UNIVERSIDAD LUTERANA SALVADOREÑA

FACULTAD DE CIENCIAS DEL HOMBRE Y LA NATURALEZA

LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

Oolong-PHP framework

Programación II

CICLO I-2017

INTEGRANTES:

Karina Vanessa Molina Servellón
Emerson Elenilson Martínez Maravilla
Misael Antonio Mejía Bonilla
Jessica Patricia Beltran Torres
William Stanley Navas

DOCENTE:

LIC. JOSÉ LUIS ALVARADO AGUILAR

Manual del programador:

Oolong-PHP framework: Tal cual su nombre lo dice es un framework desarrollado en PHP bajo los mejores y mas recientes estándares de programación y desarrollo.

Como lo son:

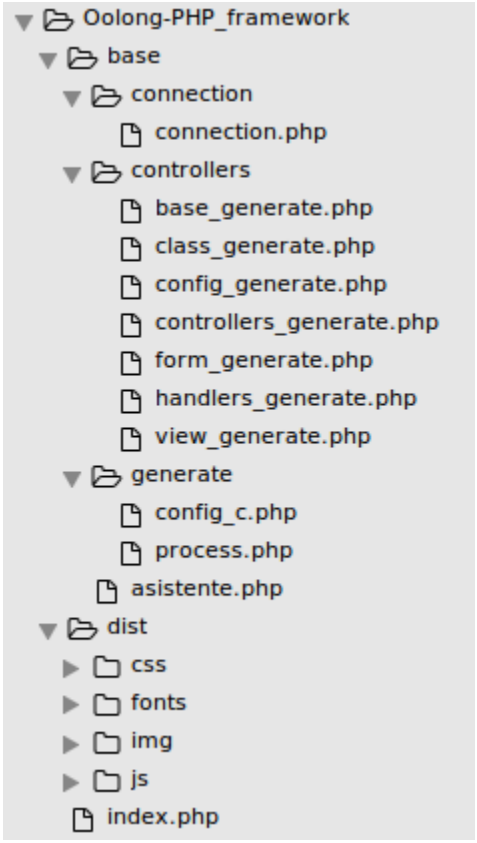
1. Modelo vista controlador(MVC).
2. Programación orientada a objetos(POO)
3. PHP 7
4. HTLM5

Como ya se sabe los framework son de mucha utilidad al momento de hacer un nuevo proyecto ya que podrían ayudarnos a ahorrarnos mucho tiempo en la elaboración de tarea comunes(CRUD).

La lógica de Oolong-PHP framework es la lectura y mapeo de una base de datos MySQL la cual tiene como misión proveer toda la información necesaria para el desarrollo de nuestro proyecto.

Se crean manejadores, controladores, clases y vistas como archivos principales y ademas partiendo de una pronta conexión con nuestro servidor de base de datos se creara un archivo de conexión único y funcional con sus respectivas funciones adicionales como lo son las que ejecutan las consultas de SELECT.

Estructura de Oolong-PHP framework



Requisitos para ejecutar Oolong-PHP framework

1. Tener instalado PHP 7 de preferencia para evitar errores.
2. Versión de servidor MySQL y Apache de preferencia un actualizado.
3. Contar con una base de datos MySQL funcional en el servidor la cual sera la base para esta generación de un proyecto.

    3.1. Todas las tablas de dicha base de datos deben de tener una única llave primaria(PRIMARY KEY) y deben de ser auto incrementables(auto_increment).

    3.2. No pueden haber mas campos que sean auto incrementables.

    3.3. Los campos que sean de tipo timestamp deben de tener el siguiente atributo ‘on update CURRENT_TIMESTAMP’ para que se llenen solos ya que el framework ignorar este campo.

    3.4. Si se admiten las llaves foraneas.

4. Saber usuario y contraseña de tu servidor MySQL.
5. Dar todo los permisos necesarios de lectura y escritura ala carpeta de publicación del servidor y también a la carpeta del instalador.

    5.1. En Linux: Abrimos la terminal y como Súper usuario, tecleamos. Ya sobre el directorio de tu servidor: chmod -R 777 “Oolong-PHP_framework”. Esto le dará un permiso universal al directorio para poder ejecutarse.

    Ejemplo de Linux.

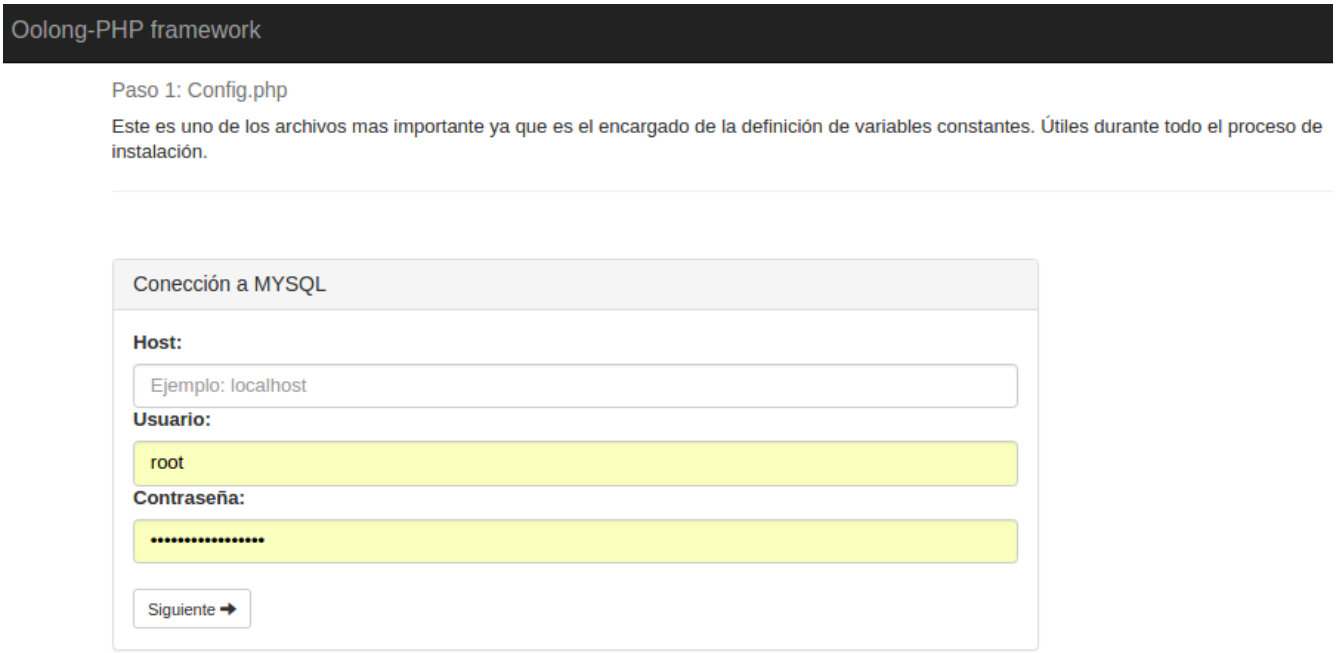
    5.2. En Windows no tendría que tener problema pero igual asegurate que tienes los permisos necesarios para trabajar sobre la carpeta de publicación de tu servidor.

