	<p style="text-align: center;">UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN</p>
<p style="text-align: center;">CICLO I-2021</p>	<p style="text-align: center;">Lenguajes Interpretados en el Servidor Guía de práctica No. 12 AJAX y aplicaciones web con frameworks</p>

I. OBJETIVOS

- Adquiera habilidad en el proceso de aplicar AJAX dentro de una aplicación web.
- Haga uso de diferentes frameworks para trabajar en sus aplicaciones con AJAX.
- Realice aplicaciones avanzadas en las que use AJAX y acceso a bases de datos.
- Adquiera habilidad obtener, instalar y configurar un framework PHP como CodeIgniter.
- Realice una pequeña aplicación haciendo uso del framework CodeIgniter.

II. INTRODUCCIÓN TEÓRICA

Origen y definición de AJAX

En primer lugar, hay que decir que AJAX no es en sí mismo, una tecnología, y mucho menos, se trata de algo nuevo. AJAX solamente es un término que se ha adoptado y aceptado a nivel de desarrolladores web para hacer referencia a la utilización de un objeto de JavaScript, denominado XMLHttpRequest, para obtener información de un servidor web de modo dinámico o asíncrono. La posibilidad de hacer uso de este objeto es posible desde 1998. Microsoft incorporó este concepto desde Internet Explorer 4.0, con muchas dificultades de programación para los desarrolladores, y en Internet Explorer 5.0, ya con el objeto XMLHttpRequest, pero adoptando otro nombre. En ese entonces se le denominó Remote Scripting. Lastimosamente nadie le vio mucho potencial, hasta que Microsoft mostró su utilización con el Outlook Web Access incluido con Exchange. Ahora es mucho más popular con el desarrollo que ha tenido, principalmente en los servicios ofrecidos por Google en sus interfaces de usuario.

Inicialmente AJAX utilizó técnicas como marcos invisibles con elementos IFRAME para hacer uso de esta técnica, hoy en día se prefiere utilizar el objeto XMLHttpRequest.

La popularización de su utilización se debe principalmente a empresas como Google, Yahoo, Flickr, etc. Que han desarrollado diversas aplicaciones web en las que han utilizado el concepto AJAX para realizar aplicaciones web con excelentes servicios de interfaz de usuario.

Hoy en día la utilización de AJAX tiene todo el soporte necesario en los diversos navegadores web, como Firefox, Opera, Safari, Chrome, etc. Todos ellos ofrecen una versión clon del objeto XMLHttpRequest para que las aplicaciones AJAX funcionen sin inconvenientes.

Elementos de la definición de AJAX

Los elementos de la definición de AJAX son tres principalmente; sin embargo, en la práctica son varios más. Veamos cada uno de estos:

- **Asíncrono:** Significa que el usuario puede seguir interactuando con la página web mientras el navegador espera obtener la información solicitada al servidor desde la aplicación, sin que esto signifique que se tenga que recargar la página web.
- **JavaScript:** A pesar de que AJAX descansa sobre la utilización de diversas tecnologías, como: XHTML, CSS y DOM, además de JavaScript, se ha preferido tomar la letra inicial de esta última para que el nombre resulte más atractivo. La principal utilización de JavaScript en una aplicación AJAX consiste en el uso del objeto XMLHttpRequest y generación de contenido de forma dinámica.
- **XML:** Es uno de los tantos formatos de datos que pueden utilizarse para transferir contenido entre el servidor y la página web. Además, pueden utilizarse lenguajes estáticos como (X)HTML o lenguajes dinámicos como PHP, ASP o JSP. Además, de archivos de texto.
-

Tecnologías involucradas en la utilización de AJAX

En la práctica las tecnologías involucradas en el desarrollo de una aplicación web son:

- XHTML y CSS para la estructura de la página web y su presentación, respectivamente.
- El Modelo de Objetos de Documento (DOM) para la visualización y manipulación de las páginas.
- El objeto XMLHttpRequest de JavaScript para transferir datos entre el cliente y el servidor web.
- XML como formato para los datos que fluyen entre el cliente y el servidor. No obstante, también puede hacerse uso de texto normal para este propósito.
- JavaScript para mostrar e interactuar dinámicamente con todo lo anterior.

Una aplicación AJAX actúa como intermediario entre el usuario y el servidor. De modo que si la acción del usuario no requiere una llamada al servidor, el motor de AJAX se ejecuta asincrónicamente, para que el usuario siga interactuando con la aplicación. El motor de AJAX actualizará el sector de la página cuando los datos estén disponibles, muy probablemente, sin que el usuario lo note.

Aplicaciones web con AJAX y aplicaciones web sin AJAX

Una aplicación web que no utilice AJAX estará confinada a respetar 100% el modelo petición/respuesta del protocolo HTTP. En cambio, en una aplicación web con AJAX, la diferencia fundamental es que el usuario puede seguir interactuando con la página web, mientras se procesa en el servidor de forma asíncrona. Las siguientes figuras ilustran este hecho:



¿Qué es un framework?

Conceptualmente, framework es un conjunto estandarizado de conceptos, prácticas y criterios utilizados para normalizar un tipo de problemática particular que sirve para enfrentar y resolver nuevos problemas de índole similar. Ahora bien, en el desarrollo de software o de una aplicación informática un framework constituye un esquema o un patrón conceptual y tecnológico definido con módulos de software concretos, con base en el cual puede organizarse otro proyecto de software más complejo. El propósito del framework es contribuir al rápido desarrollo de las aplicaciones, permitiendo tiempos de desarrollo significativamente más cortos, en comparación al tiempo que requeriría desarrollarlas sin un framework.

Modelo Vista-Controlador

El Modelo Vista Controlador es un patrón de programación que se fundamenta en la separación de la lógica del negocio del aspecto visual. Toda aplicación desarrollada bajo este esquema estará estructurada en tres capas: la capa de datos, la capa de interfaz y la capa lógica.

- Modelo: que procesa u obtiene datos, permitiendo gestionar la entrada y salida de la información almacenada en una base de datos.
- Vista: invocada desde el controlador y que representa la forma en que los datos son presentados en pantalla. En las aplicaciones para la web, es la encargada de mostrar las páginas html.
- Controlador: controla qué sucede en la aplicación, gestionando las peticiones, obteniendo los datos solicitados de un modelo, los procesa y se los envía a una vista para que puedan ser mostrados de forma adecuada.

