**Laravel**

Contenido

[Introducción Laravel 2](#_Toc2782138)

[Estructura de Carpetas 3](#_Toc2782139)

[Crear Proyecto 5](#_Toc2782140)

[Auth y generalidades 6](#_Toc2782141)

# Introducción Laravel

Antes de trabajar con **Laravel**, se necesita tener instalado *composer*:

<https://getcomposer.org/>

*Composer* es un manejador de dependencias de PHP, parecido a *npm* (dependencias de JavaScript).

*Para verificar que Composer quedo instalado solo basta digitar el comando: composer.*

Manipulación de Base de datos

Laravel permite conectarse de dos maneras a la base de datos:

* **Eloquent ORM:** No usa sentencias SQL. Laravel por debajo convierte a SQL.
* **Query Builder:** Parecido a las sentencias SQL.

Artisan

Parte de Laravel que permite ejecutar comandos. Es como un artesano, permite crear cosas, controladores, modelos.

# Estructura de Carpetas

Directorio App

Es el directorio donde se almacena todo el código personal del proyecto. Es considerado el directorio más importante de nuestro proyecto ya que es en el que más trabajaremos.

Uno de los subdirectorios más importantes es el **HTTP** en el cual se ubican Controllers y Middlewares.

Los **Modelos** se ubican a nivel de raíz de esta carpeta. Sin embargo, es posible almacenarnos en una carpeta llamada: *models.*

Directorio Config

Aquí se definen la configuración del framework y de la aplicación. La configuración de Laravel existe como un conjunto de archivos PHP que contienen matrices clave-valor. Entre los archivos más usados del directorio config se encuentran:

* *app.php:* En este archivo nos puede interesar configurar el lenguaje de nuestra aplicación, la zona horaria, los providers y aliases de las clases más comunes.
* *database.php:* En este archivo podemos configurar principalmente el motor de base de datos al cuál deseamos conectarnos.

Directorio Database

Aquí se encontrarán los archivos relacionados con el manejo de la base de datos. Dentro de este directorio se encuentran los subdirectorios:

* factories: Aquí escribiremos nuestros model factories.
* migrations: Todas las migraciones que creamos se ubican en este subdirectorio.
* seeds: Contiene todas las clases de tipo seed.

Directorio Public

Dentro de este directorio colocaremos todos los recursos estáticos de nuestra aplicación, es decir, archivos css, js, imágenes y fuentes. Es recomendable crear una carpeta por cada tipo de recurso.

Directorio Resources

Dentro de este directorio se encuentran los subdirectorios:

* **lang**: Aquí se encuentran todos los archivos de internacionalización, es decir, los archivos para poder pasar nuestro proyecto de un idioma a otro. Normalmente habrá una carpeta por cada idioma, ejemplo:
  + en: idioma inglés
  + es: idioma español
* **views**: Aquí ubicaremos nuestras vistas en formato php o php.blade, es recomendable crear una carpeta por cada controlador, además agregar una carpeta **templates** para las plantillas. Una plantilla es una vista general, que tiene segmentos que pueden ser reemplazados mediante la herencia de plantillas, más adelante se hablará de este tema

Directorio Storage

Cuando Laravel necesita escribir algo en el disco, lo hace en el directorio storage. Por este motivo, tu servidor web debe poder escribir en esta ubicación. Aquí podemos encontrar otros directorios entre los cuales el más relevante es el subdirectorio framework, es ahí donde se almacena el cache y las vistas compiladas.

Directorio Vendor

Esta carpeta contiene una cantidad de librerías externas, creadas por diversos desarrolladores que son dependencias de Laravel. La carpeta vendor no la debemos tocar para nada, porque la gestiona Composer, que es nuestro gestor de dependencias.

Si nosotros tuviésemos que usar una librería que no estuviera en la carpeta vendor la tendríamos que especificar en el archivo composer.json en el campo requerido. Luego hacer un "composer update" para que la nueva dependencia se instale.