6. Tener cualquier navegador web.

Pasos a seguir para generar tu proyecto y explicación de cada función.

Paso 1: Ingresa al siguiente url en tu navegador http://127.0.0.1/Oolong-PHP_framework/

Nota: Si estas trabajando en local utiliza 127.0.0.1 o localhost de lo contrario utiliza la dirección que convenga. Se abrirá esta ventana:



El siguiente formulario es el encargado de recoger y generar un archivo config.php ubicado en la carpeta 'connection' del framework el cual contendrá las variables constantes que servirán para establecer conexión a tu servidor de base de datos durante todo el proceso de instalación.

La información anteriormente recibida se envían al archivo config_c.php de la carpeta 'generate' que esta en la raíz del framework.

config_c.php

Es el encargado de recibir y enviar a un método la información del formulario anterior.

config_c.php

```php
<?php
//Creación y validación del primer archivo de conección y testeo de config.php
include_once '../controllers/config_generate.php';
//Instanciación de los metodos
$conf = new config_generate();
//Enviamos 3 variables a al metodo instanciado y esperamos respuesta
$conf->createConfig($_POST['txtHost'],$_POST['txtUser'],$_POST['txtContr']);
?>
```

Se instancia la clase contenedora del método y se envían los datos. Los datos se mandan al archivo 'config_generate.php' ubicado en la carpeta 'controllers' que es el encargado final de generar el archivo 'config.php'

config_generate.php

Aquí esta el método que creara el archivo config.php de la carpeta 'connection' en la raíz del framework.

config_generate.php

```php
<?php
class config_generate{
    //Metodos
    function createConfig($Host,$User,$Password){
        if(file_exists("../connection/")){
            //echo "Directorio: /connection, ya existe<br>";
        }else{
            mkdir("../connection");
            chmod("../connection", 0777);
            //echo "Directorio: /connection<br>";
        }
        if(file_exists("../connection/config.php")){
            echo "Archivo: connection/config.php, ya existe.<br>";
        }else{
            //fopen crea y abre un archivo con los permisos necesarios para la escrtura
            $gestor = fopen("../connection/config.php", "w");
            chmod("../connection/config.php",0777);
            $varsalida = "<?php\n";//llenar el config.php
            $varsalida .= "define(\"HOST\",\"".$Host."\");\n";
            $varsalida .= "define(\"USUARIO\",\"".$User."\");\n";
            $varsalida .= "define(\"CLAVE\",\"".$Password."\");\n";
            $varsalida .= "\n?>";
            fputs($gestor,$varsalida);
            fclose($gestor);
            echo "<p>Archivo principal: config.php, creado correctamente</p>";
        }

    }
}
?>
```

Nota: Basándonos en el paradigma de programación orientada a objetos. Tenemos clases con sus respectivas funciones que son las que instanciamos desde paginas externas.

Ejemplo: Así instanciamos este método. '$conf = new config_generate();' es lo que presenciamos en **config_c.php** un claro ejemplo de instancia que a partir de este punto se utilizara muy frecuentemente.

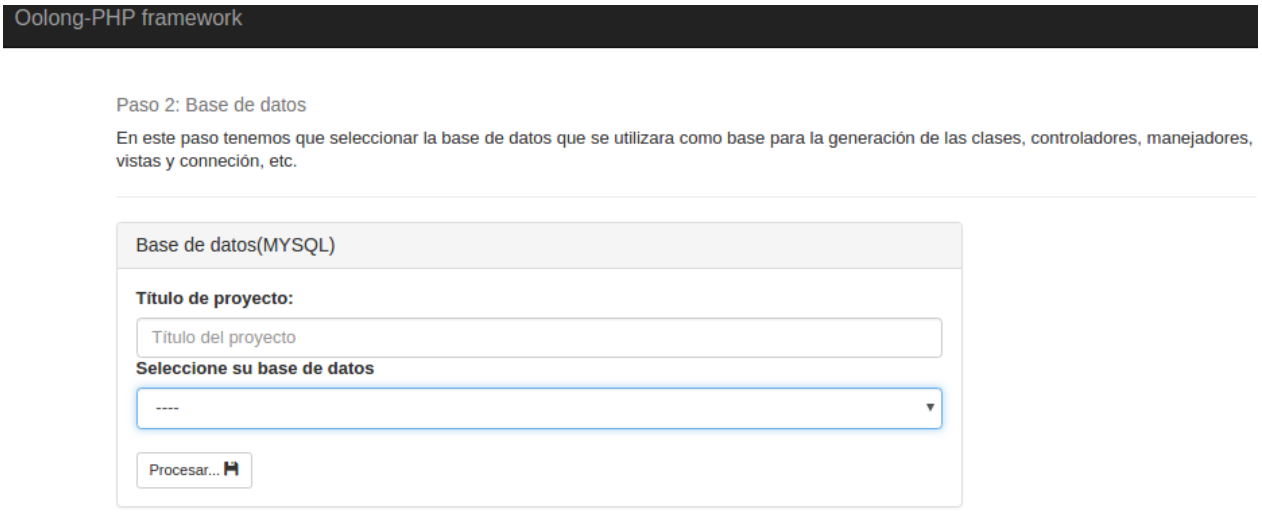
config.php

Todo lo anterior para llegar a este archivo y ver que queda de la siguiente manera. Es un archivo sencillo pero demasiado útil. Mediante esta lógica de creación de archivos se fundamenta todo lo que le resta al framework en la creación automática de archivos.

