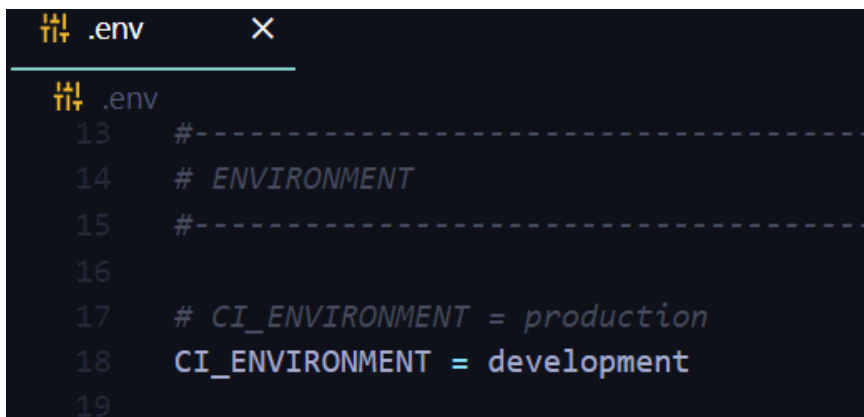


PASOS PARA TRABAJAR CON CODEIGNITER

PASOS INICIALES

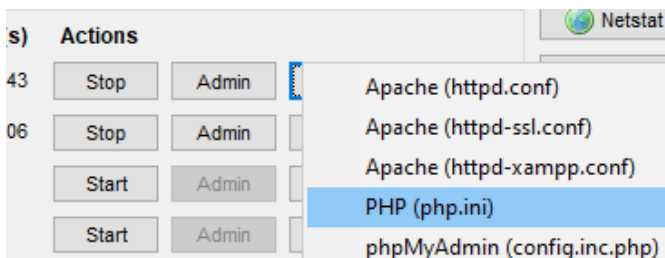
1. Descargar e instalar de la página oficial <https://codeigniter.com/download>
2. Tener instalado Xampp
3. En VSC instalar plugins como Bootstrap snippet
4. Instalar plugin CodeIgniter Spark en VSC
5. Se descomprime el proyecto y se ubica en la carpeta raíz de Htdocs

Para iniciar es posible que haya que hacer una modificación en el archivo .env. Se copia la línea comentada como # CI_ENVIROMENT y se le quita el numeral a la copia, y donde dice production cambiamos por development,



```
.env
13 #-----
14 # ENVIRONMENT
15 #-----
16
17 # CI_ENVIROMENT = production
18 CI_ENVIROMENT = development
19
```

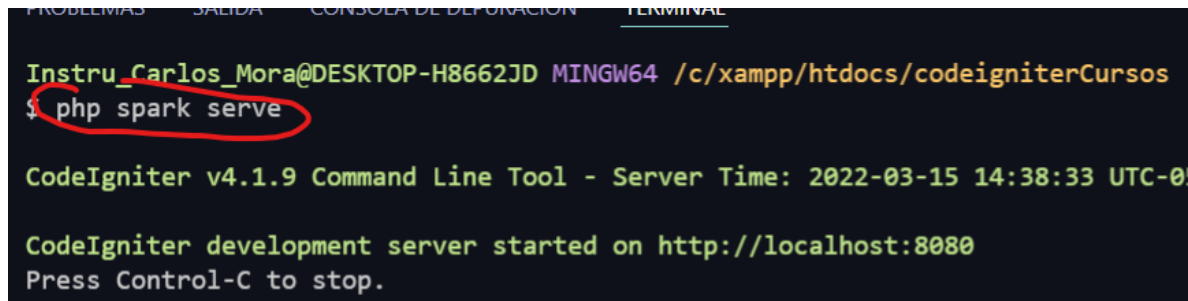
Si sigue arrojando algún error, se debe hacer una configuración en el archivo php.ini. Así que en el Panel de Control de Xampp, para el servicio de Apache, clickeamos en config:



En ese archivo se hace el siguiente cambio, se quita el punto y coma (;) que está antes de la línea extensión=intl

```
extension=fileinfo
;extension=gd
extension=gettext
;extension=gmp
extension=intl
;extension=imap
;extension=ldap
extension=mbstring
```

