Mansori Amar Sara

smansori@uoc.edu

Abstract

Desarrollo de web de noticias utilizando el framework de PHP Laravel

PEC 3 – dESARROLLO BACKEND CON PHP

LARAVEL

# Contenidos

[1. Contenidos 1](#_Toc105781453)

[2. Instalación de Laravel de forma local 2](#_Toc105781454)

[2.1 Instalación a través de composer 2](#_Toc105781455)

[2.2 Comparativa entre Laravel y CodeIgniter o gestores de contenidos 2](#_Toc105781456)

[3. Migraciones y modelos 2](#_Toc105781457)

[3.1 Crear la base de datos 2](#_Toc105781458)

[3.2 Creación de migraciones y modelos 2](#_Toc105781459)

[3.3 Crear elementos con Tinker 6](#_Toc105781460)

[4. Paso 3 6](#_Toc105781461)

# Instalación de Laravel de forma local

## Instalación a través de composer

Tras comprobar que el equipo tiene tanto PHP como Composer instalados, se pueden crear proyectos de Laravel utilizando el CLI de Artisan, e inicializar el servidor local de desarrollo con el comando “serve”:

“composer create-project laravel/laravel name-of-the-app”

“php artisan serve” -> Hace que esté disponible en http://localhost:8000

## Comparativa entre Laravel y CodeIgniter o gestores de contenidos

Laravel me ha parecido más parecido a CodeIgniter que a gestores de contenidos como Wordpress o Laravel, ya que no es tan opinionado como estos y permite una mayor capacidad de customización (además de que ambos siguen el Modelo Vista Controlador).

Por un lado, la parte positiva de Laravel es este nivel de customización o capacidad de desarrollar al gusto del programador, aunque por otro lado también se hace más complejo debido a que debe hacerse por linea de comandos y requiere cierto conocimiento de programación.

En cuanto a las diferencias entre Laravel y CodeIgniter en términos de su instalación, me ha sorprendido la cantidad de opciones disponibles en Laravel, pudiendo instalarse de forma local o dockerizada, y habiendo diferentes maneras de hacerlo dentro de cada una de las opciones mencionadas, lo que me hace considerar que Laravel se pueda adaptar mejor a las necesidades de cada desarrollador.

# Migraciones y modelos

## Crear la base de datos

En primer lugar deberemos crear la base de datos (en nuestro caso se llamará news) en la que posteriormente añadiremos las tablas, filas y columnas necesarias. Para eso, podemos hacerlo desde la interfaz gráfica o inicializando desde la terminal SQL y usando los siguientes comandos:

* mysql -uroot -p
* create database news

## Creación de migraciones y modelos

Una vez la base de datos esté creada y corroboremos que tiene el mismo nombre que la base de datos referenciadas en el archivo de entorno (.env) de nuestro proyecto de Laravel, necesitaremos crear tres migraciones y dos modelos para el ejercicio propuesto: modelo Article con migración create\_article\_table, modelo Category con migración create\_category\_table y migración para crear una tabla que albergue las relaciones many to many entre Category y Article.

Para crear estas migraciones en Laravel, podemos ejecutar los comandos del php artisan por separado, o crear un modelo y una migración relacionada usando un flag.

* php artisan make:model Article -m
* php artisan make:model Article -m
* php artisan make:migration create\_article\_category\_table

Tras crear estos archivos, derebemos realizar ciertas modificaciones para que reflejen los campos y valores que queremos que tenga nuestra base de datos.

En las migraciones (que estarán en el directorio “database/migrations”) añadiremos en la función que se le pasa como segundo parámetro al método create() del Schema, todas las columnas que queremos que tenga nuestra tabla, indicando el tipo de columna en base a los datos que va a albergar (basándonos en los [tipos de columna de Laravel](https://laravel.com/docs/9.x/migrations#available-column-types)).

Text

Description automatically generated

Figure . "create-categories-table"

Text

Description automatically generated

Figure . "create-articles-table"

Text

Description automatically generated

Figure . "create-article-category-table"

En los modelos (que están en el directorio “App/Models/”, teniendo en cuenta que la relación entre los modelos es de many to many (un artículo puede tener muchas categorías y una categoría pertenecer a muchos artículos), desarrollaremos un método para cada modelo que nos permita acceder a las categorías desde el modelo Article y a los artículos que contengan la categoría del modelo Category.

También podemos introducir una variable llamada $fillable que indica qué valores pueden ser asignados a través de lo que se conoce como “mass assignment” o asignación masiva, debido a que esta forma de asignar valores puede presentar vulnerabilidades, esto previene que se asignen campos que no está previsto que cambien).

