Proyecto de Software.

Trabajo final sobre el sistema de Buffet basado en Laravel.

UNLP
Facultad de Informática

Integrantes:
Cristian Gallardo
Emiliano Retamar
Felipe Arana

Sobre la elección del framework elegido.

La elección se realizó en base a varios factores:

- Tiempo que tuvimos de desarrollo.
- Dificultad de aprendizaje.
- Documentación sobre el tema.
- Conocimientos entre los integrantes del grupo.

Dentro del marco de frameworks de php, decidimos elegir Laravel. Estuvo en su momento la discusión por implementarlo en Codeigniter, cuya curva de aprendizaje parecía ser leve y también respeta MVC, pero optamos por algo más complejo. Entre los 3 integrantes tuvimos la elección de elegir entre Symfony y Laravel, debido a la cantidad de documentación por ser unos de los frameworks más utilizados. Al final, optamos por Laravel por las siguientes razones:

- Más documentación con respecto a otros frameworks.
- Utiliza un sistema de autenticación listo para usar, ahorrando tiempo de desarrollo.
- La herramienta artisan posee diferentes herramientas en cuanto a la construcción de los objetos a la base de datos, y un simple mecanismo para montar el sistema en el servidor local.

Tenemos los conocimientos básicos de cómo construir una aplicación y en la web existe mucho código implementado que podemos reutilizar para nuestros proyectos, ahora entonces sabiendo esto ¿por qué utilizamos un framework?

El framework dispone de componentes que simplifican la tarea de programación y reducen enormemente nuestro trabajo. Esta herramienta nos permite evitar perder tiempo en la lógica básica del proyecto que ya deberíamos saber y así poder concentrar más tiempo en los requerimientos de la aplicación. Por otra parte estos componentes son probados por una amplia comunidad alrededor del mundo, por lo cual nos “aseguramos” de que este código desarrollado por una cierta cantidad de persona nos simplificarán nuestra.

La seguridad siempre es un punto crítico. Estos frameworks ¿nos proveen mecanismos de seguridad necesaria? La respuesta es sí, siempre y cuando se realicen las validaciones de datos y filtros de los mismos. Las consultas a la base de datos están automatizadas y validadas por ORMs, en nuestro caso Eloquent, el cual nos evita en gran parte este gran problema de seguridad.

Laravel al igual que otros frameworks, provee un esqueleto en el cual poner nuestro código. Estas mismas facilitan la colaboración, desarrollo de software entre usuarios ya que están estandarizados.

Módulos aprovechados para ser usados por el framework.

La versión actual del sistema brinda los siguientes servicios:

- Stock de productos.
- Compras de productos a proveedores.
- Ventas a clientes.
- Pedidos online para clientes dentro de la facultad de informática.
- Configuración del sitio

En cuanto a los módulos brindados, pudimos utilizar las herramientas brindadas por Laravel para simplificar las siguientes implementaciones:

- Registro de usuario: Laravel trae por defecto una clase que se encarga del registro de los usuarios del sistema, en donde el programador sólo necesita configurar dicha clase con los datos necesarios para el alta de un usuario y las validaciones correspondientes, sobreescribiendo métodos que nos provee el framework.

- Login: Así como el registro, el login de usuarios también viene implementado, y se configura de la misma forma que el anterior, solo es necesario setear los valores correspondientes y decirle a Laravel con que campo de la base de datos va a comparar, ya sea email o username. También en la misma clase se pueden declarar variables para configurar hacia donde se debe redirigir en caso de que el usuario sea autenticado correctamente y hacia donde en caso de que sea erróneamente.

- Mecanismo de ruteo: Laravel provee un mecanismo muy simple para las rutas, estas se definen en un archivo llamado “routes.php” donde se declaran todas las rutas del sistema y el método HTTP empleado para acceder a esa ruta. Por ejemplo una ruta llamada “login” accedida con mediante “GET” no es la misma que una ruta llamada “login” accedida mediante “POST”. Pueden tener el mismo nombre pero ambas se diferencian y pueden dirigirse a un controlador o método diferente. En la declaración de las rutas se debe especificar el controlador y el método a ejecutar para una determinada ruta. También es posible aplicar una especie de filtros denominados “middlewares” los cuales se ejecutan antes del llamado al método especificado en una ruta, en estos filtros se puede verificar por ejemplo si el usuario está autenticado y autorizado para acceder a dicha ruta, y en caso de no tener los permisos necesarios redirigirlo hacia otra página.

- Base de Datos: El framework viene con un ORM propio llamado “Eloquent” el cual facilita drásticamente las consultas a la base de datos, declarando las relaciones en los modelos y así pudiendo manipular los datos como objetos, en donde los atributos de un objeto se corresponden con un campo de su tabla en la base de datos. Por ejemplo para crear un usuario se lo instancia como un objeto, se le setean los campos y por último se lo guarda, olvidándonos de la clásica sintaxis de SQL, Eloquent hace todo el trabajo.