Razones para utilizar el patrón MVC

La razón de mayor peso para utilizar este patrón de programación es la facilidad para el mantenimiento del código en el futuro, ya que se encuentran separadas las secuencias de comando en modelo, vista y controlador, de modo que se hace fácil la localización de alguna funcionalidad, a la hora de desarrollar. Además de esto, se puede decir que la velocidad para

el desarrollo de las aplicaciones es otra de las razones de peso para utilizar un framework MVC.

Frameworks de PHP

Existen numerosos frameworks de PHP, entre los que se pueden mencionar Zend Framework, Cake, Symfony, CodeIgniter, Akelos, Prado, ZooP, etc. Cada uno proporciona características diversas. Es difícil, sugerir cuál puede ser el mejor de todos, habría que conocer cada una. A la hora de decidir cuál utilizar, debe influir el conocimiento de las características que proporciona cada una. La siguiente imagen muestra un cuadro comparativo entre algunos de los Frameworks más conocidos y populares

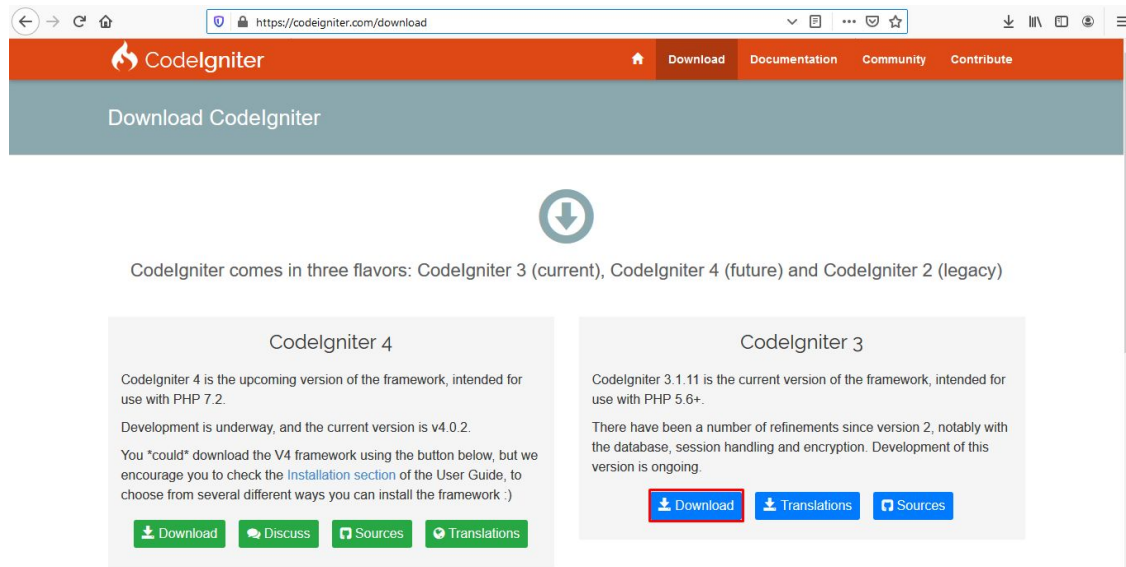
El framework CodeIgniter.

Una de las primeras cosas que debe saber antes de comenzar a utilizar un framework, es tener, al menos, un dominio medio, si no es que amplio, del lenguaje de programación en el que está hecho. En este caso, es preciso tener un nivel de conocimiento intermedio-avanzado de PHP. Además, debe poseer un conocimiento amplio sobre programación orientada a objetos, ya que el Framework CodeIgniter (y la gran mayoría de Frameworks PHP) está construido sobre este paradigma de programación. Por último, debe poseer alguna noción de lo que es el Modelo Vista Controlador, debido a que este y otros frameworks están diseñados con este patrón de desarrollo de software. Dicho lo anterior, comencemos a explorar las características de CodeIgniter:

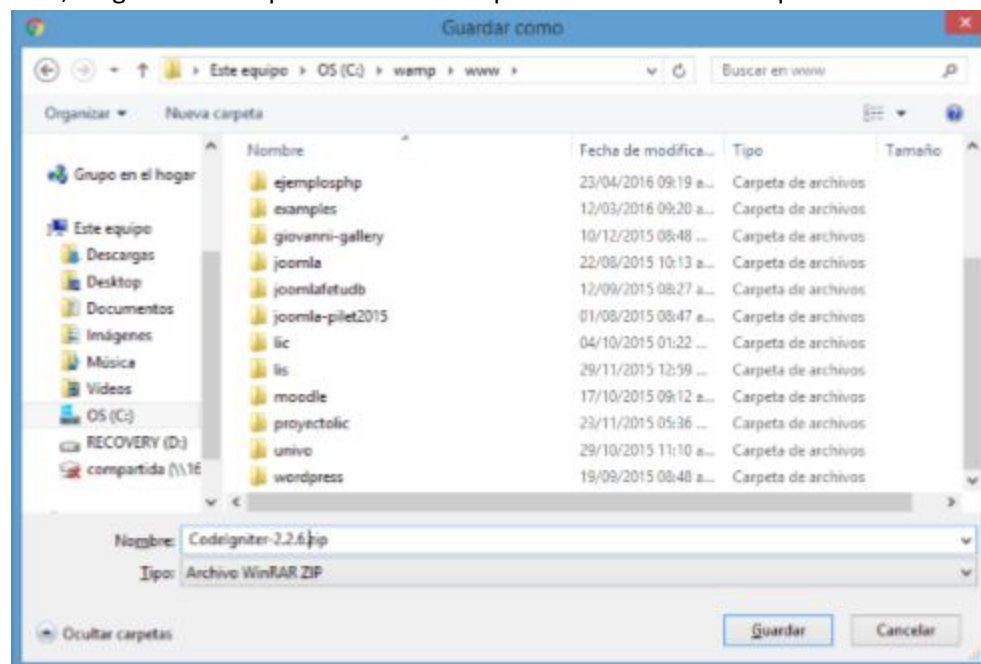
- Utiliza el patrón de desarrollo MVC (Modelo Vista Controlador).
- Posee clases para manejo de bases de datos de varias plataformas.
- URLs amigables basadas en segmentos.
- Librerías para validación de datos de formularios.
- Librerías para seguridad y filtros XSS. Soporte para manejo de sesiones.
- Envío de e-mails.

Instalación de CodeIgniter.

