

LARAVEL PILL

1. ANÁLISIS GENERAL

1.1. INVESTIGACIÓN

¿Qué es un **Framework** y qué ventajas tiene?

Un framework, se puede traducir como entorno de trabajo o marco de trabajo. Este marco ofrece 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.

¿Qué ventajas y desventajas tiene el uso de un framework?

Ventajas	Desventajas
Ventajas de la programación orientada a objetos (POO)	Curva de aprendizaje más elevada.
Patrón de diseño Modelo Vista Controlador (MVC)	Desconocimiento del funcionamiento del core del framework.
Trabajo con bases de datos, mapeo relacional de objetos(ORM), constructores de consultas, etc.	A mayor abstracción, menor rendimiento.
Trabajo con formularios y validación facilitado	A veces, sufren muchos cambios de código entre versión y versión.
Enrutamiento y URLs amigables.	
Seguridad.	
Librerías y funcionalidades ya hechas.	

¿Qué es **Laravel** en términos generales?

Laravel es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7. Su filosofía es desarrollar código PHP de forma elegante y simple, evitando el "código espagueti". Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC.

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.²

Gran parte de Laravel está formado por dependencias, especialmente de Symfony, esto implica que el desarrollo de Laravel dependa también del desarrollo de sus dependencias.

¿Qué requisitos son necesarios para usar **Laravel**?

El framework Laravel tiene algunos requisitos del sistema. Todos estos requisitos son cubiertos por la máquina virtual Laravel Homestead, así que es altamente recomendable que uses Homestead como tu entorno local de desarrollo de Laravel.

Sin embargo, si no estás utilizando Homestead, deberás asegurarte de que tu servidor cumpla con los siguientes requisitos:

- PHP >= 7.2.0
- BCMath PHP Extension
- Ctype PHP Extension
- JSON PHP Extension
- Mbstring PHP Extension
- Extensión OpenSSL de PHP
- Extensión PDO de PHP
- Extensión Tokenizer de PHP
- Extensión XML de PHP

¿Qué es **blade** y para qué sirve?

Blade es el sistema de plantillas de Laravel, el cual nos permite generar HTML dinámico con una sintaxis mucho más limpia que si usáramos PHP plano.

¿Qué es **Artisan** y para que se usa?

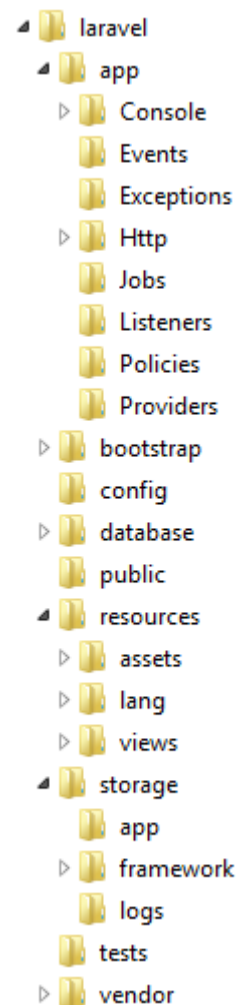
Artisan es la interfaz de línea de comandos (CLI por sus siglas en inglés de Command-line interface), la cual es un medio para la interacción con la aplicación donde los usuarios (en este caso los desarrolladores) dan instrucciones en forma de línea de texto simple o línea de comando. Artisan está basado en el componente Console de Symfony y nos ofrece un conjunto de comandos que nos pueden ayudar a realizar diferentes tareas durante el desarrollo e incluso cuando la aplicación se encuentra en producción.

¿Cómo se organiza **Laravel** a nivel de arquitectura por defecto?

La arquitectura de Laravel es un flujo de comunicación entre el *Foundation* del framework, los *Services Providers*, una estructura de Controllers con *Middlewares* y una capa de servicios que se comunica con el acceso a datos del *ORM* y al final, con la base de datos.

Estructura de carpetas

Al empezar una nueva aplicación con Laravel, se creará una estructura muy ordenada de directorios que nos facilitará trabajar con él.) El uso de convención sobre configuración genera un ambiente muy estandarizado que facilita, incluso, el integrar nuevos programadores a tus desarrollos. El directorio de una aplicación de Laravel se ve de la siguiente forma:



¿Dónde almacena los recursos públicos (css, js, imágenes, etc) ?

