A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

26-12-2017

# Trabajo de investigación - Laravel

Several thin, curved lines in shades of blue and grey originate from the bottom left and sweep upwards and to the right.

Jaime Clemente Carbonel  
DESARROLLO DE APLICACIONES WEB

## Contenido

1.- Descripción, características y ventajas del framework. ....	2
2.- Instalación y/o configuración del framework. ....	3
2.1.- Instalar Composer: .....	3
3.- Ejemplo de aplicación web que utiliza el framework: .....	6
3.1.- Script de creación de la base de datos, creación de la tabla e inserción de datos.....	6
3.2.- Creación de un proyecto usando el framework.....	6
3.3.- Estructura del proyecto generado (indicando las carpetas y los archivos más importantes).....	8
3.4.- Mapa de navegación entre las páginas de la aplicación web. ....	11
4.- Bibliografía y/o webgrafía.....	12

## 1.- Descripción, características y ventajas del framework.

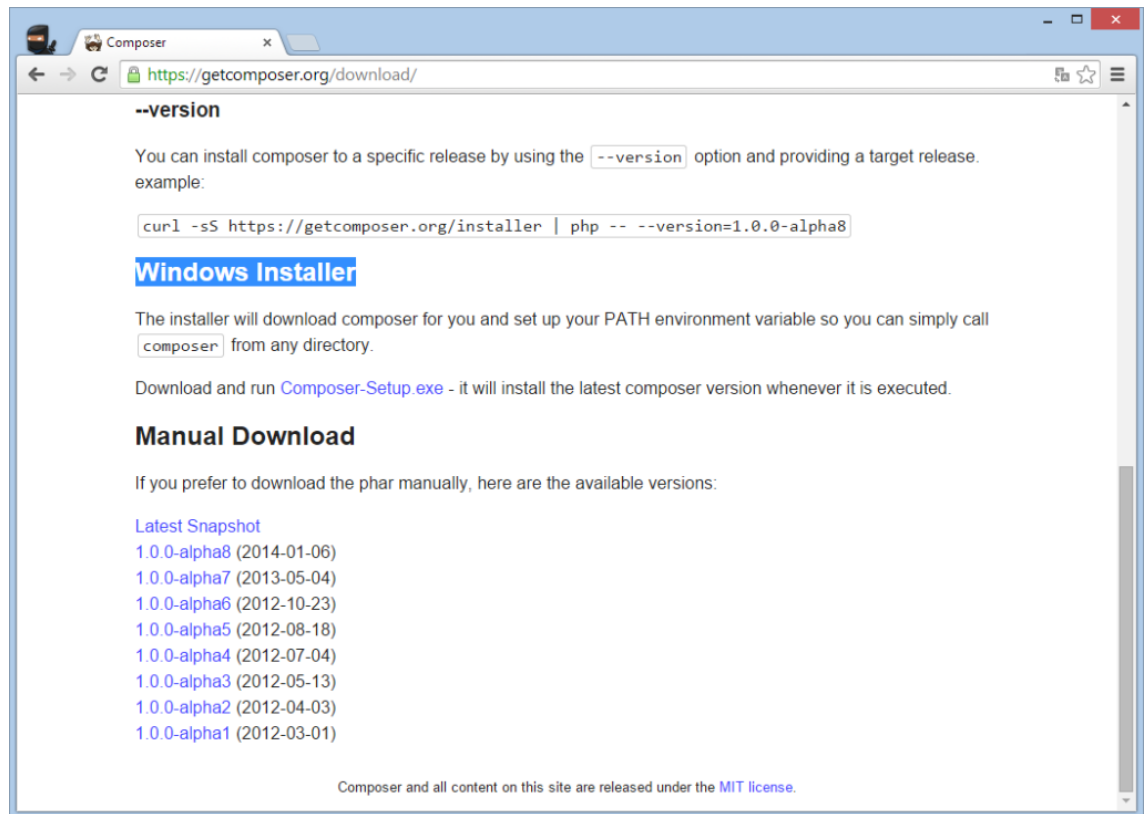
- Reducción de costos y tiempos en el desarrollo y mantenimiento.
- Curva de aprendizaje relativamente Baja (en comparación con otros framework Php).
- Flexible y adaptable no solo al MVC Tradicional (Modelo vista controlador) sino que para reducir código propone usar “Routes with clousures”
- Buena y abundante documentación sobre todo en el sitio oficial.
- Posee una amplia comunidad y foros.
- Es modular y con una amplio sistemas de paquetes y drivers con el que se puede extender la funcionalidad de forma fácil, robusta y segura.
- Hace que el manejo de los datos en Laravel no sea complejo; mediante Eloquent (que es un ORM basado en el patrón active record) la interacción con las bases de datos es totalmente orientada a objetos, siendo compatible con la gran mayoría de las bases de datos del mercado actual y facilitando la migración de nuestros datos de una forma fácil y segura. Otro punto es que permite la creación de consultas robustas y complejas.
- Facilita el manejo de ruteo de nuestra aplicación como así también la generación de url amigables y control de enlaces auto–actualizables lo que hace más fácil el mantenimiento de un sitio web.
- El sistema de plantillas Blade de Laravel, trae consigo la generación de mejoras en la parte de presentación de la aplicación como la generación de plantillas más simples y limpias en el código y además incluye un sistema de cache que las hace más rápidas, lo que mejora el rendimiento de la aplicación.
- También cuenta con una herramienta de interfaces de líneas de comando llamada Artisan que me permite programar tareas programadas como por ejemplo ejecutar migraciones, pruebas programadas, etc.