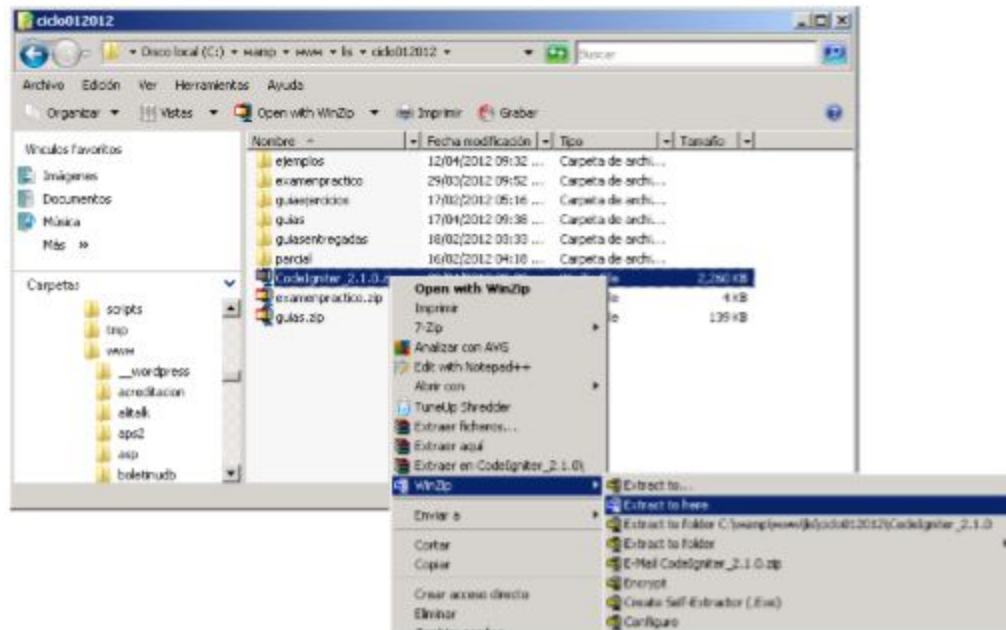
La descarga es simple, hay que dirigirse al sitio oficial del proyecto CodeIgniter y buscar el enlace de descargas: <https://codeigniter.com/download>. En esa página encontrarás la última versión de CodeIgniter. El archivo descargado está comprimido. Lo único que hay que hacer es descomprimirlo y moverlo a la carpeta web del servidor. Si lo desea puede moverlo a una carpeta personalizada, siempre dentro de la carpeta web



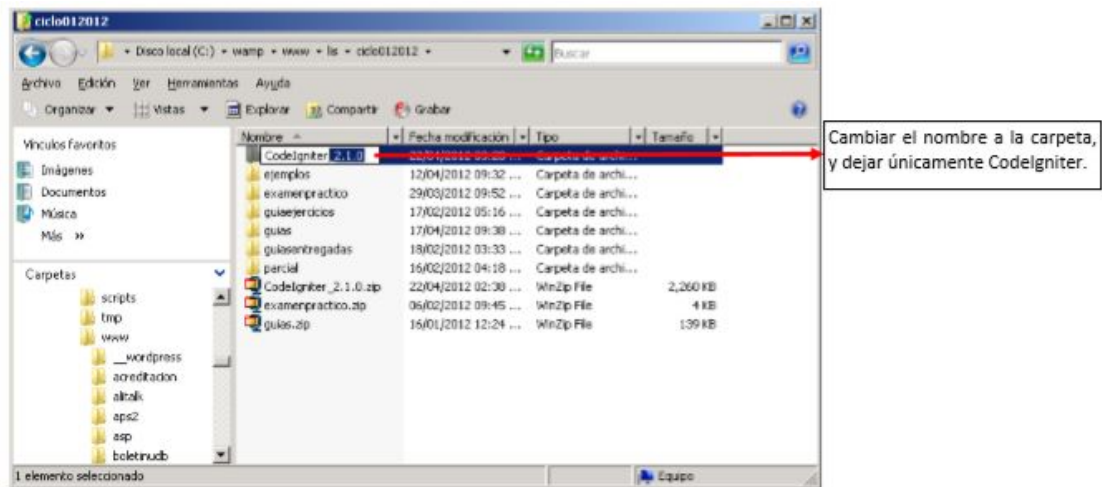
Al dar clic en el botón Download CodeIgniter 2 inmediatamente se abrirá el cuadro de diálogo Guardar como. Puede guardar el archivo comprimido en cualquier parte del disco en este momento, luego al descomprimir los archivos puede moverlos a la carpeta web.



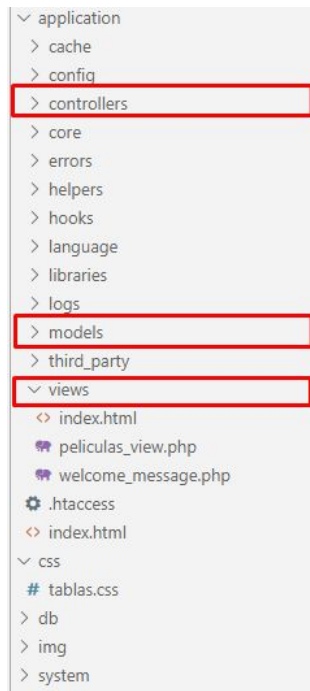
Mueva el archivo comprimido a la carpeta web o a una carpeta dentro de esta, en donde residirán los archivos del CodeIgniter.



Si lo desea cambie el nombre de la carpeta que se crea al extraer los archivos para comodidad al ingresar la URL en la barra de dirección del navegador.



Si observamos, la versión 3 de CodeIgniter muestra tres carpetas y dos archivos, como se muestra a continuación



Estos directorios forman la base de lo que se conoce como Modelo Vista-Controlador (MVC), como sus nombres indican, cada uno contiene archivos relacionados con tareas específicas que dividen las aplicaciones realizadas en tres capas.

Modelos, controladores y vistas.

En la carpeta models se almacenará todo el código que tiene que ver con el acceso a la base de datos, manteniendo encapsulada toda la complejidad de la lógica del negocio. En esta capa se crearán clases o funciones para acceder, insertar, actualizar o borrar información de las tablas de la base de datos. La idea es que desde otras partes de la aplicación, sólo se invoquen a los métodos de las clases de nuestro modelo, de modo que sin importar toda la lógica y complejidad que haya en dicho modelo, si se solicitan datos, estos siempre serán obtenidos.

