Introducción

Un Framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar. De esta manera, propone una manera de trabajar y organizar el desarrollo para poder trabajar en equipo, crecer rápida y poderosamente.

Puntualmente Laravel tiene como objetivo ser un framework que permita el uso de una sintaxis elegante y expresiva para crear código de forma sencilla y permitiendo multitud de funcionalidades. Intenta aprovechar lo mejor de otros frameworks y aprovechar las características de las últimas versiones de PHP.

A lo largo de este documento estaremos repasando los conceptos propuestos por Laravel vistos en las clases de Digital House. Dado esto, el foco de este documento tiene 2 objetivos además de comprender las herramientas técnicas:

1. Comprender **por qué** Laravel propone trabajar de esta manera.
2. Entender qué herramientas nos sirven para que momento y como todas congenian entre sí.

Dado esto, dividiremos a las herramientas en 3 categorías

1. Flujo habitual de la aplicación:

Dentro de este título enfocaremos las herramientas más típicas de Laravel apuntando a una pregunta concreta: Si tengo que crear un sitio web, ¿por dónde empiezo?

2. Herramientas de base de datos

En esta sección veremos qué herramientas nos da Laravel para consultar y modificar la base de datos.

3. Funcionalidades específicas

En esta última parte veremos herramientas puntuales para problemas específicos.

Durante este documento, también nos apoyaremos mucho en la documentación oficial de Laravel dado que es una fuente completa y constantemente actualizada, siendo la mejor herramienta para aprender y entender cómo se trabaja con Laravel diariamente en miles de contextos.

Flujo habitual de la aplicación

Routing

Documentación oficial: https://laravel.com/docs/master/routing

Al querer entrar en una página lo primero que nos preguntamos es "*¿por dónde entro?*". Dado esto, Laravel nos da un archivo de rutas para organizar las mismas. Esto nos permite:

1. Tener URLs SEO-friendly bastante sencillas de leer.
2. Control de los métodos HTTP por los cuales llega el pedido.
3. Manejo de comodines
4. En caso de querer ver **todos los puntos de entrada de mi aplicación** basta con revisar este archivo.

En una ruta, en general, se aclara:

- Método HTTP
- Ruta
- Controlador y método al que se deriva

Ejemplo:

```
Route::get('user/profile', 'UserController@showProfile');
```

Controllers

Documentación oficial: https://laravel.com/docs/master/controllers

En segundo lugar, una ruta, en la mayoría de los casos, deriva en un controlador. El controlador es la parte del código que va a **preparar todos los datos necesarios** para poder imprimir el HTML.

Es aquí donde aparece la lógica. Es aquí donde se consulta la base de datos. Es aquí donde podemos modificar la base de datos. Es aquí donde podemos validar. Es aquí donde podemos revisar permisos. Esta es formalmente nuestra capa de Backend.

Esto nos permite **separar responsabilidades** pudiendo un maquetador de Frontend ocuparse únicamente de la vista mientras que una persona responsable del Backend estaría modificando el controlador. Esto deriva en una mejor organización en equipos.

A través del método ***view*** podemos decir en qué vista se imprimirán los datos deseados.

```php
return view('user.profile', ['user' => $user]);
```

Esta vista se buscará siempre dentro de la carpeta de vistas y podemos dividir subcarpetas lo cual se ve en el ejemplo separado por un punto.

Vistas

Documentación oficial: https://laravel.com/docs/master/views
Documentación oficial: https://laravel.com/docs/master/blade

Las vistas serán nuestros documentos ***.blade.php*** que se comportan como un php en donde incluimos todo nuestro HTML y lo que el usuario ve.

Puntualmente, para facilitar ciertas cosas, Laravel permite sintaxis blade incluyendo:

- Estructuras de control
- Herencia de plantillas
- Mostrar datos provenientes del controller
- Incluir sub-vistas

Repaso parcial

Para crear una página en Laravel, **ya sea que queremos responder a un pedido mostrando información o modificando información**, de manera casi-obligatoria necesitamos crear:

1. Ruta
2. Controlador
3. Vista

Sin embargo, muchas veces, queremos traer, mostrar y modificar nuestra base de datos. Dado esto, sumamos dos herramientas más.

Modelos y Eloquent

Documentación oficial: https://laravel.com/docs/master/eloquent
Documentación oficial: https://laravel.com/docs/master/eloquent-relationships

Recordemos previamente, que para poder utilizar una base de datos necesitamos configurar la conexión en el archivo ***.env***

Una vez configurada la base de datos, en primer lugar, crearemos modelos que nos permitirán indicar cuales son tablas en la base de datos y **que objetos vamos a necesitar**

en nuestro sistema. Esto es exactamente la versión de las Clases que tenías en PHP para modelar nuestras tablas de la base de datos.

De la misma manera, le podemos sumar atributos y métodos aunque todos los atributos que se correspondan con columnas de base de datos ya son accesibles a través de Eloquent. Para que Eloquent funcione amenamente sin mayores aclaraciones se recomienda seguir en base de datos las convenciones especificadas por Laravel aquí.

Eloquent es un ORM. Es decir, Eloquent maneja completamente la conexión a base de datos, dándonos utilidades frecuentes para operar con ella de manera rápida y eficaz. A su vez, Eloquent se encarga de que podamos trabajar fácilmente con objetos a través de los modelos creados previamente.