## 2.- Instalación y/o configuración del framework.

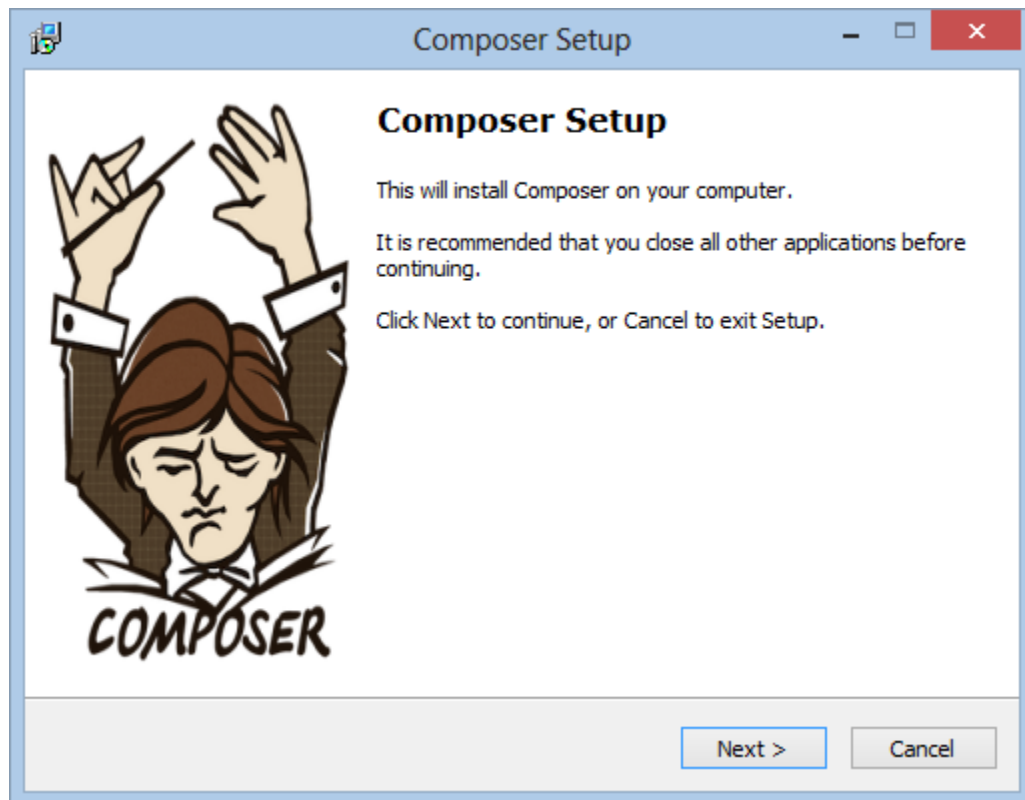
### 2.1.- Instalar Composer:

Composer es una herramienta para gestionar las dependencias en PHP. Te permite declarar las librerías de las cuales tu proyecto depende o necesita y las instala en el proyecto por ti.