La carpeta controllers debe contener los scripts que manejarán la lógica de la aplicación auxiliándose de los modelos para obtener datos, si son solicitados o cargando las vistas con dichos datos para mostrárselos al usuario en el navegador. Los controladores sirven de enlace entre los modelos, las vistas y cualquier otro recurso que se tenga que procesar en el servidor para generar la página web.

Por último, la carpeta views contiene los scripts que constituyen las páginas web con las que interactúa el usuario. En otras palabras, en las vistas estará principalmente, todo el código (X)HTML, CSS y JavaScript que se tiene que generar para producir la página web que deseamos sea vista por el usuario

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Material	Cantidad
1	Guía de práctica #12: AJAX y aplicaciones web con frameworks	1
2	Computadora con WampServer instalado y funcionando correctamente	1
3	Editor PHP sublime Texto Eclipse PHP	1
4	Memoria USB o disco flexible	1

IV. PROCEDIMIENTO

Digite los scripts que se enumeran a continuación y guárdalos en el formato apropiado como páginas .html o como scripts .php según se indique.

Ejemplo #1:

El siguiente ejemplo muestra cómo crear un campo de texto de formulario con funcionalidad de auto sugerencia para llenar el campo mientras se ingresan caracteres en dicho campo. Las cadenas sugeridas son tomadas de una tabla de una base de datos en base a coincidencias con los caracteres ingresados usando una sentencia SELECT que se actualiza constantemente cada vez que se ingresa un nuevo carácter en la caja de texto.

Script 1: index.php

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="stylesheet" href="css/style.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.j
s"></script>
</head>
<body>
<div class="header">
  <h1>Autocompletado</h1>
  <p>Utilizando MYSQL y AJAX</p>
</div>

<p>Ingrese nombre de país:</p>

<!--Make sure the form has the autocomplete function switched off:-->
<form autocomplete="off" >
  <div class="autocomplete" style="width:300px;">
```



```

        <input id="myInput" type="text" name="myCountry"
placeholder="Country">
    </div>
</form>

<script src="js/script.js"></script>

</body>
</html>

```

Script 2: ajax_refresh.php

```

<?php
    function connect() {
        return new
PDO('mysql:host=localhost;dbname=autocompletedb;charset=utf8', 'root',
'', array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)
    );
    }

    $pdo = connect();
    $result_array = array();
    $sql = "SELECT * FROM country
        ORDER BY country_id ASC";
    $query = $pdo->prepare($sql);
    $query->execute();
    $list = $query->fetchAll();

    foreach ($list as $rs){

        array_push($result_array, $rs);

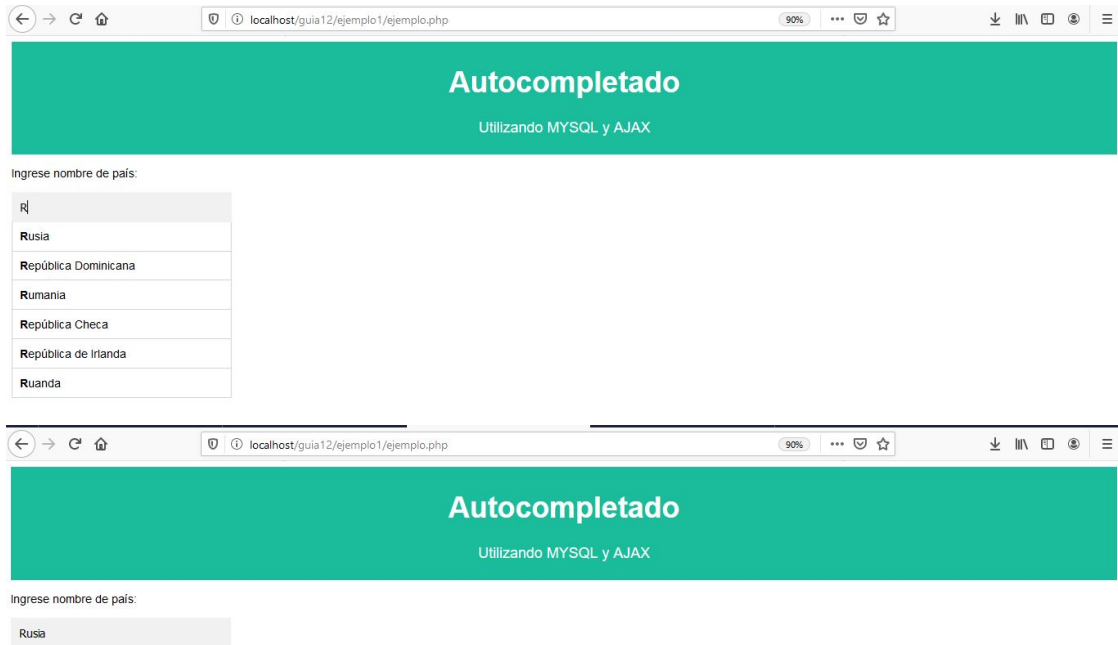
    }

    header('Content-type: application/json');
    echo json_encode($result_array);

?>

```

Al ejecutar el script index.php en el navegador podrá observar lo siguiente:



Ejemplo #2: El siguiente ejemplo muestra una aplicación que utiliza pestañas para mostrar tres tipos de contenido, uno en cada pestaña. La información que se muestra en cada pestaña se carga directamente de una base de datos usando AJAX.

Script 1: tabs.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta http-equiv="Content-type" content="text/html; charset=utf-8"
/>
  <title>Ventana con pestañas usando AJAX</title>
  <script src="js/ahahLib.js"></script>
  <script src="js/activetab.js"></script>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.mi
n.css"
integrity="sha384-Gn5384xqQ1aowXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGg
FAW/dAiS6JXm" crossorigin="anonymous">
</head>
<body class="container">
<section>
<article>
<ul id="tabmenu" class="nav nav-tabs">
  <li id="t1" class="nav-item" onclick="makeactive(1)" >
    <a class="nav-link active" id="tab1">Noticias</a>
  </li>
  <li id="t2" class="nav-item" onclick="makeactive(2)">
    <a class="nav-link" id="tab2">Sociales</a>
```

```

        </li>
        <li id="t3" class="nav-item" onclick="makeactive(3)">
            <a class="nav-link" id="tab3">Deportes</a>
        </li>
    </ul>
    <div id="content">
        (Aquí se mostrará el contenido de cada pestaña)
    </div>
</article>
</section>
</body>
</html>