No olvidar que se debe iniciar el servidor en la ventana de comandos, con php spark serve:

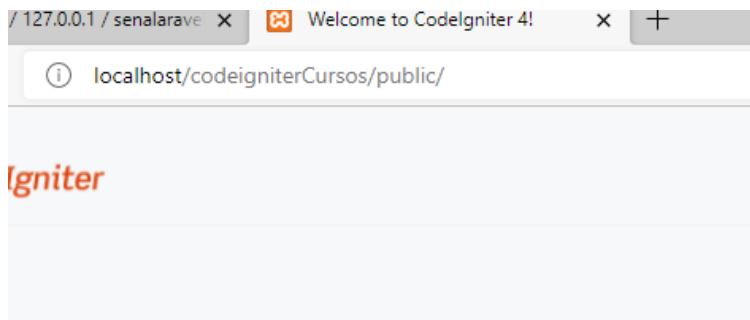


```
Instru_Carlos_Mora@DESKTOP-H8662JD MINGW64 /c/xampp/htdocs/codeigniterCursos
$ php spark serve

CodeIgniter v4.1.9 Command Line Tool - Server Time: 2022-03-15 14:38:33 UTC-0

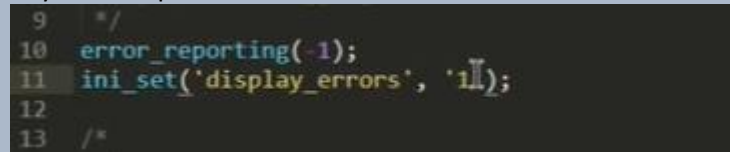
CodeIgniter development server started on http://localhost:8080
Press Control-C to stop.
```

Se visualiza en el navegador indicando la ruta donde está el proyecto y la ruta public. Ejemplo:



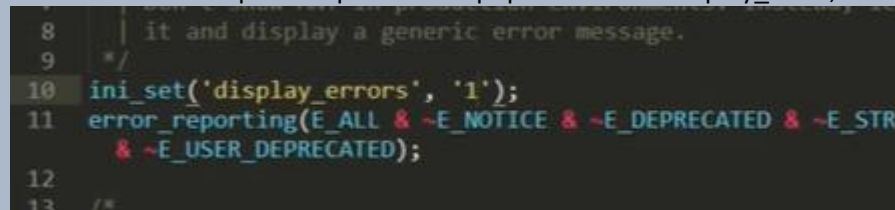
Las siguientes configuraciones son opcionales para resolver el problema de que el Framework no inicia en el navegador

y también en app/config/Boot/development.php, en la siguiente captura, en display_errors, si hay un cero poner un 1.



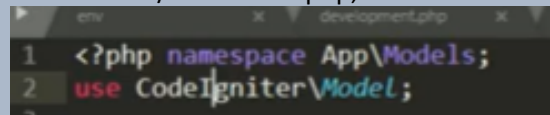
```
9  /*
10 error_reporting(-1);
11 ini_set('display_errors', '1');
12
13 /*
```

Y en la misma ruta pero en production.php también en display_error, si hay un cero poner un 1



```
8  | it and display a generic error message.
9  /*
10 ini_set('display_errors', '1');
11 error_reporting(E_ALL & ~E_NOTICE & ~E_DEPRECATED & ~E_STRICT
    & ~E_USER_DEPRECATED);
12
13 /*
```

Y en Models/userModel.php, cambiamos en el use, el Codeigniter por CodeIgniter



```
1  <?php namespace App\Models;
2  use CodeIgniter\Model;
3
```

CONFIGURACION DE BASE DE DATOS

1. En App/Config/Database.php se hace la siguiente configuración (en este caso nuestra bd se llamará Instituto).

```
public $default = [  
    'DSN'       => '',  
    'hostname'  => 'localhost',  
    'username'  => 'root',  
    'password'  => '',  
    'database'  => 'Instituto',  
    'DBDriver'  => 'MySQLi',  
    'DBPrefix'  => '',  
];
```