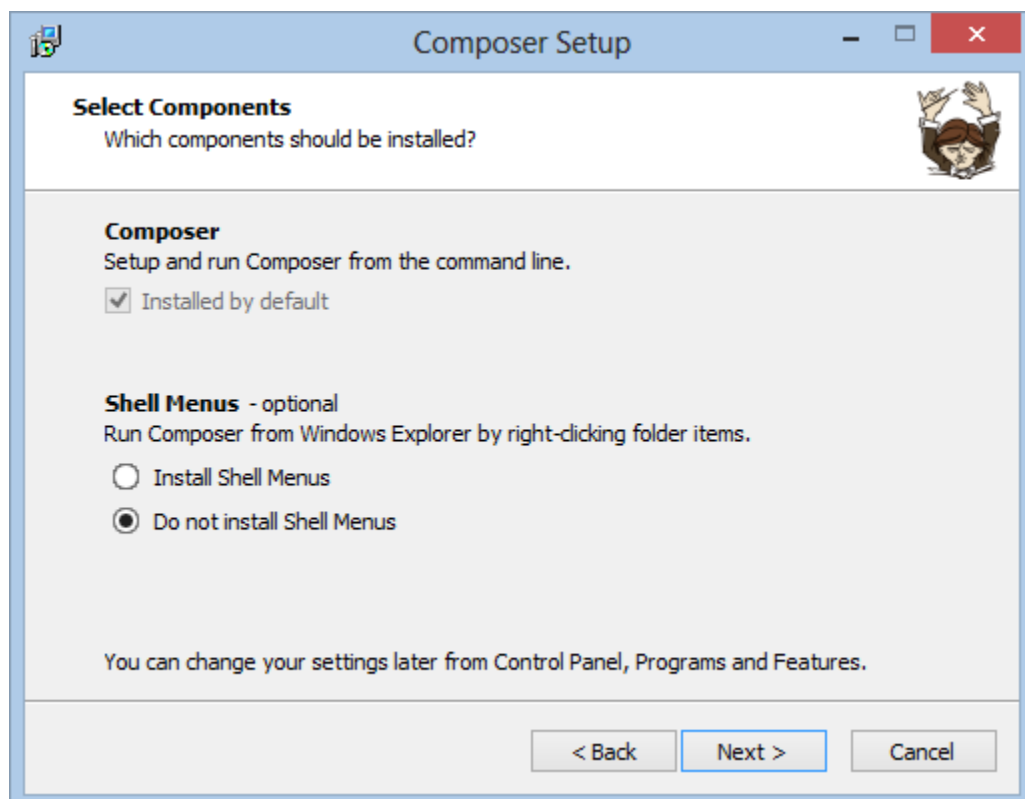
Para instalar Composer en Windows debemos descargarlo de su página oficial y en la sección Windows Installer seleccionar Composer-Setup.exe.