```

Script 2: content.php

```

<?php
    $host = "localhost";
    $user = "root"; //Cambiar por el usuario de la base de datos en su
servidor
    $pass = ""; //Cambiar por la contraseña de la base de datos en su
servidor
    $db = "prensa";
    //Establecer conexión con el servidor MySQL
    $cn = new mysqli($host, $user, $pass, $db);
    if($cn->connect_errno){
        printf("Falló la conexión: %s\n", $cn->connect_error);
        exit(0);
    }

    $tipo = isset($_GET['content']) ? $_GET['content'] : 0;
    if($tipo != 0){
        $qr = "SELECT titulonoticia, textonoticia, imgnoticia FROM
noticia ";
        $qr .= "WHERE idnota = $tipo";
        $rs = $cn->query($qr);

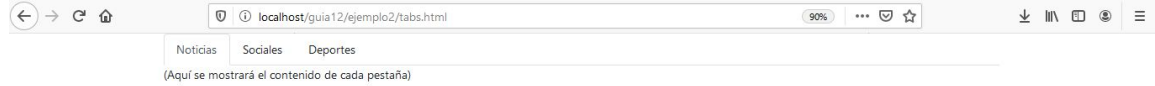
        while($row = $rs->fetch_object()){
            $info = "<div
id=\"titulo\">\n<h2>{$row->titulonoticia}</h2>\n</div>\n";
            $info .= "<div id=\"texto\">\n<p>\n{$row->textonoticia}\n";
            $info .= "<img src=\"{$row->imgnoticia}\n\" />\n";
            $info .= "</p>\n</div>\n";
        }

        if (!$cn->set_charset("utf8")) {
            printf("Error cambiando el juego de caracteres utf8: %s\n",
$cn->error);
        } else {

```

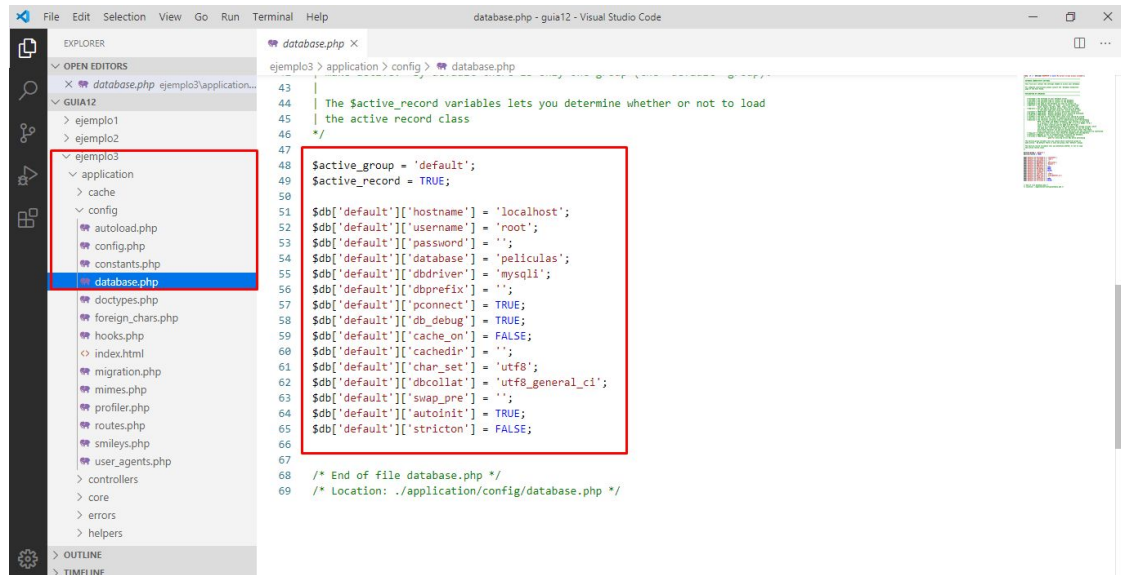
```
        printf($info, $cn->character_set_name());  
    }  
}  
?>
```

Al visualizar el ejemplo en el navegador de su preferencia, obtendremos lo siguiente:



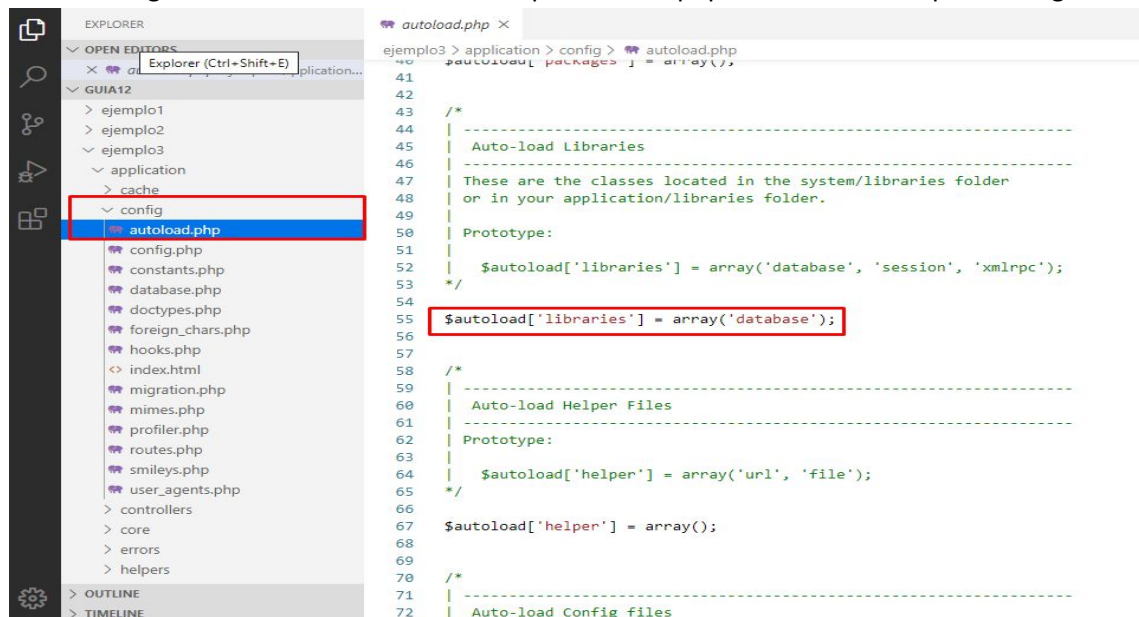
Ejercicio #3: En este ejemplo se realizará una pequeña aplicación para interactuar con bases de datos desde una aplicación realizada con el framework CodeIgniter.