config.php

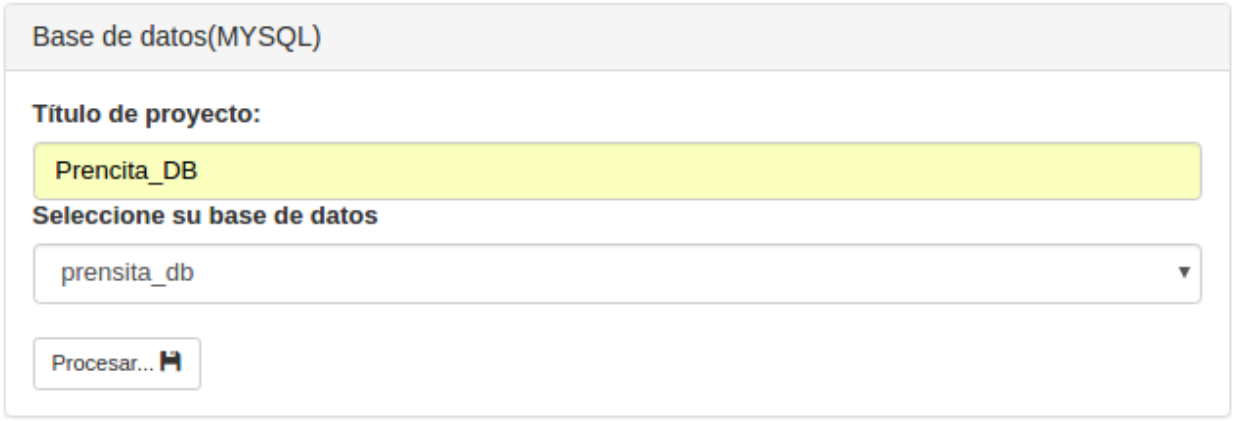
```php
<?php
define("HOST","127.0.0.1");
define("USUARIO","root");
define("CLAVE","162411905046070-5");

?>
```

Paso 2: El formulario anteriormente explicado nos redirigirá automáticamente ala segunda ventana, el cual es otro formulario.



Si los datos del archivo config.php son los correcto para tu servidor de base de datos(MySQL) en esta lista desplegable de este formulario estarán disponible las bases de datos de tu servidor. Ahí ya debe de estar la base de dato ala cual nos interesa generarle todo el proyecto.

En este formulario llenamos con el nombre que sera el cual le daremos al proyecto nuevo. Ejemplo: Para un sistema de biblioteca el nombre seria Biblioteca de la ULS y debemos de seleccionar la base de datos. Y Clic en siguiente y esperar unos segundos.



En este caso nuestra base de natos se llama prensita_db. Al dar clic en ‘Procesar’ se empezara a crear nuestros proyecto, no obstante la información anteriormente recogida se enviá y procesa en distintos métodos en el archivo ‘process.php’ en la carpeta ‘generate’ de la raíz del framework.

A continuación se explicara cada proceso interno que ocurre cuando presionamos el botón ‘Procesar’, es importante saber que todo esto ocurre en segundos. Es un proceso verdaderamente instantáneo.

process.php

En este archivo se reciben y enviá la información escrita en el segundo formulario. Aquí se trabaja con el nombre de la base de datos que es el seleccionado anteriormente.

Este código es bastante grande por lo cual nos concentraremos en los métodos e instancias.

process.php

```php
<?php
//Inclución de archivos que contienen cada clase generadora
include_once '../connection/config.php';
include_once '../controllers/base_generate.php';
include_once '../controllers/class_generate.php';
include_once '../controllers/controllers_generate.php';
include_once '../controllers/view_generate.php';
include_once '../controllers/form_generate.php';
include_once '../controllers/form_generate.php';
include_once '../controllers/handlers_generate.php';
//Instanciación de los metodos
$insgenerate = new base_generate();
$classgenerate = new class_generate();
$controllers = new controllers_generate();
$view = new view_generate();
$form = new form_generate();
$handlers = new handlers_generate();
?>
```

Como primer punto incluimos con ‘include_once’ todos los archivos que utilizaremos que son los que contienen todas las clases necesarias para toda la generación del nuevo proyecto.

Los incluimos en process.php esto nos permitirá gozar de las clases de dichos archivo, y los instanciamos al mismo tiempo en variables diferentes que se utilizara mas adelante. Por ahora tenemos 6 instancias.

```php
<?php
echo "Estamos trabajando por favor espere...<br>";
//creación de estructura base de directorios de proyectos
$insgenerate->createBase();
$insgenerate->recurse_copy("../../dist","../../full-project/dist");
echo "Se completó la estructura base del proyecto...<br>";
//Creación de contenidos dinamicos, index y conexión
$insgenerate->createConnection($_POST['txtName'],$_POST['txtbase']);
//Empezando recorrido para la generación de las clases con sus respectivos metodos get
y set
$classgenerate->CreateClass($_POST['txtbase']);
//Creación de controladores de proyecto
$controllers->CreateControllers($_POST['txtbase']);
//Aqui se manda información para la cración de las vistas(index principal)
$view->createBaseIndex($_POST['txtName'],$_POST['txtbase']);
//Vistas principales de cada tabla
$view->createBaseView($_POST['txtName'],$_POST['txtbase']);
//Para las vistas de guardar con formularios
$form->createBaseSave($_POST['txtName'],$_POST['txtbase']);
//Aqui se manda información para la clase que elabora un manejador por tabla para
controlar funciones de CRUD
$handlers->createBaseHandlers($_POST['txtName'],$_POST['txtbase']);
?>
```

A continuación se explicara que enviá y que devuelve cada método ya que ala mayoría se le pasan 2 parámetros.

Explicación de cada instancia.

```
//Instanciación de los metodos
$insgenerate = new base_generate();
$classgenerate = new class_generate();
$controllers = new controllers_generate();
$view = new view_generate();
$form = new form_generate();
$handlers = new handlers_generate();
```

Cada instancia de clase anteriormente descrito sera brevemente detallada para su mejor comprensión. La lógica a utilizar es la misma anteriormente descrita.

```
$insgenerate->createBase();
```

Esta instancia es la primera puesto a que crea la estructura base de directorio, osea todo los directorios que sirven para el llenado con archivos de las instancias de métodos siguientes.

Aquí no enviamos ningún parámetro. Y a continuación la información se enviá y recibe en '../controllers/base_generate.php'.

base_generate.php

Este archivo genera toda la estructura base del proyecto, incluyendo todos los directorios y sub directorios que sirven para la mejor organización de todos los archivos.

Este archivo contiene una clase llamada 'base_generate' que es la principal de este archivo y en la instancia anterior enviamos datos asía esta clase pero a una de sus funciones que en este caso seria un objeto llamado 'createBase()'.

```
function createBase()
```

Esta función es la encargada de generar el directorio principal llamado 'full-project' y a su ves una serie de sub-directorios que a mi parecer son los más comunes en todo tipo de proyecto.

