Informe Final de Promoción

Proyecto de Software - Cursada 2018

Autores:

Borgognoni, Fernando
Ciancio, Benjamín

Índice:

1. Fundamentos de la elección

Nuestra elección del framework se basó en varios detalles: la popularidad, la seguridad, la documentación, la facilidad de uso del framework y el rendimiento.

En cuanto a popularidad, ningún framework se compara con Laravel [1][2][3][4][5][6][7]. Tener una comunidad grande provee beneficios en muchos aspectos. Entre ellos, la seguridad es altamente beneficiada porque al haber más desarrolladores realizando programas con este framework, se encuentran con mayor velocidad las fallas de seguridad del framework, y se desarrollan más componentes seguras. Otro aspecto muy beneficiado es la documentación, que además de tener más desarrolladores trabajando sobre la documentación oficial, hay muchísimos tutoriales online de Laravel, como por ejemplo CodigoFacilito, el cual usamos nosotros, entre otros.

La seguridad es un tema sensible, puesto a que es muy importante en el desarrollo pero muchas veces queda relegado por el tiempo necesario para una implementación segura, por la reducción en la usabilidad del programa, por la incrementada dificultad de la programación y/o por el elevado costo de hacer programas más seguros.

---

[1] "laravel/laravel - GitHub." https://github.com/laravel/laravel.
[2] "symfony/symfony - GitHub." https://github.com/symfony/symfony.
[3] "bcit-ci/CodeIgniter: Open Source PHP Framework (originally ... - GitHub." https://github.com/bcit-ci/CodeIgniter.
[4] "CakePHP - GitHub." 28 May. 2019, https://github.com/cakephp.
[5] "GitHub zendframework." https://github.com/zendframework/zendframework.
[6] "Yii PHP Framework 1.1.x - GitHub." https://github.com/yiisoft/yii.
[7] "10 Popular PHP frameworks in 2019 · Raygun Blog." 21 Nov. 2018, https://raygun.com/blog/top-php-frameworks/.

Nosotros consideramos que la seguridad provista por Laravel es la más adecuada, debido a varias razones. Una razón importante es la compatibilidad con los paquetes provistos por PHP, gracias a composer, que permiten al desarrollador no reinventar la rueda, sino simplemente elegir al paquete más adecuado para el trabajo, y aplicarlo correctamente. Otra razón importante es la comunidad, mencionado anteriormente.

La documentación de un framework es muy importante, puesto a que es la principal fuente de información sobre su funcionamiento. Laravel tiene una documentación muy amplia [8], que provee, no sólo para la última versión, sino que para todas sus versiones, gran cantidad de información en su página principal. Además de la documentación oficial, es importante considerar también los aportes de la comunidad que usa Laravel, como fue mencionado hablando de la popularidad.

Laravel resulta fácil de usar, debido a que usa el paradigma de diseño de convención sobre configuración [9], y tiene una filosofía de diseño basada en dos cosas, la velocidad y la felicidad del desarrollador [10].

El rendimiento de Laravel no es el mejor entre los diferentes frameworks [11], pero nosotros creemos que para la dimensión de nuestro proyecto, la diferencia no es la suficiente para elegir otro framework, ya que en los demás puntos que consideramos, la ventaja de usar Laravel es considerable.

2. Aspectos técnicos evaluados

2.1 Mecanismos provistos para el manejo de seguridad y routing

Laravel provee muchos mecanismos y herramientas para que el sistema que se desarrolla sea seguro. En primer lugar, hace que la **autenticación** de los usuarios, que consiste en determinar que quien dice ser alguien efectivamente sea ese alguien, sea algo sencillo, porque provee de controladores que gestionan el registro de los usuarios, la autenticación propiamente dicha, el envío de mails en caso de olvido de la contraseña y el restablecimiento de la contraseña [12]. Las funcionalidades que dan estos controladores son suficientes para el desarrollo de la mayoría de los sistemas, incluído el nuestro, proporcionando automáticamente, por ejemplo, el cifrado de las contraseñas con Bcrypt.
Además, Laravel provee el comando ‘php artisan make:auth’ que genera todas las rutas y vistas necesarias para toda la autenticación, haciendo mucho más sencillo todo el trabajo.
En caso de que un usuario que no está autenticado intente acceder a un recurso donde es necesario que sí lo esté, Laravel ofrece un middleware para proteger aquellas rutas que lo necesiten. Incluso, si un usuario intenta iniciar sesión muchas veces ingresando credenciales no válidas, el usuario no podrá volver a intentarlo hasta que pase un minuto.