Una vez creados los modelos, podemos traer o modificar información de base de datos con las siguientes flexibilidades:

- Traer una tabla entera
- Traer una sola fila de una tabla
- Insertar o modificar registros en base de datos
- Eliminar filas

En caso de querer trabajar con relaciones basta con modificar los modelos y aclarar cuales son las relaciones en la base de datos. Esto se puede ver aquí

Puntualmente podemos definir relaciones:

- Uno a uno
- Uno a muchos
- Muchos a uno
- Muchos a muchos
- Y más!

Repaso

Al crear un sitio tenemos que tener en cuenta:

1. Ruta
2. Controller
    a. En caso de querer acceder a base de datos podemos hacerlo a través de Eloquent y los Modelos
3. Vista

Herramientas de base de datos

Además de Eloquent, Laravel nos da ciertas herramientas para poder acceder y modificar la base de datos:

Seeders

Documentación oficial: https://laravel.com/docs/master/seeding

Los seeders son archivos que nos permitirán llenar la base de datos de información real o ficticia para poder tener datos concretos.

Para escribir un seeder debemos realizar los siguientes pasos:

1. php artisan make:seeder NombreDelSeeder
2. Escribir el seeder
3. Modificar *DatabaseSeeder* para explicitar que seeders se correrán
4. php artisan db:seed

Para escribir el seeder, tenemos 3 opciones:

- Utilizar DB::insert
- Utilizar DB::table
- Utilizar Model Factories

Las 3 opciones se pueden ver con mayor detalle en las filminas o en la documentación oficial. Sí cabe destacar que al utilizar Model Factories, se nos permite insertar datos azarosos de forma masiva y en ese caso es importante escribir la Fábrica de ese modelo como se especifica aquí

Migrations

Documentación oficial: https://laravel.com/docs/master/migrations

Las migrations nos permiten modificar la estructura de la base de datos, teniendo estos archivos en nuestro mismo proyecto, pudiendolos compartir de manera más prolija así como tener un registro de los cambios que se fueron haciendo.

Para esta herramienta, sí recomendamos meterse de lleno en la documentación ya que es bastante expresiva y completa.

Funcionalidades específicas

Colecciones

Documentación oficial: https://laravel.com/docs/master/eloquent-collections

A diferencia de los arrays con los que trabajamos en PHP plano, Laravel suele trabajar con collections. Operan de forma muy similar a los arrays pero con versatilidades útiles. Se pueden ver en mayor detalle en la documentación.

Request

Documentación oficial: https://laravel.com/docs/master/requests

Para poder tomar los datos que llegan a través del pedido (Request), Laravel deja de trabajar con $_POST, $_GET y otras variables superglobales y trabaja con un objeto Request. Para verlo en mayor detalle, se recomienda utilizar la documentación.

Validaciones

Documentación oficial: https://laravel.com/docs/master/validation

Laravel nos provee una forma sencilla de validar información provista por el usuario. Una vez definida la ruta y el controlador, podemos validar utilizando lo que se presta aquí.

Para mostrar los errores de forma sencilla se puede ver aquí.

La documentación provee más información en caso de querer hacer validaciones más complejas y/o específicas.

Middleware

Documentación oficial: https://laravel.com/docs/master/middleware

Un Middleware es una capa intermedia que se ejecuta luego de ser identificada la ruta pero previo a ejecutar el código en el controlador. Muchas veces, un middleware es compartido por más de una ruta y tiene el objetivo de hacer algo en común para muchas rutas. Un ejemplo típico, es crear un middleware para aquellos sitios que requieran estar logueados y poder denegar el acceso al usuario en un solo punto en el código.

Para generar esto debemos:

1. Generar el Middleware
2. Registrar el Middleware a las rutas o grupos de rutas deseadas

Manejo de sesión

Documentación oficial: https://laravel.com/docs/master/session

Laravel tiene una forma particular de manejar Sesión distinto a $_SESSION como vimos previamente.
Si bien la documentación provee bastantes detalles, aquí podemos encontrar cómo almacenar cosas en sesión mientras que aquí podemos ver cómo recuperarla.

i18n

Documentación oficial: https://laravel.com/docs/master/localization

Como en muchos ecosistemas, traducir nuestro sitio puede ser un problema tedioso.

En nuestros templates, basta con imprimir, en vez de textos fijos, cosas del estilo de:

```
{{ trans('messages.welcome') }}
```

De esta manera, el motor de Laravel, buscará un mensaje con el nombre *welcome* dentro del archivo *messages*. En la documentación se ve explícitamente dónde deben ubicarse estos archivos.

Para elegir el lenguaje seleccionado basta ejecutar:

```
App::setLocale($locale);
```

Como ejemplo, podríamos atrapar el lenguaje seleccionado en un Middleware obteniendo el dato de algo guardado en sesión o en base de datos según el usuario logueado.

Creaciones de APIs

Para generar una API rest con Laravel es muy sencillo. Un controlador está preparado para que si termina con un:

return $variable;

y dicha variable es un Objeto proveniente del modelo o una colección, Laravel devolverá un JSON que es lo que espera recibir una API.

Para interesados en métodos más complejos y eficaces, recomendamos investigar DINGO.

Registración y Login

Documentación oficial: https://laravel.com/docs/master/authentication

Laravel provee una forma sencilla de generar una registración y un Login en nuestro sitio, a sabiendas de que es una funcionalidad muy común y requerida.
Para esto, basta con ejecutar:
php artisan make:auth

Sin embargo, recomendamos explorar el código y leer la documentación para entender qué sucede de trasfondo al ejecutar dicho comando. En principio, un buen punto de partida, es analizar el archivo de rutas y ver que diferencias tiene tras correr dicho comando.