Routes: Rutas en Laravel

Cómo ejecutar mi aplicación Laravel

Opción 1: artisan
Ya vimos que con **`php artisan serve`** podíamos mostrar nuestra aplicación en un browser. Simulaba un servidor, y abría la aplicación en el localhost:8000

Opción 2: Homestead
Es un entorno virtual para utilizar Laravel, mucho más complicado. Requiere un Virtual Box u otra herramienta de virtualización.
Ofrece que los proyectos tengan una extensión, es decir, mi proyecto sería accesible desde un navegador con la URL aplicacion1.dev
"dev" sería la extensión, podría haber varias, como "test", etc

A efectos prácticos, utilizaremos **artisan**.

Cómo definir las rutas de mi aplicación en Laravel

Actualmente, las routes (clases de Laravel) estarán en una **carpeta routes** colgando de la raíz de mi aplicación, y se definen en un fichero **web.php**
En versiones anteriores de Laravel, colgaban de app, y se definían en un fichero routes.php

Dentro de routes tendremos varios php donde podremos definir las diversas rutas.

Estructura de un proyecto Laravel:

Carpetas:

- **app**: Carpeta principal, donde tendremos nuestros **controladores y modelos**
- **bootstrap**: Arranque de la aplicación
- **config**: Configuración de todo, conexión a BD, mailing, auth, etc
- **database**: Aquí creamos la base de datos, las tablas, nuestras factorías, etc
- **public**: nuestro index.php, y el resto de mi aplicación
- **resources**: Podré varios ficheros en un js, un sass, etc. Dentro está la carpeta **views**, con mi html, css, etc.
- **routes**: Donde están localizadas las rutas. Aquí crearé las rutas para mi aplicación.
- **tests:** Carpeta para las pruebas unitarias
- **storage:** Contiene los ficheros generados por Laravel
- **vendor**: Donde están los fuentes de todas mis dependencias, incluido Laravel

Ficheros importantes:

1. **.env**, con información sobre constantes de entorno, con datos del sistema. No debemos subirlo a git (ver .gitignore), y debe ser oculto.
2. También tendré el **programa artisan**
3. **composer.json**: Por supuesto, con phpunit en require-dev

Introducción a Route

Ruta: La URL que pones en el navegador.

Fichero donde se registran las rutas web de mi aplicación:

En versiones antiguas de Laravel, se utilizaba el fichero routes.php.
En versiones actuales utilizaremos el fichero

```
./routes/web.php
```

Cómo definir una ruta

Una ruta se define con el método get de la clase Route, que tiene dos parámetros:

1. El nombre de la ruta, es lo que ponemos en el navegador, el request que nos hacen
2. Una función closure. Su return corresponde al response.

Ejemplo 1:

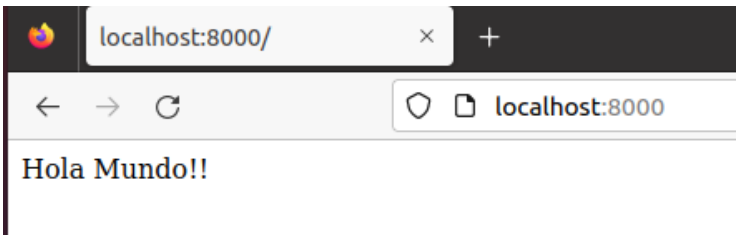
```
Route::get('/', function () {
    return view('welcome');
});
```

En el ejemplo, el método estático **get**, de la clase **Route**, hará que se ejecute la función en el caso de que la URL tecleada sea “/” (el raiz de mi aplicación).

La función es un closure, y retorna la vista llamada “welcome”, siendo una vista el HTML que se enviará en el response. Podría retornar cualquier cosa, por ejemplo un texto, u otro HTML que yo quiera, o simplemente hacer un echo.

Ejemplo 2:

```
Route::get('/', function () {
    //return view('welcome');
    return "Hola Mundo!!";
});
```



Si la ruta tecleada no está definida como ruta, entonces saldrá un error indicando que la página no existe (440).

Si la ruta tecleada está definida como ruta, saldrá la vista indicada en la función.

La ruta no tiene por qué corresponder con un directorio real. De hecho, podrá no existir ningún directorio real con ese nombre.

Ejemplo 3:

```
Route::get('/palabraNoDirectorio', function () {
    return view('welcome');
});
```

Podemos añadir tantas rutas como queramos, llamando de nuevo al método get.