En el directorio public. Dentro de este directorio colocaremos todos los recursos estáticos de nuestra aplicación, es decir, archivos css, js, imágenes y fuentes.

¿Dónde almacena las vistas?

En el directorio resources encontramos las siguientes carpetas:

- assets : Aquí se ubican todos los archivos less de nuestra aplicación (útil para desarrolladores front-end).
- 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.

¿Dónde almacena los controladores? ¿Dónde se almacenan las rutas?

El directorio app es usado para ofrecer un hogar por defecto a todo el código personal de tu proyecto. Eso incluye clases que puedan ofrecer funcionalidad a la aplicación, archivos de configuración y más. Es considerado el directorio más importante de nuestro proyecto ya que es en el que más trabajaremos.

El directorio app tiene a su vez otros subdirectorios importantes, pero uno de los más utilizados es el directorio Http en el cuál ubicaremos nuestros Controllers, Middlewares y Requests en sus carpetas correspondientes, además dentro del subdirectorio Http encontremos también el archivo routes.php donde escribiremos las rutas de la aplicación.

¿Dónde almacena los modelos?

A nivel de la raíz del directorio app encontraremos el modelo User.php, los modelos comunmente se ubicarán a nivel de la raíz de la carpeta app, aunque igual es posible estructurarlos de la forma que queramos, por ejemplo, en una carpeta llamada Models.

¿Qué son los **middlewares**?

Los Middlewares son objetos que cubren las cosas que siempre necesitas en una web app. Cosas como cifrado (encriptación) de cookies, autenticación y usuarios, protección contra CSRF y XSS, etc.

¿Qué es **Eloquent** y para que se usa?

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.

¿Qué es un ORM?

El Mapeo Objeto-Relacional o como se conocen comúnmente, ORM (del inglés Object Relational Mapping) te permite convertir los datos de tus objetos en un formato correcto para poder guardar la información en una base de datos (mapeo) creándose una base de datos virtual donde los datos que se encuentran en nuestra aplicación, quedan vinculados a la base de datos (persistencia).

¿Qué ventajas y desventajas consideras que tiene el uso de un ORM respecto al uso de SQL Queries?

Ventajas	Desventajas
Facilidad y velocidad de uso	En entornos con gran carga poner una capa más en el proceso puede mermar el rendimiento.
Abstracción de la base de datos usada.	
Seguridad de la capa de acceso a datos contra ataques.	Aprender el nuevo lenguaje del ORM.

¿Qué es y cómo funciona un sistema de logs?

Cada vez que alguien accede a una página web lo que realmente está haciendo es requerir información al servidor (Código web) donde se encuentran todos esos datos alojados. A estas peticiones al servidor también se les puede llamar “Request”.

Bien, una vez se piden al servidor estos datos, este los brinda y en función de los datos recibidos se pinta la parte visual de la propia página web. Lo que tu propiamente ves en tu navegador.

Al brindar este formato visual, lo que se realmente está sucediendo es que el navegador una vez recibidos los datos del servidor tras realizar todas las consultas sobre todos los archivos que componen la página web como son los archivos HTML, CSS y JS, las imágenes, etc interpreta dichos archivos y los “traduce” a nivel visual para que tú los entiendas.

Es decir, por cada visita que se hace a cada una de las URLs de tu página web se pueden llegar a registrar hasta 50 logs entre archivos HTML, CSS, JS, imágenes, etc.

Si encima esa visita navega por 10 páginas distintas se habrán registrado 500 logs...Imagínate la cantidad de logs que se generarán en un portal de 2000 visitas al día...

Esto no es del todo correcto ya que para eso se guardan de forma temporal ciertos archivos en el caché, pero para que entiendas el funcionamiento general sirve.