Estas carpetas las crea esta función. Por el momento son carpetas vaciás.

"class","controllers","handlers","includes","view","connection","dist"

También se crea el archivo de conexión vació por el momento que sera el único y universal y que llenaremos más adelante en una instancia aparte.

A continuación el código fuente de dicha clase.

```php
class base_generate {
    function createBase(){
        if(file_exists("../../full-project")){
            echo "Directorio: /full-project, ya existe<br>";
        }else{
            mkdir("../../full-project");
            chmod("../../full-project", 0777);
            echo "Directorio: /full-project, creado correctamente<br>";
        }
        $directory = array("class","controllers","handlers","includes","view","connection","dist");
        for ($i=0; $i <= 6; $i++){
            if(file_exists("../../full-project/".$directory[$i])){
                echo "Directorio: /".$directory[$i].", ya existe<br>";
            }else{
                mkdir("../../full-project/".$directory[$i]);
                chmod("../../full-project/".$directory[$i], 0777);
                echo "Directorio: /".$directory[$i].", creado correctamente<br>";
            }
        }
        if(file_exists("../../full-project/.htaccess")){
            echo ".htaccess ya existe<br>";
        }else{
            $gestor3 = fopen("../../full-project/.htaccess", "x+");
            chmod("../../full-project/.htaccess", 0777);
            echo ".htaccess creado correctamente<br>";
            fclose($gestor3);
        }
        if(file_exists("../../full-project/connection/connection.php")){
            echo "connection.php ya existe<br>";
        }else{
            $gestor2 = fopen("../../full-project/connection/connection.php", "x+");
            chmod("../../full-project/connection/connection.php", 0777);
            echo "connection.php creado correctamente<br>";
            fclose($gestor2);
        }
    }
```

Con una serie de validaciones con la estructura de control if/else nos aseguramos de ordenar todo y comprobar si existe o no, de esa manera evitamos errores o sobrescritas.

Glosario:

mkdir: Crea un directorio.

file_exists: Comprueba si existe un fichero o directorio.

echo: imprime en pantalla.

chmod: Cambia el modo de un fichero.

fopen: Abre un fichero o directorio.

fclose: Cierra un puntero a un archivo abierto.

Hasta este punto solo hemos creado una estructura de directorio bastante básica que se ira complicando cada vez más, ademas ya tenemos listo el archivo de conexión listo para su escritura.

```php
$insgenerate->recurse_copy("../../dist","../../full-project/dist");
```

Esta instancia es la encargada de conectar a una función que es la que genera una copia completa de la carpeta “dist” que es la que contiene todos los archivos de estilo(CSS y JavaScritp), los copia a la carpeta principal del nuevo proyecto generado.

Al igual que la función anterior esta nueva instancia “recurse_copy” nos manda dos parámetros al archivo de **base_generate.php** en el cual definimos la función de la siguiente manera.

La primer parámetro contiene la url de la carpeta que copiaremos y el segundo parámetro contiene la url de destino de donde copiaremos la carpeta completa.

Aquí encontraremos la función que recibe los dos parámetros enviados anteriormente y los procesa.

```php
function recurse_copy($src,$dst){
```

```php
function recurse_copy($src,$dst){
    $dir = opendir($src);
    if(file_exists($dst)){
        echo "Directorio: ".$dst."/, ya existe<br>";
    }else{
        mkdir($dst);
        chmod($dst, 0777);
        echo "Directorio: ".$dst."/, creado correctamente<br>";
    }
    while(false !==($file = readdir($dir))){
        if (($file != '.') && ($file !='..')){
            if (is_dir($src.'/'.$file)){
                $this->recurse_copy($src.'/'.$file,$dst.'/'.$file);
            }else{
                copy($src.'/'.$file,$dst.'/'.$file);
                chmod($dst.'/'.$file, 0777);
            }
        }
    }
    closedir($dir);
}
```

Esta función recibe ambos parámetros y los evaluá en diferentes partes en forma de validación con if/else y nacionalmente los recorre con ‘while’ a modo de crear una copia recursiva de todos los archivos de la carpeta de origen a la carpeta de destino.

Glosario:

opendir: Abre un directorio para poder ser usado en llamadas poesteriores.

while: Estructura de control(bucle) repetitiva que ejecuta las sentencias anidadas en ella

readdir: Lee una entrada desde un gestor de directorio abierto anteriormente.

if/else: Permite la ejecución condicional de fragmentos de código, la expresión es evaluada a su valor booleano. Si es true se ejecuta si no lo pasa a su contraparte ‘else’ si es que lo tiene.

is_dir: Indica si el nombre de archivo es un directorio.

$this: Ase referencia hacia en mismo como un objeto.

Copy: Realiza una copia de ficheros.

closedir: Cierra un gestor de directorio abierto anteriormente.

Hasta este punto el framework a realizado una copia completa de todos los archivos de la carpeta ‘dist’ que esta ubicada por defecto en la carpeta raíz del framework, que contiene todos los archivos de del tema bootstrap que es el que usamos por defecto para los proyectos generados

```
//Creación de contenidos dinamicos, index y conexión
$insgenerate->createConnection($_POST['txtName'],$_POST['txtbase']);
//Empezando recorrido para la generación de las clases con sus respectivos metodos get y set
```

Esta nueva instancia al igual que las dos anteriores manda la información a ‘ base_generate.php’ junto con dos parámetros que son el nombre del proyecto que se esta generando y nombre de la base de datos.

```
function createConnection($ProyectName,$DbName){
```

La función recibe dos parámetros y los procesa, de tal modo que genera un archivo llamado connection.php que es el encargado de conectar y realizar las consultas en la basa de datos ya en el proyecto generado.

```
60      function createConnection($ProyectName,$DbName){
61          $fp = fopen("../../full-project/connection/connection.php", 'w');
62          $varsalida = "<?php
63  function connect(){ ...
78  }
79  function Consultas(\$conect, \$sql){ ...
113 }
114 function Selects(\$conect, \$sql){ ...
125 }
126 ?>";
127         fputs($fp, $varsalida);
128     }
129 }
```

Aquí tenemos la función ya completa, la cual tiene una variable ’$varsalida’ que es la que contiene información que sera escrita en el nuevo archivo que apreciamos ‘connection.php’

El archivo anteriormente creado contendrá código PHP con una clase y 3 funciones primarias, que harán la conexión, las consultas y una ultima que hará select.

Hasta este punto tenemos un archivo de conexión funcional en todas las peticiones que realice el proyecto.

```
//Empezando recorrido para la generación de las clase
respectivos metodos get y set
$classgenerate->CreateClass($_POST['txtbase']);
```

Es una instancia a un método, función que recibe y procesa los datos de la siguiente manera.