---

[8] "Installation - Laravel." https://laravel.com/docs/5.8.
[9] "Doctrine | Ruby on Rails." https://rubyonrails.org/doctrine/.
[10] "1. Why Laravel? - Laravel: Up and Running [Book] - O'Reilly." https://www.oreilly.com/library/view/laravel-up-and/9781491936078/ch01.html.
[11] "11 Best PHP Frameworks for Modern Web Developers in 2019." https://coderseye.com/best-php-frameworks-for-web-developers/.
[12] "Authentication - Laravel - The PHP ...." https://laravel.com/docs/5.8/authentication.

Otro mecanismo de seguridad que provee Laravel es el de **autorización** [13]. Esto quiere decir que provee un mecanismo para verificar que un usuario esté autorizado a acceder a un recurso determinado. Existen dos maneras de autorizar una acción: a través de gates o de políticas. Los gates son funciones anónimas que reciben un usuario y devuelven verdadero o falso, dependiendo si el usuario tiene autorización o no para realizar cierta acción. Las políticas, en cambio, son clases que manejan la lógica de autorización de un modelo completo.

En nuestro trabajo, utilizamos un paquete llamado 'Entrust' [14] que te permite agregar permisos y roles al sistema de una manera rápida y flexible. Este paquete agrega las tablas correspondientes a la base de datos, para que se puedan crear y manejar como cualquier otro modelo. Para el manejo de la autorización, Entrust verifica a partir de un middleware interno si el usuario logueado tiene el rol o el permiso necesario para acceder al recurso.

Laravel brinda protección ante ataques del tipo CSRF (cross-site request forgeries) generando tokens automáticamente para cada sesión para verificar que sea el usuario que está autenticado quien realmente hace las peticiones a la aplicación [15]. Lo único que debe hacer el desarrollador es agregar un campo a cada formulario, para que cuando ese formulario se envíe, se ejecute un middleware donde se comprueba que el token recibido es el mismo que está almacenado en la sesión del usuario autenticado.

Para el routing, Laravel [16] provee por defecto dos archivos de ruteos, api.php y web.php. Ambos pertenecen a su propio grupo de middleware. En cualquier archivo de ruteo, Laravel permite limitar los métodos HTTP que pueden acceder a una determinada ruta, ya sea uno solo, varios o todos. Para redirecciones, Laravel tiene un método llamado redirect(), que permite no tener que definir una nueva ruta, sino que se puede redirigir a una ya definida. Por defecto devuelve un código 302, pero se puede modificar.

Las rutas pueden recibir parámetros, que se definen entre {}. En caso de ser opcional, lleva un "?" después del nombre del parámetro. Por cuestiones de seguridad, Laravel permite hacer pattern matching con expresiones regulares, para limitar los parámetros recibidos.
Las rutas se pueden nombrar para facilitar su acceso.

2.2 Mecanismos provistos para operaciones de CRUD

Laravel provee un comando que genera un controlador con un protocolo de mensajes definido para el CRUD de un recurso [17]. El comportamiento de cada método debe ser implementado por el desarrollador, pero el framework da la posibilidad de mapear a cada uno de esos métodos con una ruta particular en una sola declaración. El resto de las operaciones relacionadas con CRUD deben ser implementadas por el desarrollador. Esto es un punto en contra para este framework, porque otros (Symfony, por ejemplo [18]) dan la posibilidad de mediante un solo comando, generar todo lo necesario para el CRUD de un recurso. Por ejemplo, todas las vistas para listar, obtener, crear, editar y eliminar, junto con su comportamiento y confirmaciones. Sin embargo, que Laravel no tenga esa funcionalidad

[13] "Authorization - Laravel - The PHP..." https://laravel.com/docs/5.8/authorization.
[14] "Zizaco/entrust - GitHub." https://github.com/Zizaco/entrust.
[15] "CSRF Protection - Laravel - The PHP..." https://laravel.com/docs/5.8/csrf.
[16] "Routing - Laravel." https://laravel.com/docs/5.8/routing.
[17] "Controllers - Laravel - The PHP..." https://laravel.com/docs/5.8/controllers.
[18] "Controller (Symfony Docs)." https://symfony.com/doc/current/controller.html.

hace que su flexibilidad aumente, porque uno puede implementar el CRUD de la manera que uno quisiera, sin tener que estar adaptando lo que el framework ofrece.