Estructura de los logs:

Volviendo un poco a la parte teórica de lo que es un log, vamos a desgarnar que es lo que realmente cuenta cada log o línea de código:

```
66.249.66.71 www.xxxxxx.com - - yy [01/Jan/2016:12:18:08 +0100] 11852 "GET /directorio2/ficha4_5741255.htm HTTP/1.1" 301 - "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" -"
```

- **66.294.66.71** IP del cliente desde donde se hace la petición y a donde se está realizando la petición.
- **[01/Jan/2016:12:18:08]** Fecha y hora la que se realiza la petición al servidor.
- **GET** Método utilizado para requerir la información.
- **/directorio2/ficha4_5741255.htm** URL a la que se ha realizado la petición y el formato de este archivo (en este caso .htm)
- **HTTP/1.1** Protocolo con el que se ha realizado la petición.
- **301** Código de respuesta del servidor. En este caso es una redirección 301.
- **Mozilla/5.0 Googlebot 2.1** User Agent que ha realizado la consulta. Es decir, que bot o persona a realizado la petición al servidor. En este caso es Google Bot 2.1 desde un navegador Mozilla 5.0.

¿Qué son las **migraciones** y para qué se usan?

La migración de datos consiste en la transferencia de materiales digitales de un origen de datos a otro, transformando la forma lógica del ente digital de modo que el objeto conceptual pueda ser restituído o presentado por un nuevo equipo o programa informático.

Los datos pueden ser generados por múltiples aplicaciones de software, almacenados en diversos medios como archivos, servidores o bases de datos, y además intervenir en varios procesos de negocio, así que la necesidad de transferir y convertir los datos puede ser impulsada por múltiples requerimientos y el enfoque adoptado para la migración depende de esos requisitos. Sobre esta base se proponen cuatro tipos principales de migración:

- Migración de base de datos
- Migración de almacenamiento
- Migración de aplicación
- Migración de proceso de negocio

¿Cómo trabaja **Laravel** con la validación de formularios?

Laravel 5 incluye un trait que se carga en el controlador base (clase Controller), llamado "ValidatesRequests". Ese trait contiene código que estará disponible en todos los controladores que nosotros creamos, puesto que todos extienden la clase Controller.

Dentro del trait "ValidatesRequests" hay un método que nos sirve para hacer validaciones de los datos que nos llegan mediante HTTP Request, llamado validate(). Por tanto, en todos nuestros controladores podremos invocar este método para realizar validaciones de datos de entrada.

El método validate() recibe dos parámetros:

1. El objeto Request
2. Las reglas de validación que queramos definir.

Las reglas de validación son diversas y perfectamente configurables por nosotros mediante una sencilla sintaxis.

¿Cómo crear y ejecutar tests desde **Laravel**?

Laravel 5.1, además de incluir soporte para PHPUnit, cuenta también con su propio componente para que los desarrolladores puedan crear sus propias pruebas unitarias y de integración. Estas pruebas escritas por medio de código son medidas que acreditarán que todo en nuestra plataforma funciona correctamente.

Primero que nada, debes saber que dentro de tu proyecto de Laravel 5.1 se encuentra una carpeta llamada test en donde se ubicarán todas las pruebas de tu aplicación. Las pruebas creadas deben de extender de la clase TestCase.

Para realizar nuestras pruebas lo que tenemos que hacer es crear nuestra clase o bien utilizar ExampleTest que viene por defecto en el directorio test/.

En consola colocamos phpunit para así ejecutar todos los test que tenemos en nuestro directorio test/.

¿Qué es lo que más te ha sorprendido de Laravel?

La cantidad de herramientas que tiene integradas en un mismo framework.

¿Qué te parece la documentación de Laravel? ¿Crees que es útil para poder trabajar y aprender?

La documentación de Laravel es muy completa. Consta de explicaciones y ejemplos que ayudan a entender su funcionamiento, así como un buscador eficaz que facilita la labor de encontrar lo que queremos buscar y tiene una estructura clara y concisa.

Estas características hacen de la documentación de Laravel una buena herramienta para trabajar y aprender.

Investiga acerca del paquete “**Faker\Generator**” y explica brevemente para qué se usa

Faker es una biblioteca PHP que genera datos falsos. Ya sea que necesites arrancar una base de datos, crear documentos XML atractivos, completar su persistencia para realizar pruebas de resistencia o anonimizar los datos tomados de un servicio de producción.