Los archivos son mandados a class_generate.php.

class_generate.php

Este archivo php contiene una clase llamada ‘class_generate’ que es el encargado de generar archivos en la carpeta ‘class’ del nuevo proyecto.

Se genera una clase totalmente funcional con sus respectivos metodos. Las clases son las encadas de encapsular distinta información en ciertas variables.

```
function CreateClass($db){
```

Solo recibe un único parámetro el cual es el nombre de la base de datos.

```
<?php
include_once '../connection/connection.php';
class class_generate {
    function CreateClass($db){
        $sql_tablas = "SHOW TABLES from ".$db."";
        $total_tables = AllDB(con(), $sql_tablas,2);
        if(file_exists("../../full-project/class")){
            echo "Trabajando en directorio: /class<br>";
        }else{
            mkdir("../../full-project/class");
            chmod("../../full-project/class", 0777);
            echo "Directorio: /class, creado correctamente<br>";
        }
        foreach($total_tables as $value){
            foreach ($value as $dato) {
                $find = array('_', '-', ' ');
                $TableName = strtolower(str_replace($find, "", $dato));
                if(file_exists("../../full-project/class/class_".$TableName.".php")){
                    echo "Archivo: class/class_".$TableName.".php ya existe<br>";
                }else{
                    $gestor2 = fopen("../../full-project/class/class_".$TableName.".php", "w");
                    chmod("../../full-project/class/class_".$TableName.".php",0777);
                    echo "Archivo: class/class_".$TableName.".php, creado correctamente<br>";
                    $varsalida = "<?php\n";
                    $varsalida .="class class_".$TableName."{\n";
                    $sql_campos = "SHOW COLUMNS FROM ".$db."".$dato;
                    $total_Colums = AllDB(con(), $sql_campos,2);
                    foreach ($total_Colums as $datosC) {
                        $varsalida .= "\tprivate $".str_replace($find, "", strtolower($datosC['
                            Field'])).";\n";
                    }
                    //Aqui
```

```
    $sqlclaves = "SELECT table_name, referenced_table_name,
        referenced_column_name, column_name FROM information_schema.
        key_column_usage WHERE referenced_table_schema = '$db' AND
        referenced_table_name is not NULL";
    $total_ColumsID = AllDB(con(), $sqlclaves,2);
    foreach ($total_ColumsID as $foraneos=>$datosfor) {
        if($dato == $datosfor['table_name']){
            $varsalida .= "\n\tpublic \$Key".$datosfor['referenced_table_name']."
                = ";
            $varsalida .="\"SELECT * FROM ".$datosfor['referenced_table_name']."\"
                ;";
        }
    }
    $varsalida .= "\n\t//Métodos set y get para encapsulamiento";
    foreach($total_Colums as $datosMt){
        $varsalida .= "\n\tpublic function get";
        $varsalida .= strtolower(str_replace($find, "", $datosMt['Field']))."(){"
            ;
        $varsalida .= "\n\t\treturn \$this->".strtolower(str_replace($find, "",$
            datosMt['Field'])).";";
        $varsalida .= "\n\t}\n";

        $varsalida .= "\n\tpublic function set";
        $varsalida .= strtolower(str_replace($find, "", $datosMt['Field']))."(\$"
            .strtolower(str_replace("_", "",$datosMt['Field']))."){\n";
        $varsalida .= "\t\t\$this->".strtolower(str_replace($find, "", $datosMt['
            Field'])." = \$".strtolower(str_replace($find, "",$datosMt['Field'])
            ).";";
        $varsalida .= "\n\t}\n";
    }
    $varsalida .= "\n}\n?>";
    fputs($gestor2,$varsalida);
    fclose($gestor2);
}
```

Todo este código php es valido para la creación de todos los archivos que podemos apreciar en la recién carpeta generada class.

Los archivos recién generados tienen esta apariencia.

```
<?php
class class_noticias{
    private $id;
    private $titulo;
    private $cuerpo;
    private $categoria;
    private $fecha;
    private $imagen;
    private $iduser;

    public $Keyusuarios = "SELECT * FROM usuarios";
    //Métodos set y get para encapsulamiento
    public function getid(){
        return $this->id;
    }

    public function setid($id){
        $this->id = $id;
    }

    public function gettitulo(){
        return $this->titulo;
    }

    public function settitulo($titulo){
        $this->titulo = $titulo;
    }

    public function getcuerpo(){
        return $this->cuerpo;
    }

    public function setcuerpo($cuerpo){
        $this->cuerpo = $cuerpo;
    }
```

Glosario:

array: O matrices. Los arrays son esenciales para almacenar, manejar y operar sobre conjuntos de variables.

strtolower: Convierte una cadena a minúsculas.

str_replace: Reemplaza todas las apariciones del string buscado con el string de reemplazo.

```php
//Creacion de controladores de proyecto
$controllers->CreateControllers($_POST['txtbase']);
//Aqui se manda información para la cración de las vistas(index
```

Es una instancia que manda un único parámetro que es el nombre de base datos a ‘controllers_generate.php’

que esta dentro de la carpeta ‘controllers’ en la carpeta principal del framework.

controllers_generate.php

Este archivo funciona igual al anterior en lo que respecta a la generación de archivos. Hace un par de consultas y las procesa e imprime.

```php
<?php
include_once '../connection/connection.php';
//Clase generadora de las funciones de CRUD
class controllers_generate{
    function CreateControllers($db){
        $sql_tablas = "SHOW TABLES from ".$db."";
        $total_tables = AllDB(con(), $sql_tablas,2);
        if(file_exists("../../full-project/controllers")){
            echo "Trabajando en directorio: /controllers<br>";
        }else{
            mkdir("../../full-project/controllers");
            chmod("../../full-project/controllers", 0777);
            echo "Directorio: /controllers, creado correctamente<br>";
        }
        foreach($total_tables as $value){
            //var_dump($value);
            foreach($value as $dato) {
                //var_dump($dato);
                $find = array('_', '-', ' ');
                $TableName = strtolower(str_replace($find, "", $dato));
                if(file_exists("../../full-project/controllers/".$TableName."Controllers.php")){
                    echo "Archivo: controllers/".$TableName."Controllers.php ya existe<br>";
                }else{
                    $gestor = fopen("../../full-project/controllers/".$TableName."
                        Controllers.php", "w");
                    chmod("../../full-project/controllers/".$TableName."Controllers.php",0777);
                    echo "Archivo: controllers/".$TableName."Controllers.php, creado
                        correctamente<br>";
                    $varsalida = "<?php";
                    $varsalida .= "\n//Clase para funciones basicas de guardar, modificar y
                        eliminar";
                    $varsalida .= "\nclass ".$TableName."Controllers extends class_".$TableName.
                        "{\n";
                    $varsalida .= "\tpublic function saveData(\$connection,\$obj".ucfirst($
                        TableName).("){";
                    //Consulta de guardar
                    $varsalida .= "\n\t\t\$variableSql = \"INSERT INTO ".$dato."\";";
                    //Primera opcion de consulta de insert para no mostrar los id(PrimaryKey)
```

Solo colocare un fragmento de código ya que es demasiado, Esta clase genera un archivo ‘controllers.php’ en la carpeta ‘controllers’ del nuevo proyecto que ha su vez contiene funciones de CRUD. Al igual que lo anterior una por tabla.