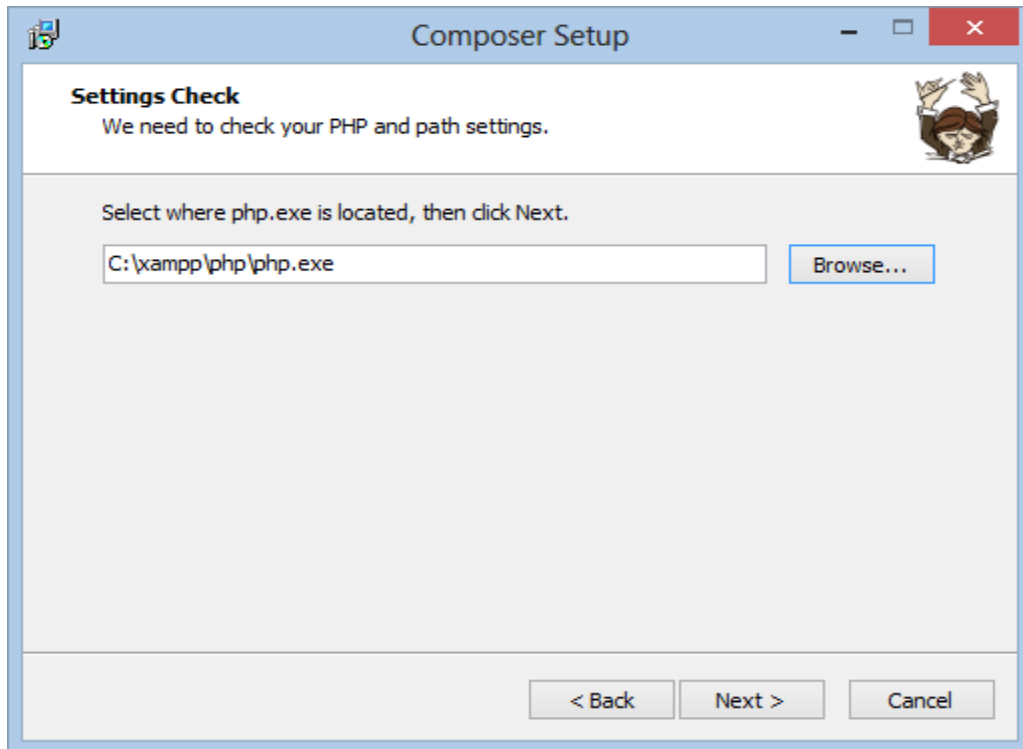
Una vez que la descarga finalice, ejecuta el instalador y haz click en Next.



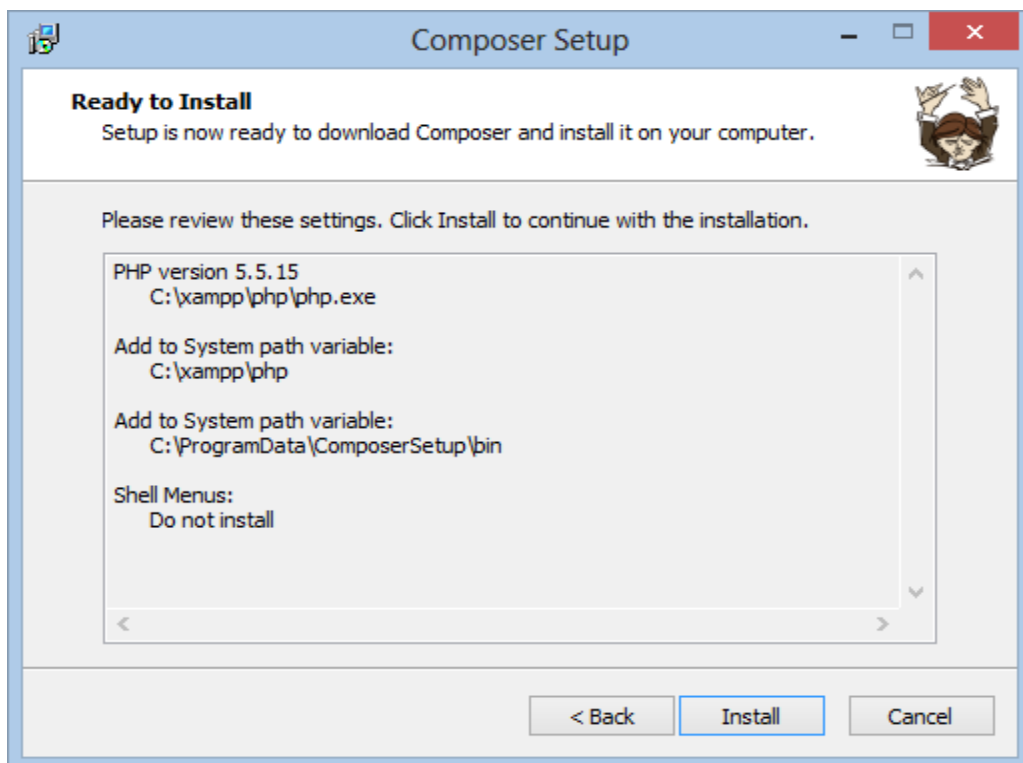
Si quieres administrar tus proyectos mediante el Explorador de Windows puedes seleccionar la opción "Install Shell Menus" aunque lo recomendable es la usar la línea de comandos.