Text

Description automatically generated

Figure . Model\Article

Text

Description automatically generated

Figure . Model\Category

Para enlazar la base de datos creadas con las migraciones y modelos creados podemos utilizar el comando:

* php artisan migrate

Y eso creara en nuestra base de datos todas las columnas con los valores definidos en las migraciones.

## Crear elementos con Tinker

Para comprobar que nuestros modelosy migraciones están correctamente creados, podemos utilizar una herramienta llamada Tinker que nos permite acceder al modelo y crear elementos a través de la terminal.

Para ello tendremos que introducir el comando: php artisan tinker

Una vez que estemos dentro del modo de tinker podemos utilizar comandos como los siguientes para popular nuestra base de datos:

* $article = new App\Models\Article; (guardamos una instancia del modelo Article)
* $article->title=’Título de la noticia’; (le asignamos los diferentes valores a las columnas declaradas en la migración)
* $article->author=’Anónimo’;
* $article->content=’Lorem ipsum dolor sit amet.
* $article->save()’ (guardamos la fila en la base de datos)

En caso de que quisiéramos asignarle alguna categoría, tenemos que tener en cuenta la existencia de la pivot table que relaciona los Article con las Category. Para ello haríamos lo siguiente (a continuación del comando anterior o seleccionando el artículo de la base de datos al que le queremos asignar una categoría con por ejemplo Article::find(id):

* $article->categories()->attach(category\_id) -> si queremos asignarle una categoría
* $article->categories()->attach( [ category1\_id, category2\_id… ] ) -> si queremos asignarle varias categorías a la vez

# Generar contenido ficticios

Para generar contenidos ficticios se usará un Factory y Faker (que ya viene instalado con Laravel). En este caso, se creará una Factory para Article puesto que será el modelo del que se necesiten crear contenidos ficticios, mientras que la creación de los elementos del modelo Category se realizará directamente desde el fichero del seed.

## Creación de los Factory

Podemos crear un archivo Factory utilizando el comando de artisan:

* Php artisan make:factory Article

Una vez creado (se guardará en el directorio “database/factories”, deberemos modificarlo de la siguiente forma para indicar que elementos queremos que se creen y de qué forma cuando se llame a la Factory:

Text

Description automatically generated

Figure . ArticleFactory

Como se puede observara en el código, se ha usado la dependencia faker para generar contenidos ficticios, a excepción de la imagen, puesto que parece que la funcionalidad de faker de imágenes ya no está operativa y se ha optado por usar como alternativa la página de picsum, que genera imágenes aleatorias cada vez que se accede a ella. Para que las imágenes de los artículos no fuesen la misma cada vez que se cargaba la página, se guardan en la base de datos cada una con un número aleatorio del 0 al 100.

## Creación de los seeder

Ahora que tenemos los Factory se puede proceder a desarrollar los archivos de seed para sembrar nuestra base de datos.

Dentro del directorio “database/seeders” podemos ver un fichero: “DatabaseSeeder.php”. Este es el fichero que tendremos que modificar para indicarle cómo queremos popular nuestra base de datos cada vez que se ejecute el comando seed.

Text

Description automatically generated

Figure . DatabaseSeeder.php

En este archivo, dentro del método run(), introduciremos la creación de las diferentes categorías que queramos (similar a cómo lo hicimos anteriormente con Tinker).

Para los artículos, sin embargo, utilizaremos la Factory creada, llamando al modelo Article, y al método factory, pasándole como parámetro un integer con la cantidad de entradas que queremos generar e indicándole que los cree con el método create().

También queremos asignar de manera aleatoria una o varias categorías a cada artículo creado, así que una vez se han almacenado todos los artículos creados y las categorías en las variables correspondientes $articles y $categories, iteramos por cada uno de los artículos y llamamos al método categories() para a su vez llamar al método attach() y asignarle el id de categorías aleatorias.

## Generación de datos ficticios

Tras esto, para generar los datos ficticios nos dirigimos a la terminal, desde donde podemos usar el siguiente comando para realizar de nuevo las migraciones necesarias, además de limpiar las anteriores y hacer seed de nuestro archivo DatabaseSeeder.php para generar los contenidos ahí indicados:

* php artisan migrate:fresh –seed

# Impementación del frontend

Para desarrollar el frontend de la aplicación se tendrán que desarrollar dos cosas principalmente: las rutas y las vistas.

Las rutas las podemos encontrar dentro del directorio “routes”, y estas en concreto se configurarán dentro del archivo “web.php”

* **Listado de noticias con su título, imagen y fecha:**

Es la página principal a la que se accede nada más acceder a la página. Esto significa que le corresponde la ruta “/”.

* **Página única de noticias con sus campos:**
* **Página de categoría:**

1. Instalación de Laravel de forma local