PASO 1: Configurar el archivo database.php almacenado en la carpeta config.



```
43 |
44 | The $active_record variables lets you determine whether or not to load
45 | the active record class
46 | */
47 |
48 | $active_group = 'default';
49 | $active_record = TRUE;
50 |
51 | $db['default']['hostname'] = 'localhost';
52 | $db['default']['username'] = 'root';
53 | $db['default']['password'] = '';
54 | $db['default']['database'] = 'películas';
55 | $db['default']['dbdriver'] = 'mysql';
56 | $db['default']['dbprefix'] = '';
57 | $db['default']['pconnect'] = TRUE;
58 | $db['default']['db_debug'] = TRUE;
59 | $db['default']['cache_on'] = FALSE;
60 | $db['default']['cachedir'] = '';
61 | $db['default']['char_set'] = 'utf8';
62 | $db['default']['dbcollat'] = 'utf8_general_ci';
63 | $db['default']['swap_pre'] = '';
64 | $db['default']['autoinit'] = TRUE;
65 | $db['default']['stricton'] = FALSE;
66 |
67 |
68 | /* End of file database.php */
69 | /* Location: ./application/config/database.php */
```

PASO 2: Cargar la librería database en el script autoload.php ubicado en la carpeta config.



```
41 | $autoload['packages'] = array();
42 |
43 |
44 | /*
45 |  Auto-load Libraries
46 |  These are the classes located in the system/libraries folder
47 |  or in your application/libraries folder.
48 |
49 |  Prototype:
50 |
51 |  $autoload['libraries'] = array('database', 'session', 'xmlrpc');
52 |  */
53 |
54 | $autoload['libraries'] = array('database');
55 |
56 |
57 | /*
58 |  Auto-load Helper Files
59 |  Prototype:
60 |
61 |  $autoload['helper'] = array('url', 'file');
62 |  */
63 |
64 | $autoload['helper'] = array();
65 |
66 |
67 | $autoload['helper'] = array();
68 |
69 |
70 | /*
71 |  Auto-load Config files
72 |  */
```

PASO 3: Crear el modelo para acceder a los datos de la base de datos películas. El modelo debe almacenarse en la carpeta models ubicada dentro de la carpeta applications con el nombre películas_model.php

```
<?php
class películas_model extends CI_Model{
    function getAll(){
        //Construir la consulta a ejecutar en el servidor MySQL
    }
}
```

```

        //Esta consulta involucrará datos de columnas de las tres
        //tablas (pelicula, genero y director) por lo tanto se hará
        //uso de la cláusula JOIN para recuperar los datos de las
        //tres columnas que son requeridas
        /*$sql = "SELECT titulopelicula, descripcion, imgpelicula,
nombre, generopelicula, duracion ";
        $sql .= "FROM pelicula ";
        $sql .= "JOIN genero ON pelicula.idgenero = genero.idgenero
";
        $sql .= "JOIN director ON pelicula.iddirector =
director.iddirector ";
        //Ejecutar la consulta
        $qr = $this->db->query($sql);*/

        $qr = $this->db->select('titulopelicula, generopelicula,
descripcion, imgpelicula, nombre, duracion')
            ->from('pelicula')
            ->join('genero', 'pelicula.idgenero = genero.idgenero')
            ->join('director', 'pelicula.iddirector =
director.iddirector')
            ->get();

        //Recorrer el conjunto de resultados con una sentencia
        //repetitiva y almacenar las filas o registros recuperados
        //en la matriz $data que se devolverá al controlador
        if($qr->num_rows() > 0){
            foreach($qr->result() as $row){
                $data[] = $row;
            }
            return $data;
        }

        function newMovie(){
            $sql = "";
        }
    }
?>

```

PASO 4: Crear el controlador peliculas.php que gestionará las peticiones del usuario que acceda al sitio web. Este controlador se debe almacenar en la carpeta controllers.

```

<?php
class peliculas extends CI_Controller{
    function index(){
        //Invocar al modelo que realizará la consulta en la tabla
        //peliculas
        $this->load->model('peliculas_model');
    }
}

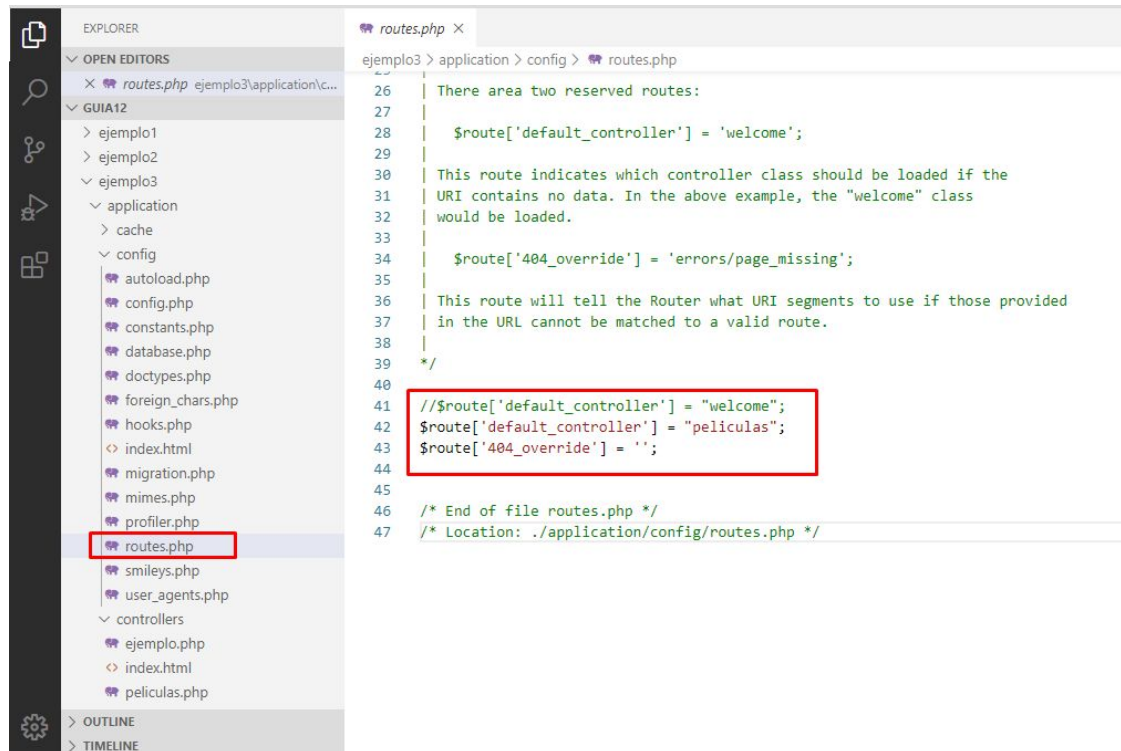
```

```

        //Los datos que sean obtenidos se cargarán en la matriz
        //asociativa $data con índice 'rows'
        $data['rows'] = $this->peliculas_model->getAll();
        //Cargar la vista y pasarle los datos devueltos por el modelo
        $this->load->view('peliculas_view', $data);
    }
}
?>