2. En App/Config/App.php cambiamos la dirección de la baseURL:

```
* @var string  
*/  
public $baseURL = 'http://localhost/codeigniterCursos/public/';
```

3. Usando la interfaz de phpMyAdmin, creamos la bd Cursos manualmente:







Nombre de la tabla: Agregar columna(s)

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	Índice	A. Comentarios
id	INT		Ninguno			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>
nombre	VARCHAR	100	Ninguno			<input type="checkbox"/>	---	<input type="checkbox"/>
imagen	VARCHAR	255	Ninguno			<input type="checkbox"/>	---	<input type="checkbox"/>

Estructura

4. Ingresamos datos manualmente en la BD. La ruta de la imagen la inventamos, solo con fines de modificación:

+ Opciones

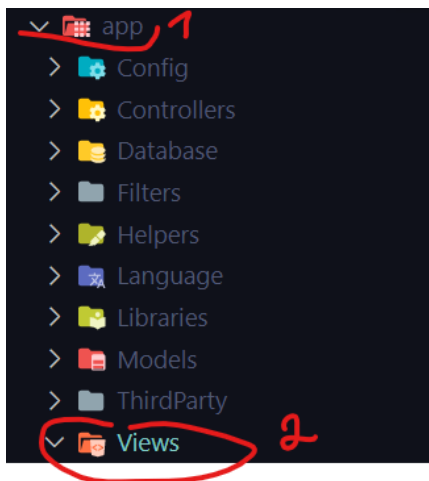
				id	nombre	imagen
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1	Aprendiendo CodeIgniter	imagen.jpg
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	2	PHP Avanzado	imagen2.jpg

El query ejecutado es algo como:

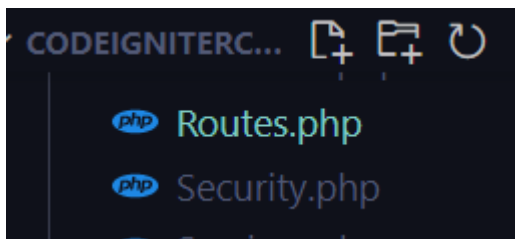
```
INSERT INTO `cursos` (`id`, `nombre`, `imagen`) VALUES (NULL, 'PHP Avanzado', 'imagen2.jpg');
```

UBICACIÓN DE LAS VISTAS Y LAS RUTAS

En **CodeIgniter** las vistas están dentro de la misma carpeta **app**, y luego en **Views**:



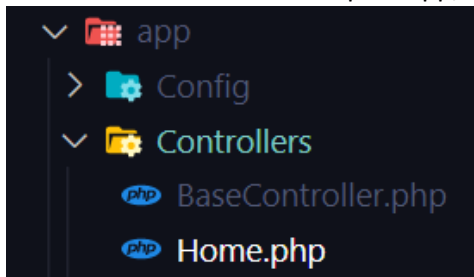
Por otra parte, las rutas las encontramos también en **app**, dentro de la subcarpeta **Config**. Allí está el archivo **Routes.php**



```
// route since we don't have to scan directories.  
$routes->get('/', 'Home::index');  
  
/*  
* -----  
*/
```

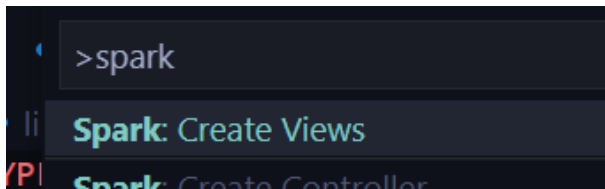
LOS CONTROLADORES

Están directamente en la carpeta app, en Controllers. Por default, allí está Home.php

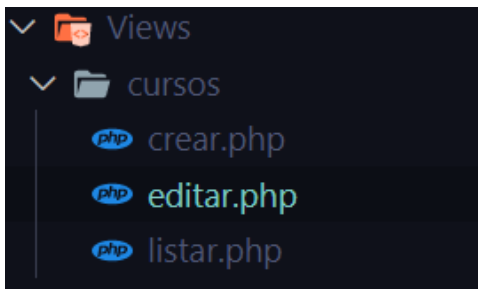


Cómo crear una vista usando la extensión Spark de VSC

Para acceder a las funcionalidades del plugin spark, presionamos F1 (o Fn F1), y en la barra de búsqueda escribimos spark. Luego damos el nombre de la vista y finalmente decimos que deseamos incluir la estructura HTML (clic en yes):

