Informe Trabajo Final, Proyecto de Software 2016.

Framework elegido: CodeIgniter
Grupo: 64

Fundamentación sobre la elección del framework CodeIgniter

Elegimos el framework Codeigniter debido a la estructura y manejo de MVC que utiliza y por el flujo de control de la aplicación que implementa Codeigniter, ya que era muy parecido a la implementación de nuestro proyecto durante la cursada de la materia. De esta forma la migración al sistema era mucho más simple y rápida que otros frameworks y la curva de aprendizaje de CodeIgniter era mucho más corta debido a sus características.

Justificación

El sistema creado durante la cursada poseía la siguiente estructura de carpetas:

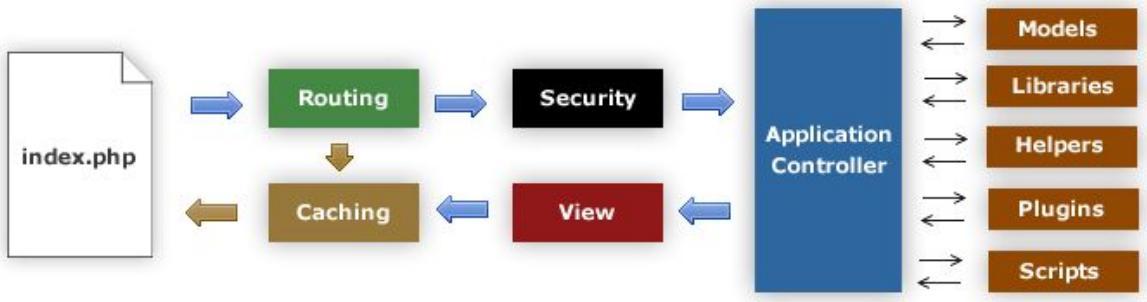
- Controller : En esta carpeta se alojan los controladores, por convención, los archivos debían terminar en *Controller.php
- Model : En esta carpeta se alojan los modelos, por estándar,por convención, los archivos debían terminar en *Model.php
- Template : En esta carpeta se alojan las vistas, divididas en dos subcarpetas. Una para el frontend del sistema y otra para el backend.

Las url eran resueltas por el archivo Loader.php mediante el pasaje de dos parámetros por get, controller y method. Loader.php verifica que el usuario que realizó la petición poseyera permisos suficiente para poder ejecutar la operación, y que existiera el controlador y el método solicitado por get.

Ejemplo: Inicio

Luego el controlador se encargaba de solicitar la información necesaria al modelo y de invocar a la vista correspondiente.