Tiene especial utilidad para dejar atrás la tediosa tarea de introducir manualmente los datos de la aplicación cuando queremos probar nuestro código.

1.2. PARTE PRÁCTICA

1.2.1. INSTALACIÓN

```
PS C:\Users\ACER-1\Documents\PILLS\Laraver_pill> composer global require laravel/installer
Changed current directory to C:/Users/ACER-1/AppData/Roaming/Composer
Using version ^3.0 for laravel/installer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Nothing to install or update
Generating autoload files
PS C:\Users\ACER-1\Documents\PILLS\Laraver_pill>
```

```
PS C:\Users\ACER-1\Documents\PILLS\Laraver_pill> composer create-project --prefer-dist laravel/laravel blog
Installing laravel/laravel (v6.12.0)
- Installing laravel/laravel (v6.12.0): Downloading (connecting)
Downloading (100%)
Created project in blog
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 85 installs, 0 updates, 0 removals
- Installing symfony/polyfill-ctype (v1.14.0): Loading from cache
- Installing phpoption/phpoption (1.7.2): Downloading (connecting)
Downloading (100%)
```

Instalación de extensiones

1) Descargar los archivos .dll

Disco local (C:) > Usuarios > ACER-1 > Descargas

Buscar en Descargas


Nombre

Fecha de modificación

Tipo

Tamaño


▼ hoy (4)



Sin confirmar 386330.crdownload

25/02/2020 10:22


Archivo CRDOWN...



json

25/02/2020 10:21


Archivo WinRAR ZIP



php_ctype.dll

25/02/2020 10:20

Archivo WinRAR ZIP



php_tokenizer.dll









25/02/2020 10:19

Archivo WinRAR ZIP

2) Copiar el archivo dll en la carpeta ext de php

C:\wamp64\bin\php\php7.2.18\ext

Buscar en ext

Nombre	Fecha de modificación	Tipo	Tam
 php_ftp.dll	01/05/2019 1:52	Extensión de la ap...	
 php_gd2.dll	01/05/2019 1:52	Extensión de la ap...	
 php_gettext.dll	01/05/2019 1:52	Extensión de la ap...	
 php_gmp.dll	01/05/2019 1:52	Extensión de la ap...	
 php_imap.dll	01/05/2019 1:52	Extensión de la ap...	
 php_interbase.dll	01/05/2019 1:52	Extensión de la ap...	
 php_intl.dll	01/05/2019 1:52	Extensión de la ap...	
 php_json.dll	25/02/2020 10:22	Extensión de la ap...	

3) Habilitar .dll en el archivo php.ini

```
*php: Bloc de notas
Archivo Edición Formato Ver Ayuda
;extension=pgsql
;extension=phpdbg_webhelper
;extension=shmop
;extension=php_tokenizer
;extension=php_json
;extension=php_ctype
; The MIBS data available in the PHP distribution must be installed.
; See http://www.php.net/manual/en/snmp.installation.php
;extension=snmp
```

1.2.2.1. CREAR UN VIRTUAL HOSTS

<https://victorrobesweb.es/2016/03/26/crear-varios-hosts-virtuales-en-wampserver/>

1.2.2.2. ERRORES LOG

Se encuentran en el directorio : /storage/logs/laravel.log

```
# laravel.log
storage > logs > laravel.log
1 [2020-02-26 08:51:00] local.ERROR: Command "server" is not defined.
2
3 Did you mean one of these?
4 make:observer
5 serve {"exception":"[object] (Symfony\\Component\\Console\\Exception\\CommandNotFoundException(code: 0): Command \"server\" is not defined.)
6
7 Did you mean one of these?
8 make:observer
9 serve at C:\\Users\\ACER-1\\Documents\\PILLS\\laravel_pill\\vendor\\symfony\\console\\Application.php(670)
10 [stacktrace]
11 #0 C:\\Users\\ACER-1\\Documents\\PILLS\\laravel_pill\\vendor\\symfony\\console\\Application.php(236): Symfony\\Component\\Console\\Application->fin
12 #1 C:\\Users\\ACER-1\\Documents\\PILLS\\laravel_pill\\vendor\\symfony\\console\\Application.php(148): Symfony\\Component\\Console\\Application->doR
13 #2 C:\\Users\\ACER-1\\Documents\\PILLS\\laravel_pill\\vendor\\laravel\\framework\\src\\Illuminate\\Console\\Application.php(93): Symfony\\Component
14 #3 C:\\Users\\ACER-1\\Documents\\PILLS\\laravel_pill\\vendor\\laravel\\framework\\src\\Illuminate\\Foundation\\Console\\Kernel.php(131): Illuminate
15 #4 C:\\Users\\ACER-1\\Documents\\PILLS\\laravel_pill\\artisan(37): Illuminate\\Foundation\\Console\\Kernel->handle(Object(Symfony\\Component\\Conso
16 #5 {main}
17 }
18
```