Variables en las rutas

También podemos poner una variable como subruta, poniéndola entre llaves. Dicha variable será parámetro de la función closure.

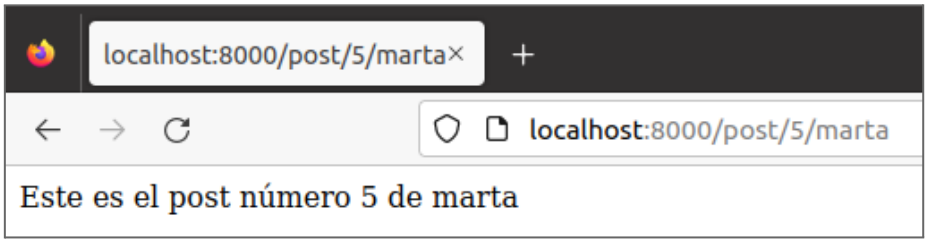
Ejemplo 1:

```
Route::get('/post/{id}', function ($id) {
    return "Este es el post número $id";
});
```

Ejemplo 2:

También puedo utilizar varios parámetros, poniendo todos como parámetro de la función closure.

```
Route::get('/post/{id}/{nombre}', function ($id,$nombre) {
    return "Este es el post número $id de $nombre";
});
```

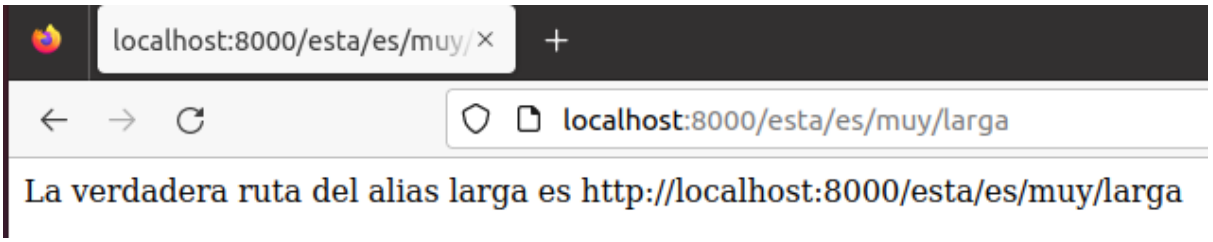


También podremos pasar esos parámetros a las vistas, ya veremos cómo.

Cómo nombrar las rutas con un alias

Podemos definir un alias para rutas largas. Ese alias podrá utilizarlo la aplicación en los blades (plantillas de Laravel). La forma de crear un alias es la siguiente:

```
Route::get('/esta/es/muy/larga', function(){
    $url=route('larga');
    return "La verdadera ruta del alias larga es $url";
})->name('larga');
```



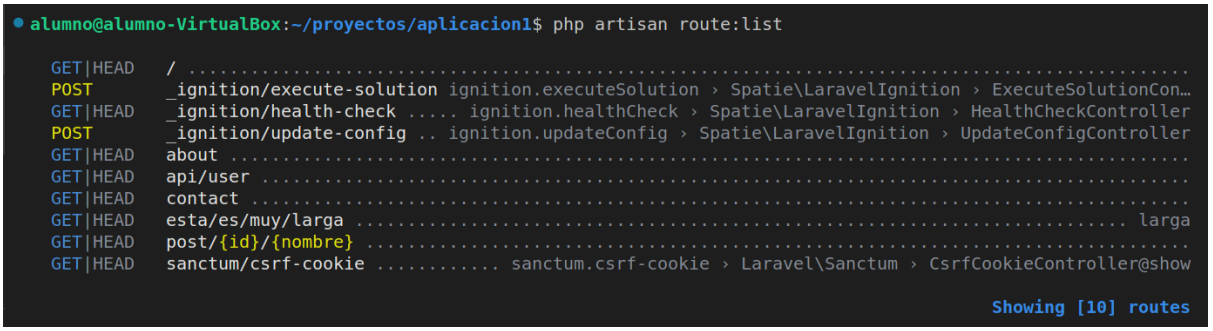
El nombre de un alias es único.

NOTA IMPORTANTE: El alias (name) no puede utilizarse directamente en la URL, si vamos a acceder desde un navegador, tendremos que escribir la ruta larga. En cambio, sí podemos utilizarlo en nuestras vistas.

Cómo visualizar las rutas de mi aplicación

Para visualizar las rutas de mi aplicación, puedo hacerlo con el comando:

```
$ php artisan route:list
```



Me mostrará el método del request, las rutas, y en su caso el alias a la derecha.