```

PASO 5: Definir como controlador por defecto peliculas en el archivo routes.php. Busque dentro de este archivo las líneas siguientes:



Debe notar que se ha comentado la línea que define el controlador por defecto y se ha agregado una nueva donde se indica el nuevo controlador por defecto.

PASO 6: Por último crear la vista, que será la página web que visualizarán los usuarios en sus navegadores y con la que podrán interactuar. Este script debe almacenarse en la carpeta views con el nombre peliculas_view.php.

```

<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Base de datos de películas</title>
    <link rel="stylesheet" href="<?==PATH;?>css/tablas.css" />
</head>

```

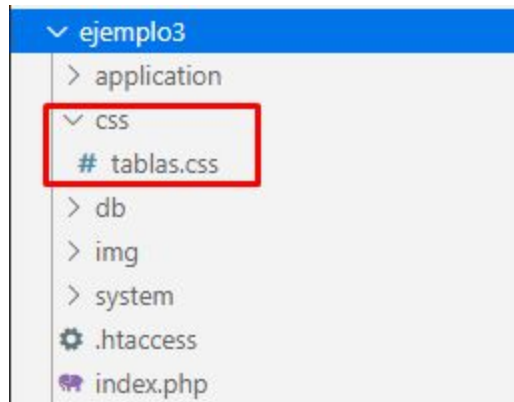
```

<body>
<section>
<article>
<div id="container">
  <?php
    $tabla = "<table class=\"tablaps\">\n";
    $tabla .=
    "<thead>\n<tr>\n<t<t<th>\n<t\t\t\t\tPOSTER\n<t\t\t\t\t</th>\n<t\t\t\t";
    $tabla .= "<th>\n<t\t\t\t\tCARACTERISTICAS\n<t\t\t\t\t</th>\n<t\t\t\t";
    $contador = 1;
    foreach($rows as $movie):
      if($contador % 2 == 1) $clase = "impar";
      else $clase = "par";
      $tabla .= "<tr class=\"\" . $clase . "\">\n<t\t\t\t";
      $tabla .= "<td>\n<t\t\t\t\t<img src=\"\" . PATH .
$movie->imgpelicula . "\" alt=\"\" . $movie->titulopelicula . "\"
/>\n<t\t\t\t</td>\n<t\t\t\t";
      $tabla .= "<td class='caracteristicas'>\n<t\t\t\t\t <h1
class='titulo'>" . $movie->titulopelicula . "<h1><br>";
      $tabla .= "\n<t\t\t\t\t SINOPSIS:" . $movie->descripcion .
"\n<t\t\t\t\t<br>";
      $tabla .= "\n<t\t\t\t\t DIRECTOR:" . $movie->nombre .
"\n<t\t\t\t\t<br>";
      $tabla .= "\n<t\t\t\t\t GENERO:" . $movie->generopelicula .
"\n<t\t\t\t\t<br>";
      $tabla .= "\n<t\t\t\t\t DURACIÓN:" . $movie->duracion . "
min\t\t\t\t</td>\n<t\t\t\t\t</tr>\n";
      $contador++;
    endforeach;
    $tabla .= "</tbody>\n";
    $tabla .= "</table>\n";
    echo $tabla;
  ?>
</div>
</article>
</section>
</body>
</html>

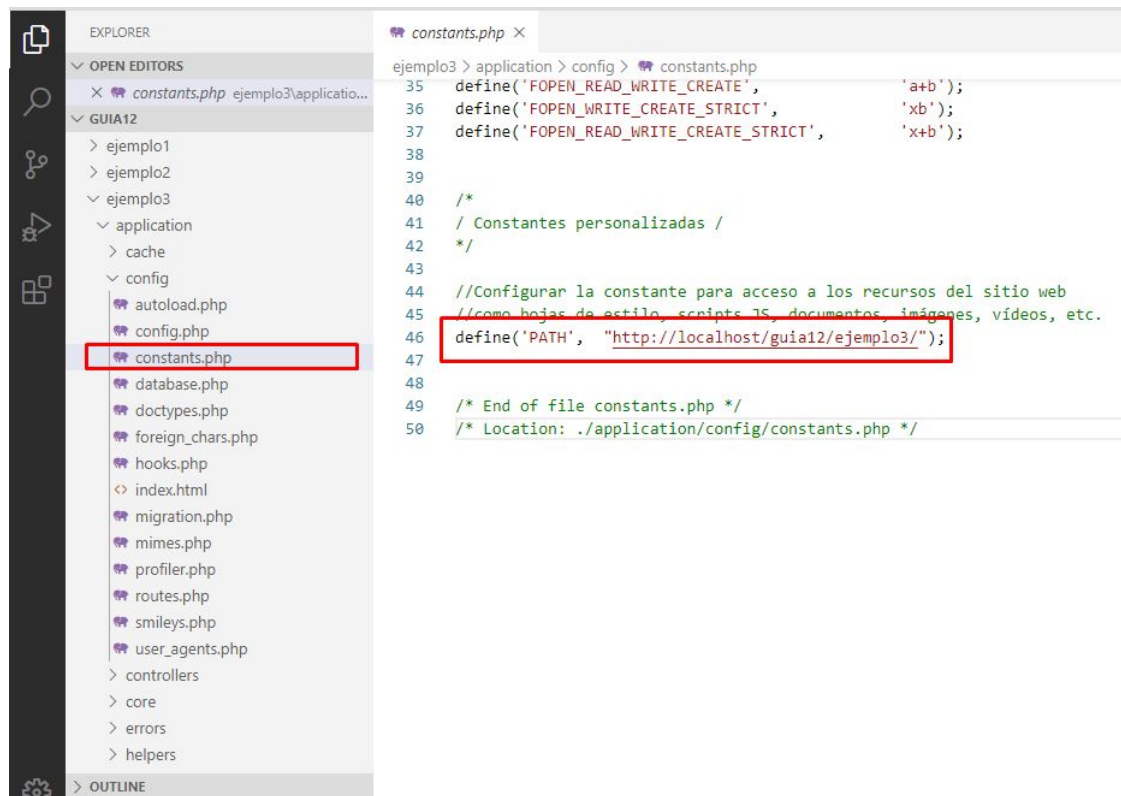
```

