cada campo.
 */
```

PHP Hypertext Preprocessor file | length : 8.444 lines : 250 | Ln : 1 Col : 1 Pos : 1 | Windows (CR LF)

Contenido de la clase mysqli (PHP 5, PHP 7, PHP 8)

```
class mysqli {
        /* Properties */
        public readonly int|string $affected_rows;
        public readonly string $client_info;
        public readonly int $client_version;
        public readonly int $connect_errno;
        public readonly ?string $connect_error;
        public readonly int $errno;
        public readonly string $error;
        public readonly array $error_list;
        public readonly int $field_count;
        public readonly string $host_info;
        public readonly ?string $info;
        public readonly int|string $insert_id;
        public readonly string $server_info;
        public readonly int $server_version;
        public readonly string $sqlstate;
        public readonly int $protocol_version;
        public readonly int $thread_id;
        public readonly int $warning_count;
        /* Methods */
        public __construct(
            string $hostname = ini_get("mysqli.default_host"),
            string $username = ini_get("mysqli.default_user"),
            string $password = ini_get("mysqli.default_pw"),
            string $database = "",
            int $port = ini_get("mysqli.default_port"),
            string $socket = ini_get("mysqli.default_socket")
        )
        public autocommit(bool $enable): bool
        public begin_transaction(int $flags = 0, ?string $name = null): bool
        public change_user(string $username, string $password, ?string $database): bool
        public character_set_name(): string
        public close(): bool
        public commit(int $flags = 0, ?string $name = null): bool
        public connect(
            string $hostname = ini_get("mysqli.default_host"),
            string $username = ini_get("mysqli.default_user"),
            string $password = ini_get("mysqli.default_pw"),
            string $database = "",
            int $port = ini_get("mysqli.default_port"),
            string $socket = ini_get("mysqli.default_socket")
        ): void
        public debug(string $options): bool
        public dump_debug_info(): bool
        public get_charset(): ?object
        public get_client_info(): string
        public get_connection_stats(): array
        public get_server_info(): string
        public get_warnings(): mysqli_warning|false
        public kill(int $process_id): bool
        public more_results(): bool
        public multi_query(string $query): bool
        public next_result(): bool
        public options(int $option, string|int $value): bool
        public ping(): bool
        public static poll(
            ?array &$read,
            ?array &$error,
```

```
            array &$reject,
            int $seconds,
            int $microseconds = 0
        ): int|false
        public prepare(string $query): mysqli_stmt|false
        public query(string $query, int $result_mode = MYSQLI_STORE_RESULT): mysqli_result|bool
        public real_connect(
            string $host = ?,
            string $username = ?,
            string $passwd = ?,
            string $dbname = ?,
            int $port = ?,
            string $socket = ?,
            int $flags = ?
        ): bool
        public real_escape_string(string $string): string
        public real_query(string $query): bool
        public reap_async_query(): mysqli_result|bool
        public refresh(int $flags): bool
        public release_savepoint(string $name): bool
        public rollback(int $flags = 0, ?string $name = null): bool
        public savepoint(string $name): bool
        public select_db(string $database): bool
        public set_charset(string $charset): bool
        public ssl_set(
            ?string $key,
            ?string $certificate,
            ?string $ca_certificate,
            ?string $ca_path,
            ?string $cipher_algos
        ): bool
        public stat(): string|false
        public stmt_init(): mysqli_stmt|false
        public store_result(int $mode = 0): mysqli_result|false
        public thread_safe(): bool
        public use_result(): mysqli_result|false
}
```

Descripción

- mysqli::$affected_rows — Gets the number of affected rows in a previous MySQL operation
- mysqli::autocommit — Turns on or off auto-committing database modifications
- mysqli::begin_transaction — Starts a transaction