Manejo de MVC dentro de CodeIgniter

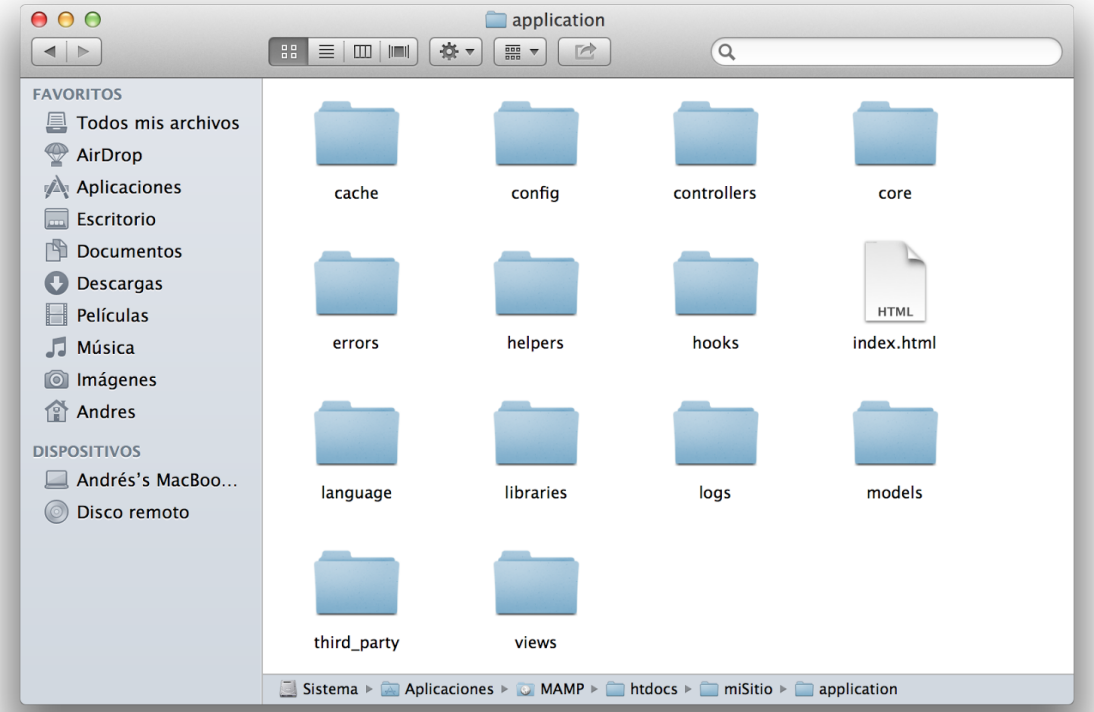


1. El archivo index.php sirve como controlador de frontend, inicializa los recursos base para que pueda funcionar CodeIgniter
2. El Router examina la petición HTTP que debe hacerse.
3. Si existe una caché, se envía directamente al navegador, salteando la ejecución normal del sistema.
4. Seguridad. Antes de que se cargue el controlador de la aplicación, la petición HTTP y cualquier información enviada por el usuario es filtrada por seguridad.
5. El controlador carga el modelo, las librerías principales y cualquier otro recurso necesario para procesar la petición.
6. La vista final es renderizada y enviada al navegador.

Estructura MVC de Codeigniter

- **Application**: Se encuentran todos los archivos de nuestro proyecto y es donde trabajaremos.
- **System**: Aquí se encuentran el core del Framework.

Carpeta Application:



Las 3 básicas serían el **modelo**, la **vista** y el **controlador**, igual que en nuestro trabajo durante la cursada, además las siguientes:

- **Helpers**: Aquí van librerías particulares, las cuales no se encuentran escritos en formato orientado a objetos y cada función realiza una tarea específica. Como por ejemplo: Cookie Helper que crea y recupera cookies de manera sencilla.
- **Hooks**: En esta carpeta se puede alterar el comportamiento que tiene normalmente el core del framework sin modificar los archivos los contenidos en la carpeta system.
- **Libraries**: En esta carpeta se almacenan todas las librerías que ayudan a manejar alguna funcionalidad compleja dentro del sistema. Como por ejemplo: Twig y el manejo de sesiones.
- **Third_party**: Aloja las librerías de terceros para extender la funcionalidad de PHP. Se considera librerías de terceros a todas aquellas librerías ajenas a

codeigniter o nuestra propia aplicación. Aquí se albergarán las librerías para la creación de gráficos y pdfs.

- **Config**: Se encuentran los archivos de configuración de codeigniter.

El mecanismo previsto para el manejo de seguridad y routing

El mecanismo de seguridad es similar al utilizado en el trabajo durante la cursada, en lugar de que el archivo Loader.php se encargue de ver si un usuario posee permisos estos serán verificados en la llamada de cada método mediante una función en el Controlador general, a su vez también se hace un control de csrf en los formularios del sistema, también dentro del controlador general. En cuanto al sistema de ruteo, se utiliza el provisto por defecto por el framework que consiste en URLs de la forma example.com/clase/funcion.
Es posible configurar la forma de manejar las URLs para que posean un formato similar al realizado durante la cursada quedando las url de la siguiente manera: index.php?controller=controller&method=method&

Para eso se debe modificar las siguientes variables en el archivo de configuración de codeIgniter:

$config['enable_query_strings'] = FALSE; (Ignora el formato de CodeIgniter y muestra el pasaje de parametros por GET)
$config['controller_trigger'] = 'controller'; (Nombre que tendrá la variable del controlador a invocar)
$config['function_trigger'] = 'method'; (Nombre que tendrá la variable del método a invocar en base a la variable que contiene al controlador)
De esta forma el archivo Loader.php es obsoleto debido a que codeigniter realiza está funcionalidad de forma automática.

Controladores

Todo controlador en CodeIgniter debe extender de la clase CI_Controller. En nuestro proyecto todos los controladores extienden de un controlador general llamado Controller.php, por lo cual simplemente bastó con hacer que dicho controlador extiende de CI_Controller.