A continuación nos pide que indiquemos la ruta del ejecutable de PHP, en mi caso como estoy trabajando con XAMPP el ejecutable de PHP se encuentra en la ruta C:\xampp\php\ y seleccionas php.exe, luego click en Next.



En este punto el instalador de Composer nos muestra la configuración de la instalación, simplemente le damos click a Install.



Una vez esté todo instalado, aparecerán otras donde simplemente debes hacer click en Next, y posteriormente en Finalizar.

### 3.- Ejemplo de aplicación web que utiliza el framework:

#### 3.1.- Script de creación de la base de datos, creación de la tabla e inserción de datos.

Se entrega junto con la memoria el script de creación de la base de datos.

#### 3.2.- Creación de un proyecto usando el framework.

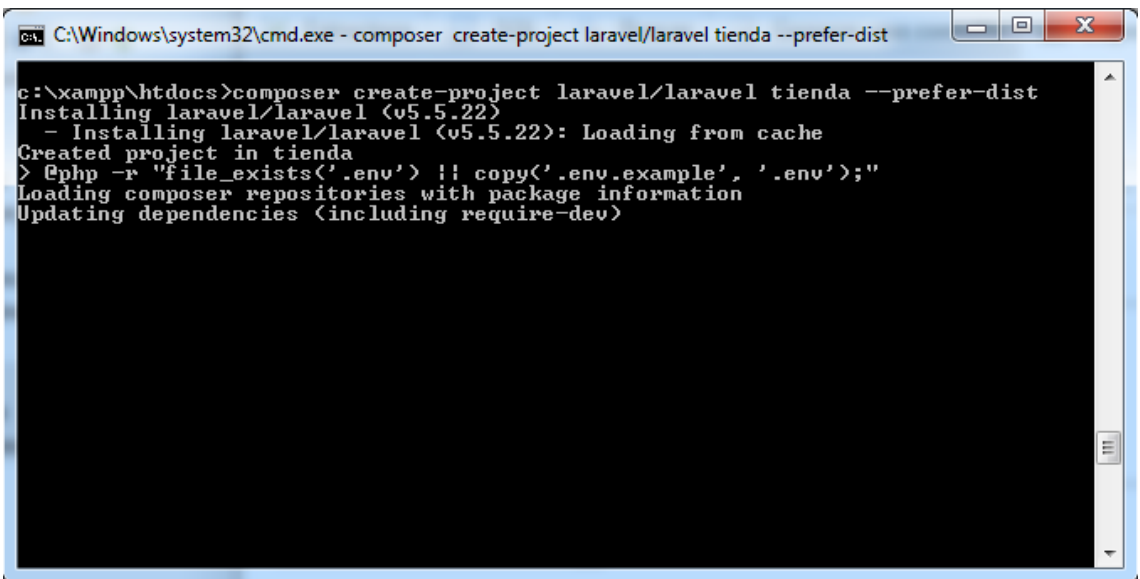
Existen dos formas de crear un proyecto con Laravel, la primera es descargando el archivo master desde su repositorio oficial de GitHub y la otra es usando Composer desde la consola que es precisamente lo que haremos en esta ocasión.