2.3 Forma de manejar el MVC

Laravel incluye un ORM basado en Active Record llamado Eloquent [19] para poder mapear modelos a tablas de una base de datos relacional. De esta manera Eloquent ofrece la posibilidad de trabajar directamente con estos modelos para poder consultar, insertar, editar o eliminar datos, permitiendo abstraerse completamente de cómo está todo almacenado realmente. Además, Eloquent ofrece una serie de convenciones para sus modelos para evitar mucha configuración. De manera automática, Eloquent mapeara un modelo a una tabla de la base de datos relacional a partir del “snake case” del plural del nombre del modelo (por ejemplo, si el modelo es Flight mapeara a la tabla flights).
Para realizar consultas a estos modelos, Laravel ofrece una interfaz llamada query builder, el cual puede usarse para la mayoría de las operaciones sobre la base de datos del sistema. Es una interfaz segura porque usa PDO automáticamente para evitar las ‘SQL Injections’ que pueden dañar nuestra información.

Los controladores agrupan toda la lógica sobre un recurso, para no tener que hacer todas funciones anónimas o Closures. Se implementan en clases que heredan funcionalidades por defecto que ofrece Laravel, por ejemplo una función para definir los middleware respectivos. Como fue mencionado anteriormente, Laravel ofrece la posibilidad de con un solo comando crear un controlador con el protocolo de mensajes necesario para hacer un CRUD sobre un recurso.

Las vistas en este framework se componen de archivos dentro de la carpeta ‘views’, y sirven para separar la lógica de presentación de la lógica de la aplicación [20]. Se utiliza un motor de plantillas llamado Blade [21], al cual se le puede pasar por parámetros la información necesaria para renderizar la vista como se desee.

3. Migración del trabajo de la cursada al framework

Para poder aprender con una mayor claridad y profundidad todas las funcionalidades que brinda Laravel, tomamos la decisión de realizar el trabajo nuevamente desde cero. Lo único que reutilizamos fue el esquema de la base de datos, el cual adaptamos al idioma inglés. Además, con el tiempo nos fuimos dando cuenta que el framework proveía funcionalidades para realizar las cosas con una facilidad interesante, por lo que no tenía sentido reutilizar ciertos módulos del trabajo de la cursada.

4. Referencias y material estudiado

Para poder hacer el trabajo, principalmente nos basamos en la documentación oficial de Laravel, la cual está muy bien explicada y tiene una versión en español. Además, hicimos

[19] "Eloquent - Laravel." https://laravel.com/docs/5.8/eloquent.
[20] "Views - Laravel - The PHP Framework For Web Artisans." https://laravel.com/docs/5.8/views.
[21] "Blade Templates - Laravel." https://laravel.com/docs/5.8/blade.

un tutorial en un canal del sitio YouTube llamado ‘códigofacilito’ [22], el cual mediante videos de ejemplos prácticos muestra cómo dar los primeros pasos en el framework. Finalmente, cuando teníamos algún problema que ni la documentación ni el canal de YouTube mencionado podían ayudarnos, teníamos que hacer una búsqueda minuciosa por internet, más que nada en foros como Stack Overflow [23], donde la gran comunidad de usuarios de Laravel ayudaban.

5. Conclusiones

En todos los meses que nos llevó realizar este trabajo de promoción, lo más importante que nos llevamos es que los frameworks brindan una enorme cantidad de soluciones que hacen del desarrollo de software algo mucho más simple y dinámico. En general, son difíciles de entender al principio, y lleva mucho tiempo poder hacer uso de todas las funcionalidades que proveen, pero sin lugar a dudas las facilidades que dan son muy notorias.

Pudimos darnos cuenta de la importancia del software libre, que nos brindó soluciones éticas para el desarrollo de nuestro sistema, evitando en muchos casos tener que crear nuestras propias funciones. En general, creemos que el uso del software libre es esencial para la innovación, y para el avance ético de la comunidad programadora. Nos gustó mucho la dinámica del trabajo en equipo, y creemos que es fundamental para desarrollar mejores sistemas, ya que permite tener más de un punto de vista a un problema determinado, y, por lo tanto, mayor probabilidad de éxito de encontrar una solución.

En cuanto a Laravel, la gran comunidad de usuarios que tiene fue de gran ayuda, ya que en cualquier momento que teníamos algún tipo de problema, siempre se encontraba alguien que ya le había ocurrido lo mismo y podía encontrar ayuda muy fácilmente. Además, como Laravel provee una estructura básica de un programa, nos resultó más sencillo adaptar nuestro sistema al framework. En términos de seguridad, creemos que usar Laravel nos brindó mejores resultados, gracias a los mecanismos ya mencionados en este informe.

[22] "1.- Curso Laravel - Introducción - CódigoFacilito." https://www.codigofacilito.com/videos/introduccion-3e794a2c-0bb3-431f-ab30-3907271769a4.
[23] "Stack Overflow - Where Developers Learn, Share, & Build Careers." https://stackoverflow.com/.