NOTA: Las carpetas de los recursos css e img deben copiarse a la carpeta de instalación del CodeIgniter, en tanto que la base de datos, que también viene con los recursos debe crearse con el nombre películas. Además, debe modificar el archivo .htaccess para que pueda accederse correctamente a la carpeta img/ desde el script de la vista, de lo contrario no será rastreable esa carpeta y no podrá visualizar las imágenes.

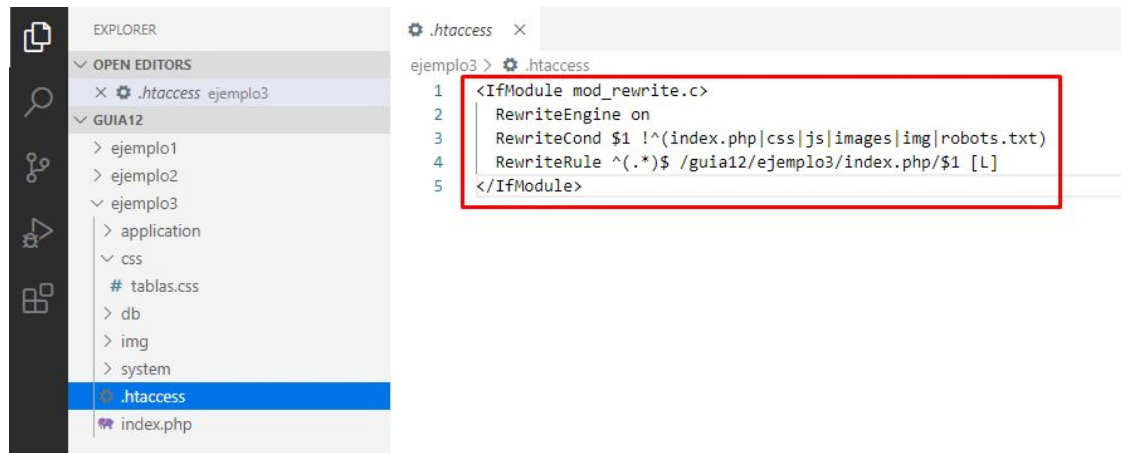
PASO 7: Agregar las hoja de estilos proporcionada en los recursos en la carpeta css/ que estará dentro de la carpeta de instalación de CodeIgniter.



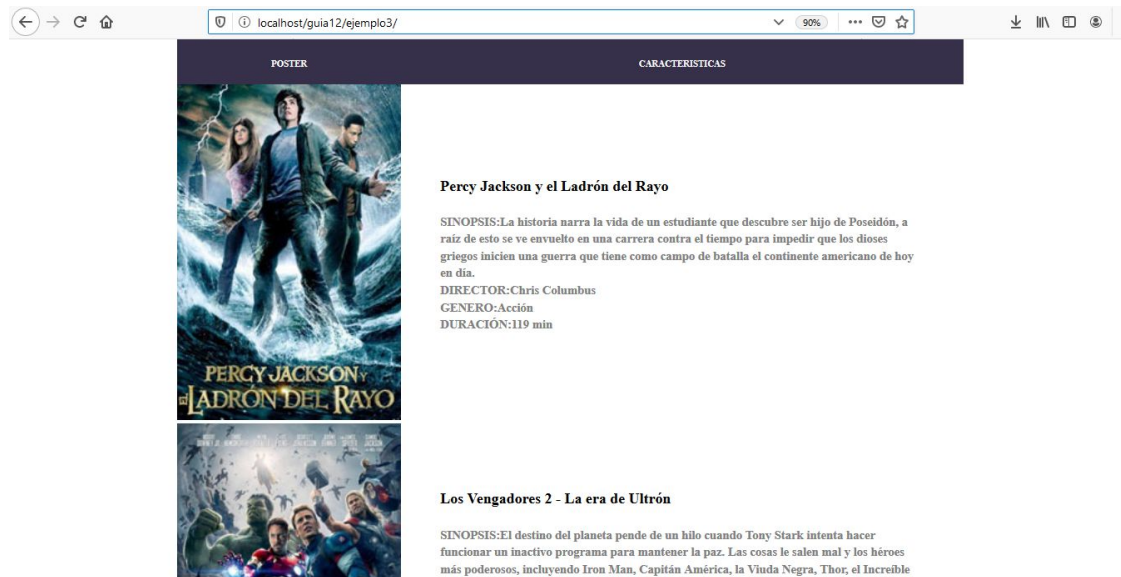
PASO 8: Configurar la constante PATH para acceder a la hoja de estilo y posiblemente a otros recursos que se agreguen en el sitio web como imágenes, scripts de JavaScript o de jQuery, documentos, etc. En este caso necesita abrir el archivo constants.php, también ubicado en la carpeta config dentro de application y agregue la línea que se muestra a continuación en dicho archivo.



PASO 9: Agregar una excepción más de redireccionamiento en el archivo .htaccess.



PASO 10: Cargue en el navegador el sitio web que ha realizado con la siguiente URL (<http://localhost/guia12/ejemplo3/>). Debería observar lo siguiente:



V. BIBLIOGRAFÍA

- Gutiérrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. Editorial Alfaomega RAMA. 1ra edición. México. Junio 2005.
- Gil Rubio, Francisco Javier/Villaverde, Santiago Alonso/Tejedor Cerbel, Jorge A. Creación de sitios web con PHP 5. Editorial McGraw-Hill. 1ra edición. Madrid, España, 2006.
- John Coggeshall. La Biblia de PHP 5. 1ra Edición. Editorial Anaya Multimedia. Madrid España.
- <http://www.php.net/manual/en>