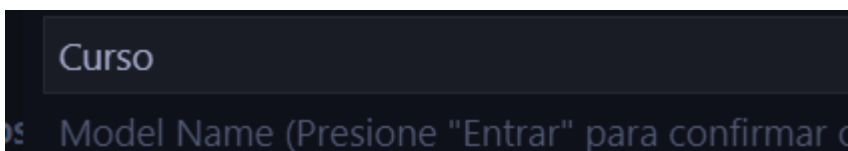
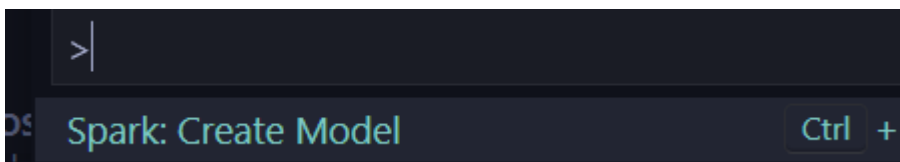


1. Estas son las vistas creadas, las cuales se incluyen en una subcarpeta Cursos:

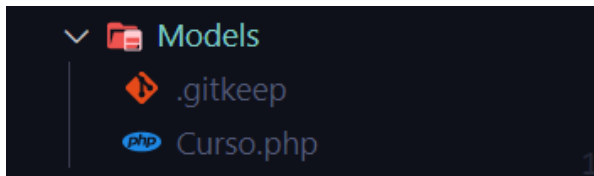


Creando un Modelo y un Controlador

Es necesario crear un modelo (Curso) el cual permitirá acceder a la tabla que debe especificarse con su nombre (cursos)



Hay que decirle cómo se llama la tabla con la que conectará y si se creará una migración. Como no estamos desde la ventana de comandos, no es necesario crear una migración. Además, cuando pregunta si deseamos crear el controlador, le decimos que sí y se llamará Cursos.



Creación de la ruta con Spark

Creamos la respectiva ruta ingresando los datos que nos pida. Simplemente oprimimos F1, luego spark create rout, y seguimos los pasos.

Para Personalizar La vista Listas

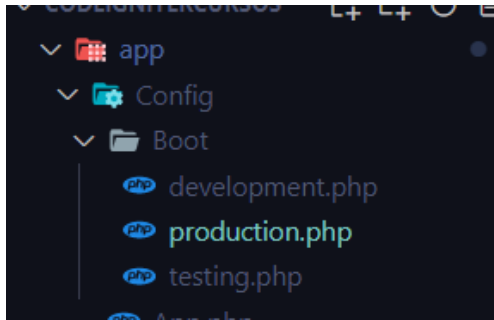
Incluimos Bootstrap.

Creamos una tabla (b-table header, usando la extensión de Bootstrap para VSC):

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist
</head>
<body>
  <h2>Listado de cursos</h2>
  <div class="container">
    <table class="table table-dark">
      <thead class="thead-light">
        <tr>
          <th>#</th>
          <th>Imagen</th>
          <th>Nombre</th>
          <th>Acciones</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td></td>
        </tr>
      </tbody>
    </table>
  </div>
</body>
</html>
```

CONFIGURANDO PARA DEPURAR ERRORES (O VER LOS ERRORES)

1. Ir a app/config/Boot/production.php



2. Cambiar 0 por 1.

CONSULTAR LA BASE DE DATOS

1. Lo primero en el modelo Curso (el único creado hasta ahora) hacemos lo siguiente:

```
class Curso extends Model{
    protected $table = 'cursos';
    // Uncomment below if you want add primary key
    protected $primaryKey = 'id';
    //activar el acceso a las columnas
    protected $allowedFields = ['nombre','imagen'];
}
```

2. Luego se debe llamar la información desde el controlador (en el ejemplo, Cursos), en el método index():