- Laravel provee algo llamado “Migraciones”, las cuales son clases que definen las tablas y los campos de dichas tablas de la base de datos. También se pueden hacer las relaciones con sus correspondientes claves foráneas. Esto simplifica mucho el trabajo ya que no es necesario estar tocando “phpmyadmin”, una vez creadas las migraciones se suben a la BD a través de un comando “php artisan migrate”. Y todo queda listo para trabajar.

- Relaciones en los modelos: Cada modelo es una tabla en la base de datos, se definen como una clase en donde se especifica con qué tabla se machea y con qué otros modelos está relacionado. Las relaciones permiten tener una mayor abstracción para el programador y pueden relacionarse tablas en “muchos a muchos”, “uno a muchos”, “uno a uno”. Con lo cual al momento de realizar una consulta en donde se requiera un merge de 2 o más tablas, simplemente se le pide a un objeto por su correspondiente objeto relación y un campo especificado. Por ejemplo un usuario para acceder a su rol que está definido en otra tabla, en vez de realizar un inner join de ambas tablas y quedarse con el id del rol que

tiene la tabla usuario, se le pide al objeto usuario su rol como si fuera un atributo de dicho objeto, y luego se pide por el atributo del rol de esta manera: $usuario->rol->nombre

- Vistas del sistema: Gracias al motor de plantillas propio de Laravel, “Blade”, pudimos aprovechar gran parte de la codificación de las vistas realizadas en el trabajo de la cursada que estaban hechas con “Twig”. Solo tuvimos que “traducir” la codificación de Twig a Blade, ahorrandonos gran parte del tiempo.

Mecanismo provisto para el manejo de seguridad y routing (si es que lo tiene).

Se implementa a través de middlewares. en donde se definen los permisos para los distintos tipos de usuarios. Si un usuario tiene permiso para acceder a una ruta, accede y sino es redirigido a otra página. Por ejemplo si un usuario no está logueado e intenta acceder a una ruta que precisa estar logueado, se niega el acceso y se redirige al login.

En cuanto al routing, como ya se ha mencionado antes, las rutas se definen en “routes.php” y de esta manera Laravel se encarga de todo el procesamiento, lo único que debe especificar el programador es el método http de la petición (get, post, put, delete), un url que será el que verá el usuario en la barra de navegación, un nombre de ruta que es único para esa ruta, los middlewares que se deseen en caso de que sean necesarios, y para ejecutar el pedido se puede definir una funcion php en la misma declaración de la ruta que puede retornar alguna vista o hacer algún procesamiento, o definir el controlador y el método a ejecutar para resolver el pedido.

Mecanismo provisto para operaciones CRUD (si es que lo tiene).

El mecanismo provisto para operaciones CRUD es a través de “Eloquent”. Es el ORM que incluye Laravel para manejar de una forma fácil y sencilla los procesos correspondientes al manejo de bases de datos en nuestro proyecto, gracias a las funciones que provee podremos realizar complejas consultas y peticiones de base de datos sin escribir una sola línea de código SQL. Eloquent funciona con diversos motores de base de datos, como PostgreSQL, MySQL, SQL Server.