- mysqli::change_user — Changes the user of the specified database connection
- mysqli::character_set_name — Returns the current character set of the database connection
- mysqli::close — Closes a previously opened database connection
- mysqli::commit — Commits the current transaction
- mysqli::$connect_errno — Returns the error code from last connect call
- mysqli::$connect_error — Returns a description of the last connection error
- mysqli::__construct — Open a new connection to the MySQL server
- mysqli::debug — Performs debugging operations
- mysqli::dump_debug_info — Dump debugging information into the log
- mysqli::$errno — Returns the error code for the most recent function call
- mysqli::$error_list — Returns a list of errors from the last command executed
- mysqli::$error — Returns a string description of the last error
- mysqli::$field_count — Returns the number of columns for the most recent query
- mysqli::get_charset — Returns a character set object

- mysqli::$client_info — Get MySQL client info
- mysqli::$client_version — Returns the MySQL client version as an integer
- mysqli::get_connection_stats — Returns statistics about the client connection
- mysqli::$host_info — Returns a string representing the type of connection used
- mysqli::$protocol_version — Returns the version of the MySQL protocol used
- mysqli::$server_info — Returns the version of the MySQL server
- mysqli::$server_version — Returns the version of the MySQL server as an integer
- mysqli::get_warnings — Get result of SHOW WARNINGS
- mysqli::$info — Retrieves information about the most recently executed query
- mysqli::init — Initializes MySQLi and returns an object for use with mysqli_real_connect()
- mysqli::$insert_id — Returns the value generated for an AUTO_INCREMENT column by the last query
- mysqli::kill — Asks the server to kill a MySQL thread
- mysqli::more_results — Check if there are any more query results from a multi query
- mysqli::multi_query — Performs one or more queries on the database
- mysqli::next_result — Prepare next result from multi_query
- mysqli::options — Set options
- mysqli::ping — Pings a server connection, or tries to reconnect if the connection has gone down
- mysqli::poll — Poll connections
- mysqli::prepare — Prepares an SQL statement for execution
- mysqli::query — Performs a query on the database
- mysqli::real_connect — Opens a connection to a mysql server
- mysqli::real_escape_string — Escapes special characters in a string for use in an SQL statement, taking into account the current charset of the connection
- mysqli::real_query — Execute an SQL query
- mysqli::reap_async_query — Get result from async query
- mysqli::refresh — Refreshes
- mysqli::release_savepoint — Removes the named savepoint from the set of savepoints of the current transaction
- mysqli::rollback — Rolls back current transaction
- mysqli::savepoint — Set a named transaction savepoint
- mysqli::select_db — Selects the default database for database queries
- mysqli::set_charset — Sets the client character set
- mysqli::$sqlstate — Returns the SQLSTATE error from previous MySQL operation
- mysqli::ssl_set — Used for establishing secure connections using SSL
- mysqli::stat — Gets the current system status
- mysqli::stmt_init — Initializes a statement and returns an object for use with mysqli_stmt_prepare
- mysqli::store_result — Transfers a result set from the last query
- mysqli::$thread_id — Returns the thread ID for the current connection
- mysqli::thread_safe — Returns whether thread safety is given or not
- mysqli::use_result — Initiate a result set retrieval
- mysqli::$warning_count — Returns the number of warnings from the last query for the given link

✓ Contenido de la clase mysqli, fue extraído de Zeal(Zeal is an offline documentation browser for software developers.), puedes descargar desde: https://zealdocs.org/

✓ Descarga el archivo guiaCrudPoo desde: https://www.dropbox.com/s/ps8v9mhza23nd5i/guiaCrudPOO.rar?dl=0

Ing. Alexis Uranga
+58(424)520.65.39 / +58(416)451.52.24
Desarrollo de Portales Web, Aplicaciones Móviles, Cableado Estructurado, Redes en General Cámaras de Seguridad, Consultoría en el Área de Informática, Docente Universitario.
Empresa: mastertradeca.com | **Email:** info@mastertradeca.com
Web : alexisuranga.com.ve | **Twitter** : @alexisuranga | **Facebook** : alexis.uranga
Instagram : alexisuranga.com.ve | **Linkedin** : alexis-uranga