```
public function index(){
    //se crea el objeto de clase Curso
    $curso = new Curso();
    /*se crea un array e indico que se orden la información por id y en forma ascendente
    y que se busquen todos los registros. Todo se va a depositar en $datos[*/
    $datos['cursos'] = $curso->orderBy('id', 'ASC')->findAll();
    return view('cursos/listar',$datos);
}
```

3. El índice del array \$datos, que en este ejemplo lo llamamos 'cursos', y que está en los corchetes, lo debemos tratar como una variable dentro de la vista listar.php. Por tanto lo llamamos en la vista como \$cursos, y así podremos mostrar la información de la tabla, en la vista:

```
y que se busquen todos
$datos['cursos'] = $cur
return view('cursos/lis
```

```
listar.php M X
app > Views > cursos > listar.php > html > body > div.co
8 </head>
9 <body>
10 <div class="container">
11 <?php
12 print_r($cursos);
13 ?>
14 <h2>Listado de cursos</h2>
15 <br>
16 <table class="table table-dark">
```

Como puede apreciarse, se utiliza el método `print_r` para hacer una comprobación y ver que funciona en el navegador. Los usuarios finales no deberían ver ese tipo de información.

4. Ya en la vista se puede apreciar la información de la tabla respectiva:

```
Array ( [0] => Array ( [id] => 1 [nombre] => Aprendiendo CodeIgniter [imagen] => imagen.jpg ) [1] => Array ( [id] => 2 [nombre] => PHP Avanzado [imagen] => imagen2.jpg ) )
```

Listado de cursos

5. Para verlo ya dentro de la tabla de estilo Bootstrap como tal, comentamos la línea anterior de `print_r` y digitamos lo siguiente dentro de `tr`:

```
<tr>Acciones</tr>
</tr>
</thead>
<tbody>
<?php foreach ($cursos as $curso): ?>
<tr>
<td><?=$curso['id'];?></td>
<td><?=$curso['imagen'];?></td>
<td><?=$curso['nombre'];?></td>
<td>Editar/Borrar</td>
</tr>

<?php endforeach; ?>
</tbody>
```


CREACIÓN DE TEMPLATES

1. Se crea la carpeta templates en Views
2. Dento se crea cabecera.php y piepagina.php
3. El siguiente código se corta (Ctrl X) de listar.php y se pega en cabecera.php

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-sc
6     <title>Document Title</title>
7     <link rel="stylesheet" href="https://cdn.jsdelivrivr.net/npm/boo
8 </head>
9 <body>
10 <div class="container">
```

4. El siguiente código se corta (Ctrl X) de listar.php y se pega en piepagina.php

```
app > Views > template > php piepagina.php
1 </div>
2 </body>
3 </html>
```

5. Luego en el controlador, dentro del método index() se llama a la cabecera y al pie de página creados así:

```
$datos['cursos'] = $curso->orderBy('id', 'ASC')->findAll();
/*Llamo a la cabecera y al pie de página desde el controlador y asignándolo también a datos
recordar que $datos es un array pero almacenamos en sus diferentes
posiciones indexadas con los nombres cursos, cabecera y pie*/
$datos['cabecera'] = view('template/cabecera');
$datos['pie'] = view('template/piepagina');
return view('cursos/listar',$datos);
}
```