Este es el contenidos del archivo generado.

```php
//Clase para funciones basicas de guardar, modificar y eliminar
class noticiasControllers extends class_noticias{
    public function saveData($connection,$objNoticias){
        $variableSql = "INSERT INTO noticias";
        $variableSql .= "(titulo,cuerpo,categoria,fecha,imagen,id_user)";
        $variableSql .= " VALUES(";
        $variableSql .= "'".$objNoticias[0]."','".$objNoticias[1]."','".$objNoticias[2]."','".$
            objNoticias[3]."','".$objNoticias[4]."','".$objNoticias[5]."');";
        return Consultas($connection, $variableSql);
    }
    public function UpdateData($connection,$objNoticias){
//Contenido para funcion de modificar
        $variableSql = "UPDATE noticias SET ";
        $variableSql .= "titulo='".$ObjNoticias['1']."', cuerpo='".$ObjNoticias['2']."',
            categoria='".$ObjNoticias['3']."', fecha='".$ObjNoticias['4']."', imagen='".$
            ObjNoticias['5']."', id_user='".$ObjNoticias['6']."' WHERE id ='".$ObjNoticias['0'].
            ';";
        return Consultas($connection, $variableSql);
    }
    public function DeleteData($connection,$objId){
        $sql = "DELETE FROM noticias WHERE id = ".$objId.";";
        return Consultas($connection,$sql);
    }
}
?>
```

Glosario:

ucfirst: Convierte el primer carácter de una cadena a mayúsculas.

array_push: Inserta uno o más elementos al final de un array.

implode: Une elementos de un array en un string.

count: Cuenta todos los elementos de un array o algo de un objeto.

```php
//Aqui se manda información para la cración de las vistas(index
principal)
$view->createBaseIndex($_POST['txtName'],$_POST['txtbase']);
//Vistas principales de cada tabla
```

Esta nueva instancia enviá 2 parámetros a un clase ‘view_generate.php’ que recibe y procesa los datos y crea un archivo que sera un menú que contiene todos los CRUD enlazados de cada tablas. Sera un index.php

view_generate.php

Este archivo ubicado en la carpetas ‘controllers’ de la raíz del framework es el que contiene las siguientes clases.

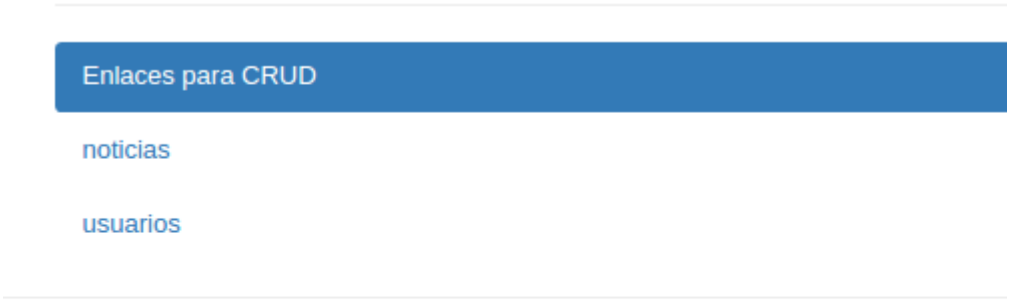
```
function createBaseIndex($ProjectN,$DbName){
  if(file_exists("../../full-project/index.php")){
    echo "index.php ya existe<br>";
  }else{
    $gestor1 = fopen("../../full-project/index.php", "w");
    chmod("../../full-project/index.php", 0777);
    //esto a creado un index.php en el directorio principal del nuevo proyecto
    $varsalida1 = "<!DOCTYPE html>
<html lang=\"es\">
<head>
  <meta charset=\"utf-8\">
  <meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">
  <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">
  <meta name=\"description\" content=\"\">
  <meta name=\"author\" content=\"Cíborg Ascend\">
  <link rel=\"icon\" href=\"dist/img/logos/favicon.ico\">
  <title>Menu principal de enlaces</title>
  <link href=\"dist/css/bootstrap.min.css\" rel=\"stylesheet\">
  <style type=\"text/css\">
    body {
      padding-top: 50px;
      padding-bottom: 20px;
    }
    #message{
      height: 30px;
    }
  </style>
</head>
<body>
<nav class=\"navbar navbar-inverse navbar-fixed-top\">
 <div class=\"container-fluid\">
   <div class=\"navbar-header\">
     <button type=\"button\" class=\"navbar-toggle collapsed\" data-toggle=\"collapse\"
       data-target=\"#navbar\" aria-expanded=\"false\" aria-controls=\"navbar\">
       <span class=\"sr-only\">Toggle navigation</span>
       <span class=\"icon-bar\"></span>
       <span class=\"icon-bar\"></span>
```

La clase ‘createBaseIndex’ genera un index.php en la nueva ruta principal del nuevo proyecto y crea un menú con tema de ‘bootstrap’.

Como pueden ver todo el contenido que esta en $varsalida1 es lo que este index.php contendra.

```
    <div class=\"row\">
      <div class=\"col-xs-9\">
        <ul class=\"nav nav-pills nav-stacked\">
          <li role=\"presentation\" class=\"active\">
            <a href=\"#\">Enlaces para CRUD</a>
          </li>
          ";
          $sql_tablas = "SHOW TABLES from ".$DbName."";
          $total_tables = AllDB(con(), $sql_tablas,2);
          foreach($total_tables as $value){
            foreach($value as $dato) {
              $find = array('_', '-', ' ');
              $TableName = strtolower(str_replace($find, "", $dat
              $varsalida1 .= "<li role=\"presentation\">
            <a href=\"view/".$TableName."/index.php\">".$TableNam
          </li>";
            }
          }
        $varsalida1 .= "
        </ul>
      </div>
    </div>
</div>
<hr>
<script type=\"text/javascript\" src=\"dist/js/jquery.min.js\">
<script type=\"text/javascript\" src=\"dist/js/bootstrap.min.js
</body>
</html>";
        echo "index.php creado correctamente<br>";
        fputs($gestor1,$varsalida1);
        fclose($gestor1);
    }
```

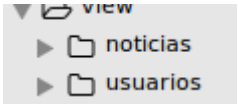
Ese foreach recorre sobre la base de datos e imprime los resultados sobre nuestro tema..