Desde la consola, dirígete al directorio donde guardas tus proyectos web (si usas XAMPP la ruta es C:\xampp\htdocs), y teclea lo siguiente:

```
cd C:\xampp\htdocs
```

Ahora crearemos el proyecto laravel escribiendo lo siguiente:

```
composer create-project laravel/laravel nombre_del_proyecto --prefer-dist
```



```
C:\Windows\system32\cmd.exe - composer create-project laravel/laravel tienda --prefer-dist

c:\xampp\htdocs>composer create-project laravel/laravel tienda --prefer-dist
Installing laravel/laravel (v5.5.22)
- Installing laravel/laravel (v5.5.22): Loading from cache
Created project in tienda
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
```

Composer empezará a descargar las librerías necesarias para nuestro proyecto, esto requiere un poco de tiempo.

```
C:\Windows\system32\cmd.exe - composer create-project laravel/laravel tienda --prefer-dist

- Installing sebastian/object-enumerator (3.0.3): Loading from cache
- Installing sebastian/global-state (2.0.0): Loading from cache
- Installing sebastian/exporter (3.1.0): Loading from cache
- Installing sebastian/environment (3.1.0): Loading from cache
- Installing sebastian/diff (2.0.1): Loading from cache
- Installing sebastian/comparator (2.1.1): Loading from cache
- Installing doctrine/instantiator (1.1.0): Loading from cache
- Installing phpunit/php-text-template (1.2.1): Loading from cache
- Installing phpunit/phpunit-mock-objects (5.0.5): Loading from cache
- Installing phpunit/php-timer (1.0.9): Loading from cache
- Installing phpunit/php-file-iterator (1.4.5): Loading from cache
- Installing theseer/tokenizer (1.1.0): Loading from cache
- Installing sebastian/code-unit-reverse-lookup (1.0.1): Loading from cache
- Installing phpunit/php-token-stream (2.0.2): Loading from cache
- Installing phpunit/php-code-coverage (5.3.0): Loading from cache
- Installing webmozart/assert (1.2.0): Loading from cache
- Installing phpdocumentor/reflection-common (1.0.1): Loading from cache
- Installing phpdocumentor/type-resolver (0.4.0): Loading from cache
- Installing phpdocumentor/reflection-docblock (4.2.0): Loading from cache
- Installing phpspec/prophecy (1.7.3): Loading from cache
- Installing phar-io/version (1.0.1): Loading from cache
- Installing phar-io/manifest (1.0.1): Loading from cache
- Installing myclabs/deep-copy (1.7.0): Loading from cache
- Installing phpunit/phpunit (6.5.5): Loading from cache
```

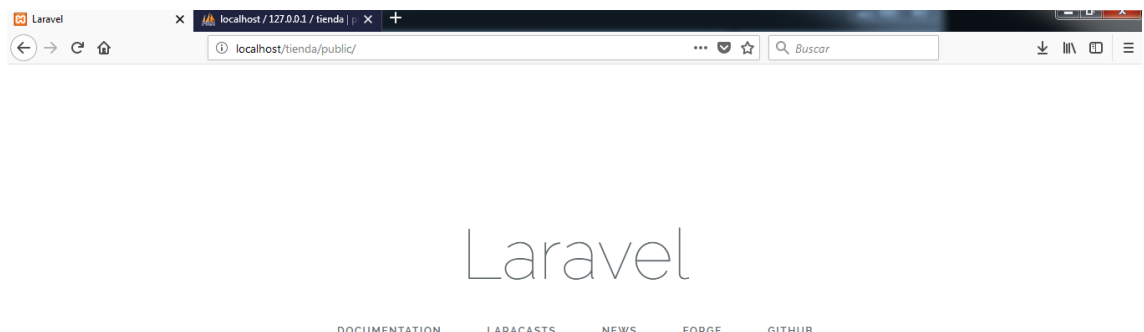
Si no ocurrió algún problema de conexión a Internet veremos que nuestro proyecto “tienda” se creó correctamente.

```
C:\Windows\system32\cmd.exe

psy/psysh suggests installing ext-posix (If you have PCNTL, you'll want the POSIX extension as well.)
psy/psysh suggests installing ext-pdo-sqlite (The doc command requires SQLite to work.)
psy/psysh suggests installing hoa/console (A pure PHP readline implementation. You'll want this if your PHP install doesn't already support readline or libedit.)
filp/whoops suggests installing whoops/soap (Formats errors as SOAP responses)
sebastian/global-state suggests installing ext-uopz (*)
phpunit/phpunit-mock-objects suggests installing ext-soap (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.5.5)
phpunit/phpunit suggests installing phpunit/php-invoker (^1.1)
phpunit/phpunit suggests installing ext-xdebug (*)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Package manifest generated successfully.
> @php artisan key:generate
Application key [base64:rMza0cjRX1/Ab8HZqL6jIiXFcKPT6nviY90M1BxEQQA=] set successfully.

c:\xampp\htdocs>
```