6. Luego usamos lo anterior directamente en la vista listar.php de la siguiente manera

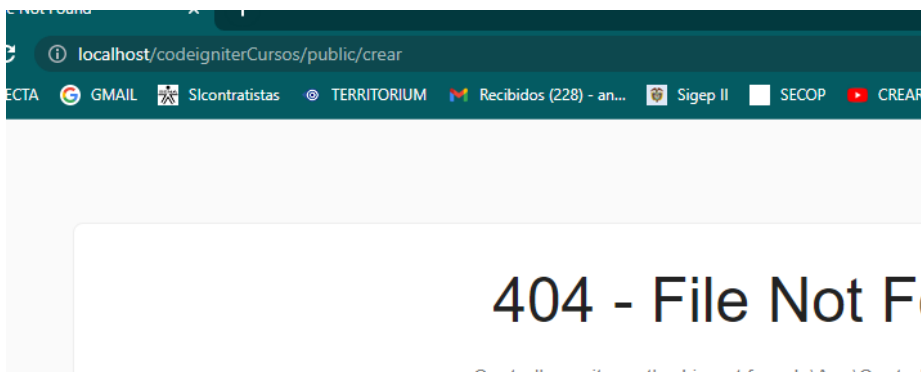
```
listar.php x
app > views > cursos > listar.php > ...
1 <?=$cabecera; //esto equivale a escribir e
2
3 <?php
4 // print_r($cursos);
5 ?>
6 <h2>Listado de cursos</h2>
7 <br>
8 <table class="table table-dark">
...
</tbody>
</table>
<?=$pie; //imprimimos el pie ?>
```

CREANDO ENLACES (LINKS) A OTRAS VISTAS

1. Creamos un link en la página Listar.php, que lleva a la vista crear.php

```
?>
<a href="<?=$base_url('crear') ?>">Crear un Libro</a>
<h2>Listado de cursos</h2>
<br>
<table class="table table-dark">
  <thead class="thead-light">
```

2. Luego probamos



3. Abrimos el archivo Routes.php que está en Boot.
4. Oprimimos F1 para poder usar los diferentes comandos
5. Luego escribimos spark
6. Elegimos Create Route
El nombre sería crear, el controlador sería Cursos. El método sería crear

```
php listar.php M      php Routes.php M X
app > Config > php Routes.php > ...
53 | $routes->get('crear', 'Cursos::crear');
```

MODIFICANDO EL FORMULARIO DE CREAR Y AGREGANDO EL FORMULARIO

1. Se incluye primero cabecera y pie en el controlador, en el método crear()

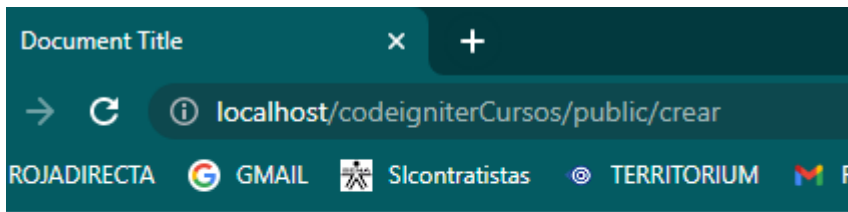
```
php Cursos.php M X
controllers > php Cursos.php > Cursos > crear
}

public function crear(){
    $datos['cabecera'] = view('template/cabecera');
    $datos['pie'] = view('template/piepagina');
    return view('cursos/crear',$datos);
}
```

2. Se borra lo que había en la vista crear y se agregan cabecera y pie:

```
php crear.php M X
app > Views > cursos > php crear.php
1 | <?=$cabecera; ?>
2 |
3 | Formulario para crear
4 |
5 | <?=$pie; ?>
```

3. El resultado es:



Formulario para crear

4. Se agrega el código para el formulario en crear.php:

```
<form method="POST" action="" enctype="multipart/form-data">
  <br>
  <div class="form-group">
    <label for="nombre">Nombre:</label>
    <input id="nombre" class="form-control" type="text" name="nombre">
  </div>
  <div class="form-group">
    <label for="imagen">Imagen:</label>
    <input id="imagen" class="form-control-file" type="file" name="imagen">
  </div>
  <button class="btn btn-primary" type="submit">Guardar</button>
</form>
<?=$pie; ?>
```

ENVÍO LA INFORMACIÓN POR POST

¿Cómo envío y almaceno la información del formulario?

1. Se debe indicar en el action a través de site_url y la ruta guardar. Se debe crear dicha ruta.

```
<form method="POST" action="<?=site_url('/guardar') ?>" enctype="multipart/form-data">
  <br>
```

2. Se abre el archivo de Routes.php. La ruta se crea con el nombre guardar, asociada al controlador Cursos y al método guardar que luego debemos crearlo también.