Modelo

Todo modelo en CodeIgniter debe extender de la clase CI_Model. En nuestro proyecto todos los modelos extienden de un modelo general llamado Model.php, por lo que simplemente bastó con hacer que dicho modelo extienda de CI_Model.

invocación del modelo

Para poder hacer una invocación al modelo dentro de los controladores dentro de codeIgniter se deben cargar utilizando la función
$this->load->model('nombre_modelo');
Esta función se encarga de instanciar una nueva variable nombre_modelo para poder invocar al modelo. En nuestro sistema se instanciable una variable $nombre_modelo = new nombre_modelo; Por lo que los cambios realizados para que el controlador y el modelo pudieran comunicarse fueron Mínimos.

CRUD en CodeIgniter

En nuestro sistema cada Modelo en particular llamaba a una función del modelo general que se encargaba de preparar la consulta PDO a la base de datos.
La estructura era la siguiente:

```
public function getUser($username, $pass)
{
 return $this -> queryPreparadaSQL('
    SELECT *
    FROM usuario
    WHERE usuario = :username
```

```
    AND clave = :pass',
    array('username' => $username, 'pass' => $pass ));
}
```

Los métodos en el modelo en codeigniter usan una versión modificada del Patrón de Base de Datos Active Record . Un beneficio mayor de usar este patrón es que permite crear una aplicación independiente de la base de datos que usa, ya que la sintaxis de consulta es genera**da por cada adaptador de base de datos. También permite consultas más seguras, y**a que los valores son escapados automáticamente por el sistema. Al modificar nuestros métodos para que se integren al patrón utilizado por CodeIgniter queda formada la siguiente estructura:

```
public function getUser($username, $pass)
{
  $this->db->select('*');
  $this->db->from('usuario');
  $this->db->where('usuario', $username);
  $this->db->where('clave', $pass);
  return $this->db->get()->result();
}
```

Vistas

Para las vistas se utilizó la siguiente librería de codeIgniter : https://github.com/kenjis/codeigniter-ss-twig

Para ello se creó una función dentro del controlador padre addData(clave, valor), que agrega información a un arreglo que luego será pasado a la librería de Twig. Para renderizar la vista adecuada se utiliza la función $this->twig->render(vista, $datos);

Como se puede observar los cambios realizados para migrar el sistema al framework CodeIgniter no son muy complicados, el mayor trabajo de migración se realizó sobre el modelo donde había que cambiar la estructura de las consultas para

que se adapten al patrón Active Record. Cabe mencionar que CodeIgniter permite realizar consultas de la misma forma en que realizamos antes en el sistem. llamando a la función $this->db->query($sql, $arreglos_variables). De esta forma el esfuerzo realizado para migrar el sistema sería prácticamente mínima, Pero se optó utilizar la estructura de Active Record por los beneficios mencionados anteriormente.

¿Por qué usar CodeIgniter?

- "Obliga" a trabajar con el modelo MVC
- "Incentiva" a trabajar con programación orientada a objetos (En CI, los controladores y los modelos en realidad son clases por default, que se instancia usando la clase Loader que trae el CI.
- La documentación del framework es extensa, hay mucha información y libros en la red sobre CI.
- La curva de aprendizaje es muy rápida, por la sencillez del framework
- Es muy flexible, ya que no obliga a tener una determinada estructura de tablas, nombres de campos, ni adherirse a una forma de programar concreta como con otros frameworks.
- Trae integrada ya una clase para hacer pruebas de unidad.
- Es muy ligero y compatible, funciona en casi cualquier hosting, y no pide ningún requisito extranormal y se instala en un par de minutos.
- Las aplicaciones que se hacen con CI, son independientes del motor de BD que

se utilice; usando la clase ActiveRecord que trae se puede migrar de motor de base de datos sin cambiar nada del código de la aplicación.

La descripción de los módulos desarrollados en el trabajo de la cursada que pudieron ser aprovechados para ser usados por el framework

Se aprovecharon todos los módulos realizados durante la cursada, solamente se realizaron los cambios mínimos explicados anteriormente

Bibliografía

- Guía de usuario de CodeIgniter: https://www.codeigniter.com/user_guide
- Tutoriales de DigitalOcean: https://www.digitalocean.com/community/tutorials/codeigniter-getting-started-with-a-simple-example