Finalmente para verificar que la creación de nuestro proyecto “tienda” se realizó de manera correcta, accede a [http://localhost/nombre\\_del\\_proyecto/public](http://localhost/nombre_del_proyecto/public) en el navegador de tu preferencia, donde debes ver lo siguiente:





### 3.3.- Estructura del proyecto generado (indicando las carpetas y los archivos más importantes).

Los archivos que tenemos sueltos en la carpeta raíz de Laravel son los siguientes:

#### **.env**

Es la definición de las variables de entorno. Podemos tener varios entornos donde vamos a mantener la ejecución de la aplicación con varias variables que tengan valores diferentes. Temas como si estamos trabajando con el debug activado, datos de conexión con la base de datos, servidores de envío de correo, caché, etc.

#### **composer.json**

Que contiene información para Composer.

Además en la raíz hay una serie de archivos que tienen que ver con Git, el readme, o del lado frontend el package.json o incluso un gulpfile.js que no vendría muy al caso comentar aquí porque no son cosas específicas de Laravel.

#### **Carpeta 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 require. Luego hacer un "composer update" para que la nueva dependencia se instale.

#### **Carpeta storage**

Es el sistema de almacenamiento automático del framework, donde se guardan cosas como la caché, las sesiones o las vistas, logs, etc. Esta carpeta tampoco la vamos a tocar directamente, salvo que tengamos que vaciarla para que todos esos archivos se tengan que generar de nuevo.

También podemos configurar Laravel para que use otros sistemas de almacenamiento para elementos como la caché o las sesiones.

En cuanto a las vistas cabe aclarar que no son las vistas que vamos a programar nosotros, sino las vistas una vez compiladas, algo que genera Laravel automáticamente en función de nuestras vistas que meteremos en otro lugar.

#### **Carpeta resources**

En Laravel 5 han creado esta carpeta, englobando distintos tipos de recursos, que antes estaban dentro de la carpeta app. En resumen, en esta carpeta se guardan assets, archivos de idioma (lang) y vistas.

Dentro de views tienes las vistas que crearás tú para el desarrollo de tu aplicación. En la instalación básica encontrarás una serie de subcarpetas con diversos tipos de vistas que durante el desarrollo podrías crear, vistas de emails, errores, de autenticación. Nosotros podremos crear nuevas subcarpetas para organizar nuestras vistas.

A propósito, en Laravel se usa el motor de plantillas Blade.

### **Carpeta Public**

Es el denominado "document root" del servidor web. Es el único subdirectorio que estará accesible desde fuera mediante el servidor web. Dentro encontrarás ya varios archivos:

#### **.htaccess**

En el caso de Apache, este es el archivo que genera las URL amigables a buscadores.

#### **favicon.ico**

Es el icono de nuestra aplicación, que usará el navegador para el título de la página o al agregar la página a favoritos.

#### **index.php**

Este es un archivo muy importante, que hace de embudo por el cual pasan todas las solicitudes a archivos dentro del dominio donde se está usando Laravel. Para el que conozca el patrón "controlador frontal" o "front controller" cabe decir que este index.php forma parte de él.