Package.json

Es de JavaScript

Composer.json

Es el mismo *package.json* pero para PHP. Cuando se instale algo en Laravel, como librerías, se irán añadiendo aquí.

.env

Archivo de configuración. Configuraciones de base de datos, de llave de API, y otras cosas.

Routes

Definir rutas que llaman a un controlador generalmente (Normalmente, la vista se llama desde el controlador).

Pueden ser las rutas *web (son internas) o api (para conectar con otras aplicaciones).*

Resources

Aquí se muestran las vistas en la carpeta Views. Las vistas en PHP usan Blade que es un motor de plantillas.

# Crear Proyecto

Via Composer:

Con el siguiente comando se comienza a creer un proyecto de Laravel:

composer create-project --prefer-dist laravel/Laravel miProyecto

Una vez cree el proyecto, el primer comando recomendado a digitar después de crear el proyecto es el de *llave de aplicación:*

|  |
| --- |
| Yojhan@YOJHAN-PC MINGW64 /c/xampp/htdocs/First Laravel Project/myFirstLaravel (master)  $ php artisan key:generate --ansi |

Si no se genera la llave de aplicación, la cual es un string random, las sesiones de los usuarios y otros datos encriptados no estarán seguros.

**Ejecutar aplicación de Laravel:**

* Servidor local: Usando el comando definido por Laravel desde artisan:

php -S localhost:8000 -t public/

* Usando un servidor XAMMP y digitando la URL
* Creando un host virtual (Crear un dominio dentro del computador)

<https://medium.com/@jhordydelaguila/creando-y-configurando-virtual-host-con-apache-para-xampp-windows-f90c2b0527ac>

# Auth y generalidades

En el manual de Laravel, en la parte de seguridad se encuentra todo lo relacionado a la autenticación.

El primer comando a utilizar es:

**Yojhan@YOJHAN-PC MINGW64 /c/xampp/htdocs/Basic-Laravel-Projects/myFirstLaravel (master)**

**$ php artisan make:auth**

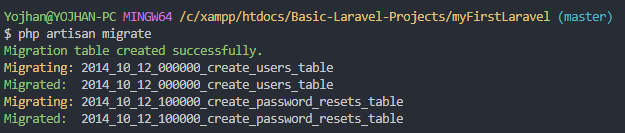
**Authentication scaffolding generated successfully.**

Al ejecutar el comando, en el proyecto hace las siguientes cosas:

* Crea rutas para la autenticación y una inicial para la página *home*
* Crea en las vistas del *login* junto con unos *layouts*
* Controladores de Auth en la carpeta de Controladores.
* Genera migraciones con la tabla de usuarios y una tabla para resetear contraseñas.

Las migraciones creadas de las dos tablas: *usuarios y reset passwords,* sirven para nuestro log-in. Solo basta ejecutar las migraciones de la siguiente manera:

**php artisan migrate**



**¡Importante! En algunas ultimas versions de Laravel, se debe definir el tamaño máximo de un String para la base de datos, en el archivo AppServiceProvider.php**

public function boot()

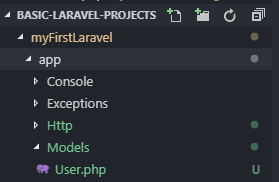
{

Schema::defaultStringLength(191);

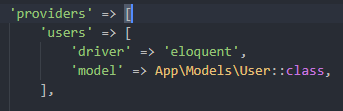
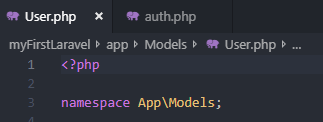
}

**Más información:** [**https://stackoverflow.com/questions/23786359/laravel-migration-unique-key-is-too-long-even-if-specified**](https://stackoverflow.com/questions/23786359/laravel-migration-unique-key-is-too-long-even-if-specified)

Es importante crear una carpeta de ***Models*** para guardar ahí los modelos y no en la carpeta raíz de la siguiente manera:



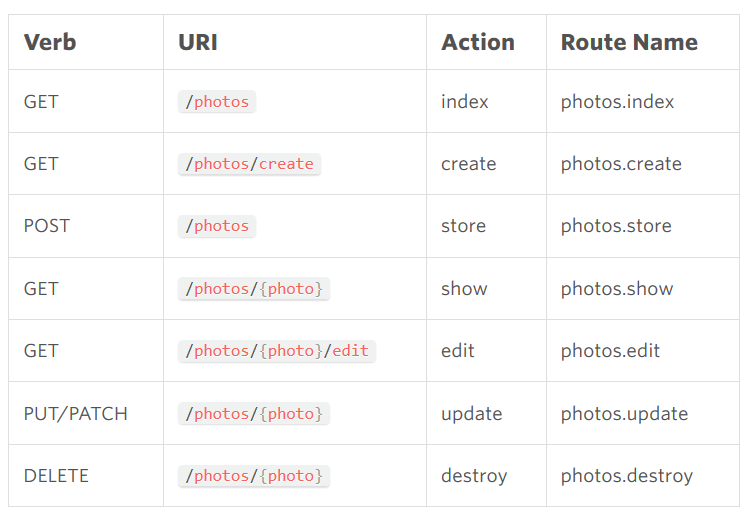
Sin embargo, se debe cambiar el ***namespace***del modelo y también en la parte de autenticación cambiar la ruta. Esto garantiza que el cambio realizado, no tenga alguna consecuencia en el proyecto.



# Controladores

Resource Controller

Los controladores tipo RESTful de Laravel se le llaman controladores de recurso (*Resource controller).* Estos controladores tienen los siguientes métodos:



Este tipo de controlador es para asignar rutas rápidamente del típico CRUD al controlador con una simple línea de código.

php artisan make:controller ProductController --resource

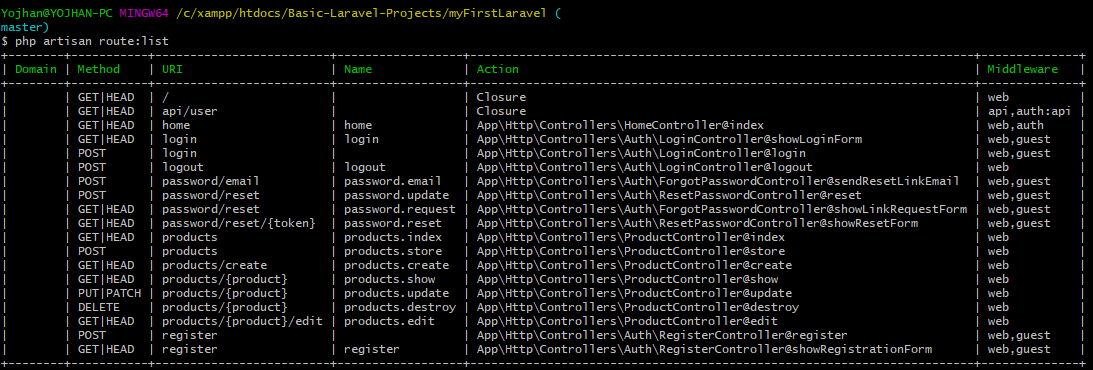
Este comando generará un controlador que contendrá un método para cada una de las operaciones HTTP. También se debe proceder configurar el archivo de rutas:

Route::resource('products', ProductController);

También para mayor comodidad, se puede asignar el Modelo al controlador, para que me cree un controlador resource implementando el modelo:

php artisan make:controller ProductController --resource --model=Product

Lista de Rutas:



Añadir métodos adicionales al Controlador Resource

Quizá se quiera añadir otros métodos adicionales a los que vienen con el Controlador RESTful.

Simplemente se usa el tipo de petición y se indica el nombre y el controlador:

*// Add as many routes as you need...*

Route::post('login', 'PostsResourceController@login');

Route::get('find', 'PostsResourceController@find');

Route::get('search', 'PostsResourceController@search');