Este es el resultado. Se genera un enlace por cada tabla generada. Que nos lleva a un idex.php

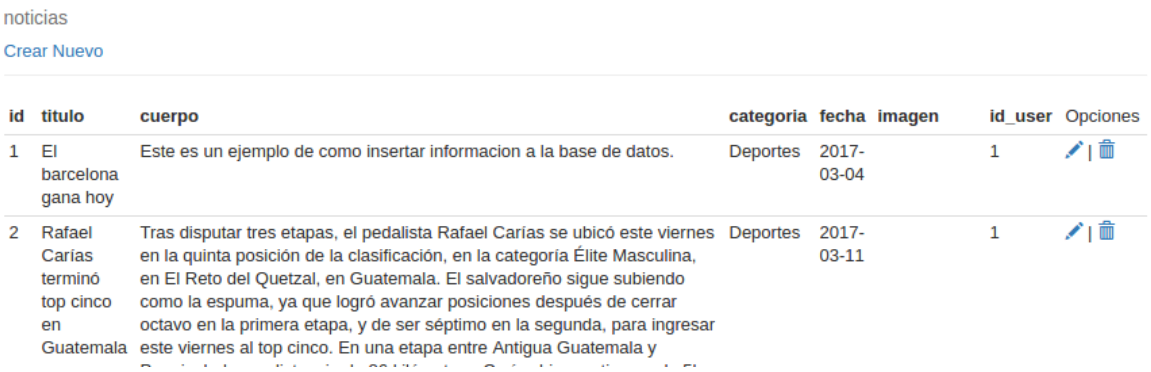
```
$view->createBaseView($_POST['txtName'],$_POST['txtbase']);
```

Esta instancia genera un archivo de ‘index.php’ que muestra los resultados de cada tabla. Esta ubicado en la carpeta de cada vista que también es generada en este instante. Se genera una por cada tabla.



```
$total_Colums = AllDB(con(), $sql_campos,2);
    foreach ($total_Colums as $datosC) {
    $varsalidal .="\t\t\t\t\t\t\t\t\t<th>".$datosC['Field']."</th>\n";
    }
  $varsalidal .="\t\t\t\t\t\t\t\t\t<td>Opciones</td>";
  $varsalidal .="\t\t\t\t\t\t\t\t</tr>
thead>
php
asql = \"SELECT * FROM ".$dato.";\";
bsql = Selects(connect(), \$asql);
reach (\$bsql as \$absql) {\n\t\t\t\t\t\t\tprint \"<tr>\";\n";
reach ($total_Colums as $datosC) {
$varsalidal .= "\t\t\t\t\t\t\tprint \"<td>\".\$absql['".$datosC['Field']."'].\"</
 td>\";\n";

lPrimaryKey = "SHOW INDEX FROM ".$dato." FROM ".$DbName." WHERE
Key_name='PRIMARY'";
otal_ColumsID = AllDB(con(), $sqlPrimaryKey,2);
reach ($total_ColumsID as $rowid) {
$aqui = $rowid['Column_name'];

$varsalidal .= "\t\t\t\t\t\t\tprint \"<td><a href='editar_".$TableName.".php?id=\"
 .\$absql['".$aqui."'].\"'><span class='glyphicon glyphicon-pencil'></span></
 a> | <a href='../../handlers/handlers_".$TableName."
 .php?action=delete&&id=\".\$absql['".$aqui."'].\"'><span class='glyphicon
 glyphicon-trash'></span></a></td>\";\n";
$varsalidal .= "\t\t\t\t\t\t\tprint \"</tr>\";\n\t\t\t\t\t\t} ?>
ble>
```

Al igual que los archivos anteriores, se deben de recorrer para mostrar los datos, en este ultimo caso se esta generando una tabla.



noticias

Crear Nuevo

id	titulo	cuerpo	categoria	fecha	imagen	id_user	Opciones
1	El barcelona gana hoy	Este es un ejemplo de como insertar informacion a la base de datos.	Deportes	2017-03-04		1	✎ | 🗑
2	Rafael Carías terminó top cinco en Guatemala	Tras disputar tres etapas, el pedalista Rafael Carías se ubicó este viernes en la quinta posición de la clasificación, en la categoría Élite Masculina, en El Reto del Quetzal, en Guatemala. El salvadoreño sigue subiendo como la espuma, ya que logró avanzar posiciones después de cerrar octavo en la primera etapa, y de ser séptimo en la segunda, para ingresar este viernes al top cinco. En una etapa entre Antigua Guatemala y	Deportes	2017-03-11		1	✎ | 🗑

```
$form->createBaseSave($_POST['txtName'],$_POST['txtbase']);
```

Esta instancia enviá dos parámetros que son el nombre del proyecto y de la base de datos y este los procesa en una función que genera un formulario de guardar. Uno por cada tabla.

No enviá los datos a ‘form_generate.php’.

form_generate.php

Este archivo contiene la clase ‘createBaseSave’ que recibe ambos parámetros y genera un archivo ‘save.php’ en la carpeta ‘view’ con su respectivo formulario.

Este archivo se genera de tal manera a los anteriores. Uno por tabla.


```
$handlers->createBaseHandlers($_POST['txtName'],$_POST['txtbase']);
?>
```

Esta clase crea un archivo llamados ‘manejadores’ que son los que reciben los datos enviados por el usuario y los conecta así los métodos y funciones anteriores.