#### **robots.txt**

Que es algo que indica las cosas que puede y no puede hacer a la araña de Google y la de otros motores de búsqueda.

En la carpeta public podrás crear todas las subcarpetas que necesites en tu sitio web para contener archivos con código Javascript, CSS, imágenes, etc.

### **Carpeta database**

Contiene las alimentaciones y migraciones de la base de datos que veremos más adelante.

### **Carpeta bootstrap**

Permite el sistema de arranque de Laravel, es otra carpeta que en principio no necesitamos tocar.

### **Carpeta config**

Esta carpeta contiene toda una serie de archivos de configuración. La configuración de los componentes del framework se hace por separado, por lo que encontraremos muchos archivos PHP con configuraciones específicas de varios elementos que seguramente reconoceremos fácilmente.

La configuración principal está en app.php y luego hay archivos aparte para configurar la base de datos, las sesiones, vistas, caché, mail, etc.

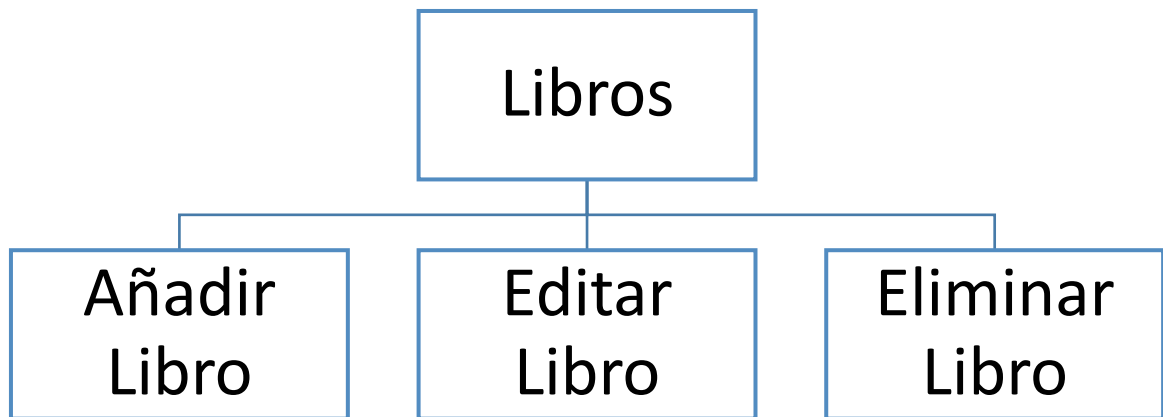
### **Carpeta app**

Es la última que nos queda y es la más importante. Tiene a su vez muchas carpetas adicionales para diversas cosas. Encuentras carpetas para comandos, para comandos de consola, control de eventos, control de excepciones, proveedores y servicios, etc.

Es relevante comentar que en esta carpeta no existe un subdirectorio específico para los modelos, sino que se colocan directamente colgando de app, como archivos sueltos.

La carpeta Http que cuelga de App contiene a su vez diversas carpetas importantes, como es el caso de aquella donde guardamos los controladores, middleware, así como el archivo de rutas de la aplicación routes.php que vimos anteriormente, entre otras cosas.

3.4.- Mapa de navegación entre las páginas de la aplicación web.



#### 4.- Bibliografía y/o webgrafía.

Para realizar la memoria he investigado y sacado información de las siguientes fuentes:

<https://www.freelancer.es/community/articles/ventajas-del-framework-moda-laravel>

[https://www.tutorialspoint.com/laravel/laravel\\_forms.htm](https://www.tutorialspoint.com/laravel/laravel_forms.htm)

<https://desarrolloweb.com/articulos/estructura-carpetas-laravel5.html>

<http://laraadmin.com/>

<https://www.cursosgs.es/cursos/login/index.php>

<https://laravel.com/>

<https://es.stackoverflow.com/>

<https://styde.net/laravel-desde-cero/>

<https://styde.net/laravel-5/>