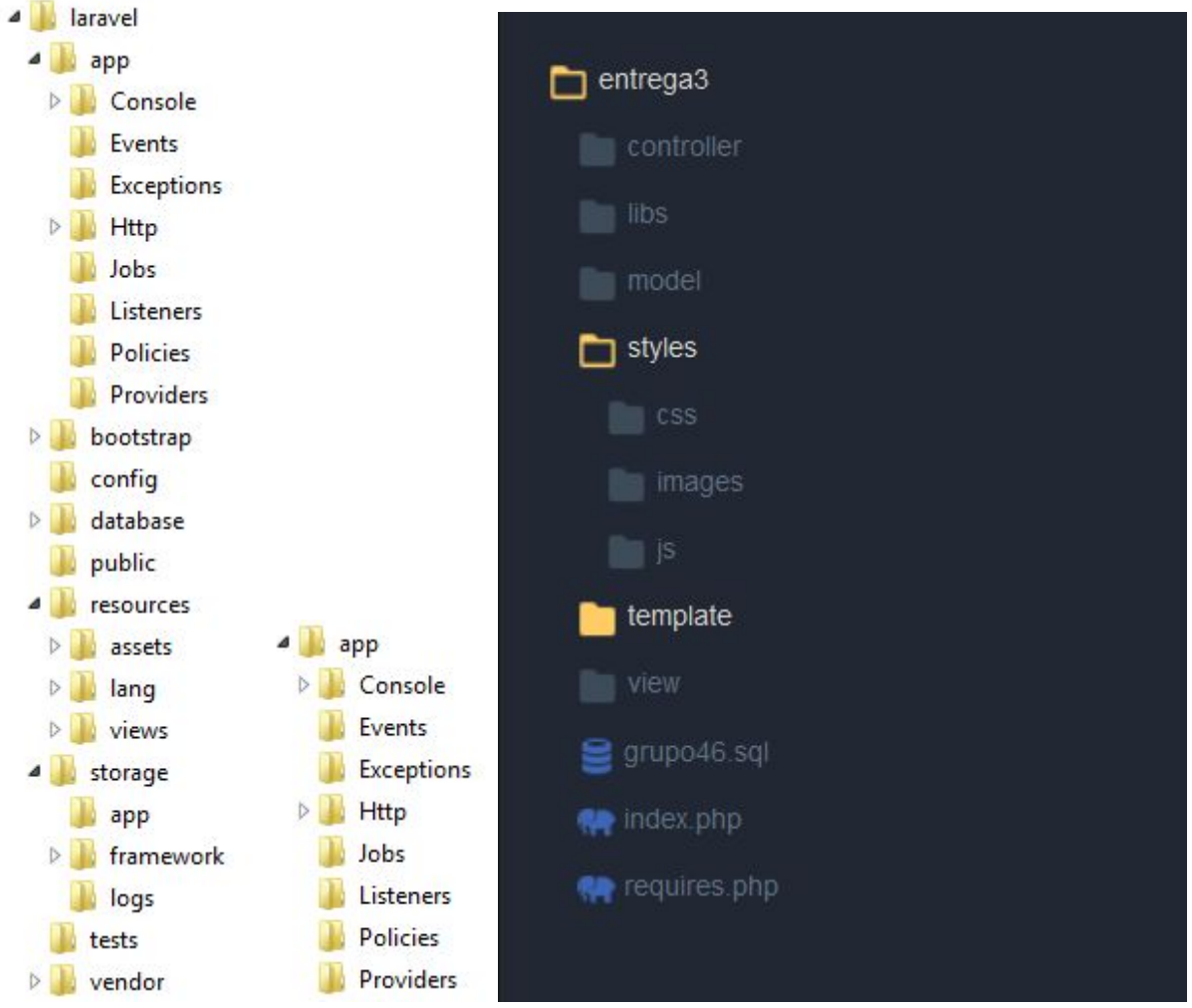
Laravel provee archivos de configuración, donde simplemente seteando los valores del servidor, nombre de la base de datos, usuario y contraseña, realiza la conexión automáticamente con PDO, por lo tanto programador no debe preocuparse por todo esto y solo se dedica a realizar su aplicación.

MVC: árbol de directorios

Laravel:

- app: donde convive toda la aplicación dentro de la misma nos encontramos con
- Http->controllers: controladores de la app
- http->route.php
- resources->views: Aquí se guardan los archivos de vistas y raw assets.
- public: archivos estáticos js,css,imágenes,etc.

La idea nuestra fue comparar a partir de nuestra estructura MVC, con el diseño de Laravel.



Si bien Laravel se basa en el patrón MVC, según Taylor Otwell el creador de Laravel, NO es un framework MVC estricto, ya que existen muchos más componentes que hacen que Laravel funcione. Un modelo, una vista y un controlador no hacen que Laravel funcione. En resumen es como un MVC más desarrollado. La curva de aprendizaje es acotada en comparación a otros frameworks y requieren conocimientos de POO y PHP para poder usarlo. Al principio fue complicado trasladar la aplicación MVC estricta, aunque una vez que se entendió el concepto, nos dimos la libertad de romper algunas estructuras.

Conclusion

Si bien el tiempo de desarrollo total brindado para migrar la aplicación al framework Laravel, fue una buena oportunidad para entrar en el mundo de frameworks. Hay muchas herramientas en las cuales podríamos haber hecho uso, como la confirmación del email registrado por parte del usuario, formularios predefinidos y otros servicios brindados por Laravel. Nos acotamos a migrar las herramientas fundamentales del trabajo, pudiendo así investigar e implementar los mecanismos de ruteo, seguridad y edición de datos de la aplicación.

El manejo de un ORM también fue nuevo para nosotros. A la hora de crear los objetos, nos simplificó la forma de leer y escribir algunos datos, otros tuvimos que dejarlos a mano como en el trabajo anterior. Por el lado de Telegram, decidimos dejarlo como estaba en la etapa anterior, aunque vimos que hay apis escritas por desarrolladores para simplificar el manejo de la api de Telegram.

Finalizando, un framework no es una necesidad absoluta, pero sin embargo es muy útil. Nos evitamos de repetir código que quizás ya desarrollamos en sistemas anteriores, siempre con alguna modificación, pero la base es la misma. Con respecto al sistema original creado durante la cursada, nos evitamos grandes pasos que no fueron necesarios migrar, ya que el framework contaba con ellos (entre ellos, ruteo, seguridad y manejo con la BD)

Bibliografía

- Documentation, en: https://laravel.com/docs/5.1/quickstart
- Laravel tutorial, en: https://www.tutorialspoint.com/laravel/
- Laracasts Laravel 5 Fundamentals: Basic MVC Workflow, en: https://laracasts.com/
- Architecture of Laravel Applications, en: http://laravelbook.com/laravel-architecture/