El archivo encargado de esto es ‘handlers_generate.php’

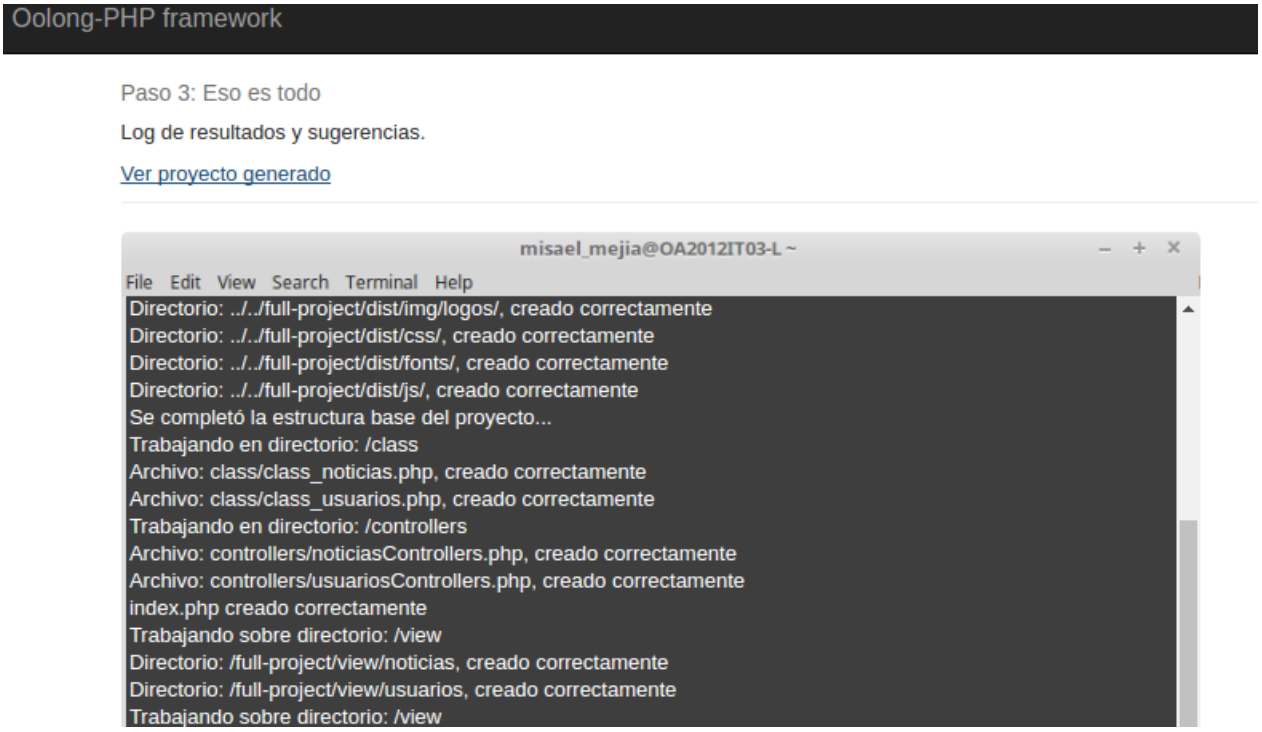
handlers_generate.php

Crea un archivo, uno por cada tabla en la carpeta ‘handlers’ del proyecto recien generado. Se encargan de ser intermediarios entre las vistas y las funciones.

```
//funcion para creacion de un manejador por cada tabla encargado de procesar cada
de CRUD
function createBaseHandlers($ProjectN,$DbName){
```

Paso 3:

Después de esa explicación de algunos de los procesos internos del código fuente del framework nos damos cuenta que estamos en la ventana final.



Y nos muestra una especie de terminal con información detallada de todo lo que el framework a generado. Y nos muestra un link hacia el proyecto ya generado.

Connection.php

Es el archivo de conexión de este framework por medio de el cual se conecta a mysql y se realizan todas las sentencias. Todas las instancias anteriores dependen de este archivo.

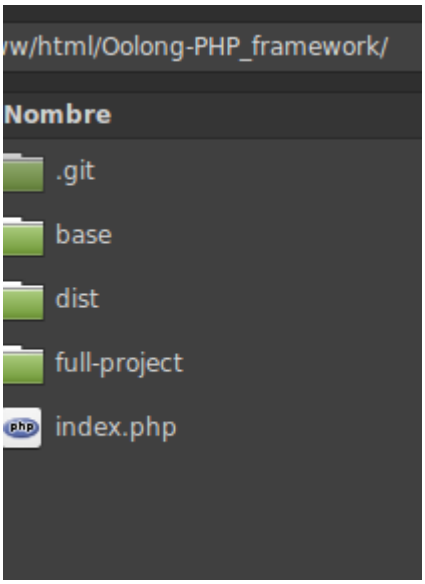
```php
<?php
include_once 'config.php';
function con(){
  $host = 'mysql:host='.HOST.'';
  $user = USUARIO;
  $password = CLAVE;
  try {
    $connection = new PDO($host,$user,$password);
    return $connection;
  }catch (PDOException $e) {
    $out ="";
    $out .= "<br>Número de error: ";
    $out .= $e->getCode();
    $out .= ". <br>información: ";
    $out .= $e->getMessage();
    print($out);
  }
}
function AllDB($conect, $sql,$type){
  if ($conect==NULL) {
    //Si la variable es null es porque ocurrio un
    hacemos nada
  }else{
    try {
      $datos = $conect->query(trim($sql));
      $datos->setFetchMode($type);
      $contenedor = $datos->fetchAll();
    }catch (Exception $exc) {
      return  "Error<br/>".$exc->getMessage();
    }
    return $contenedor;
  }
}
?>
```

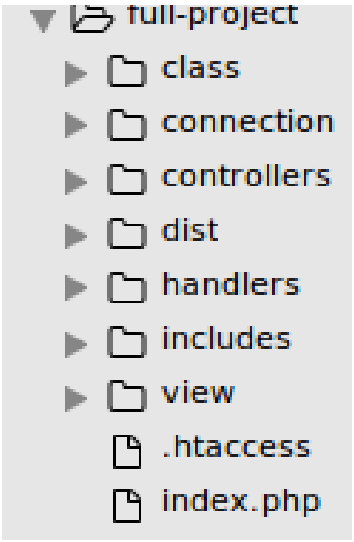
La función **con()** es la que devuelve una instancia de cadena de conexión mediante la clase de abstracción de datos pdo.

Y la función **AllDB()** Es la encargada de ejecutar consultas sobre la base de datos o sobre el servidor.

Copiando proyecto.

Como explicamos el proyecto se aloja en directorio raíz del framework en una carpeta llamada ‘full-project’. Esa carpeta la copiamos o cortamos y lo ejecutamos en el servidor php.





Si se pretende volver a utilizar el framework con datos diferentes de acceso a MYSQL debes modificar el siguiente archivo “base/connection/config.php”

Lo modificas o eliminarlo, como gustes. Solo debe de ser borrado este ‘config.php’

De lo contrario el fragmento no servirá.