1.2.8. TESTING CON LARAVEL

Test ejecutado:

```
class ExampleTest extends TestCase
{
    /**
     * A basic test example.
     *
     * @return void
     */
    public function testBasicTest()
    {
        $response = $this->get('/');

        $response->assertStatus(200);
    }
}
```

Caso correcto:

```
Route::get('/', 'MainController@welcome');

PS C:\Users\ACER-1\Documents\PILLS\laravel_pill> vendor/bin/phpunit
PHPUnit 8.5.2 by Sebastian Bergmann and contributors.

..                                                    2 / 2 (100%)

Time: 2.96 seconds, Memory: 18.00 MB

OK (2 tests, 2 assertions)
PS C:\Users\ACER-1\Documents\PILLS\laravel_pill>
```

Caso incorrecto:

```
Route::get('/welcome', 'MainController@welcome');
```

There was 1 failure:

```
1) Tests\Feature\ExampleTest::testBasicTest
Expected status code 200 but received 404.
Failed asserting that false is true.

C:\Users\ACER-1\Documents\PILLS\laravel_pill\vendor\laravel\framework\src\Illuminate\Foundation\Testing\TestResponse.php:185
C:\Users\ACER-1\Documents\PILLS\laravel_pill\tests\Feature\ExampleTest.php:19

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
PS C:\Users\ACER-1\Documents\PILLS\laravel_pill>
```

2. ORGANIZACIÓN DE LA PÍLDORA

Documentación de requisitos

- Debes de usar GIT. Es importante que las indicaciones y los commits sean los suficientemente explícitos y concretos como para poder entender los cambios sin la necesidad de requerir de información adicional en la medida de los posible.
- Crear una estructura de directorios clara y ordenada
- Tanto el código como los comentarios tienen que estar escritos en inglés
- Usa el estilo de código camelCase para la definición de variables y funciones
- En el caso de estar usando HTML nunca uses estilos en línea
- En el caso de estar usando diferentes lenguajes de programación siempre define la implementación en términos separados
- Recuerda que es importante dividir las tareas en varias sub-tareas para que de esta forma puedas asociar cada paso en particular de la construcción con un commit específico
- Debes intentar en la medida de lo posible que los commits y las tareas planificadas sean lo mismo
- Borra los ficheros que no se usen o no sean necesarios para evaluar el proyecto

Lista de tareas a realizar

Título y descripción	Prioridad	Dificultad	Tiempo
Investigación	2	1	3h
Instalación	5	1	5'
Primeros pasos con Laravel	4	2	30'
Sistema de logs	4	3	1h
Rutas, vistas, controladores y validación de formularios	4	3	1h
Trabajando con la base de datos, las migraciones y Eloquent	4	3	1h
Recuperando los contenidos de tu base de datos	3	3	30'
Desarrolla un pequeño backend para añadir nuevos artículos	3	2	45'
Testing con Laravel	1	2	15'
Vue & Laravel	2	2	30'

Listado de las herramientas usadas

- HTML
- CSS
- JS
- PHP
- Laravel
- BOOTSTRAP
- Npm

- Composer
- VueJs
- Dependencias
 - axios
 - bootstrap
 - cross-env
 - jquery
 - laravel-mix
 - lodash
 - popper.js
 - resolve-url-loader
 - sass
 - sass-loader
 - vue
 - vue-template-compiler