Creación de un proyecto con Laravel 10

Última actualización: 22/11/2023, en proceso

Índice de contenidos

Introducción

En esta práctica crearemos un proyecto desde cero con el framework Laravel (10.35 o superior) y PHP (8.2+). Los ejemplos utilizados se ejecutan sobre un Linux y una base de datos Postgresql 15+. La interacción con el cliente usa Ajax mayormente con Jquery, sin descartar algunos ejemplos con Javascript. Descartamos el uso de frameworks de cliente como Vue.js, Angular o React.

Como el objetivo es fundamentalmente didáctico, presentaremos ejemplos propios de una SPA (Single-Page Application), pero también ejemplos en los que se regenere completamente la página, a pesar de no ser estrictamente necesario.

En la medida de los posible seguiremos estos pasos válidos para la mayoría de proyectos, cambiando solamente la 'lógica de negocio'.

El proyecto incluye:

- Autenticación de usuarios de Laravel, que adaptaremos a nuestra aplicación.
- Autorización con Laravel.
- Responsividad con Bootstrap 5
- ApiRest con JSON en peticiones y respuestas. Uso de ApiKey.
- Soporte multilingüe.
- JWT, en versiones posteriores.

Preparando el entorno

Asumimos como nombre de proyecto **cmd-laravel**, pero puedes sustituirlo por otro.

Creación de una base de datos. Software administrador de la B.D (Adminer 4.8.1 o superior)

Recomendable crear la base de datos en un servidor remoto y que las versiones de la aplicación, remota y local, accedan a la misma B.D.

La B.D. se denominará igual que el proyecto, seguido del sufijo '_bd', p.e.,

cmd-laravel_bd.

El propietario de la B.D. tendrá el nombre de proyecto, seguido del sufijo '_usr', p.e.,

cmd-laravel_usr.

Desde *adminer* (u otra herramienta) creamos una b.d. vacía.

Creación de un proyecto Laravel base.

Crearemos el proyecto vía composer create-project

composer create-project laravel/laravel *nombre_del_proyecto*

A continuación lo siguiente:

1. Cambia el propietario y grupo del proyecto al usuario y grupo ***www-data,*** y hazte miembro de ese grupo. (desde la raíz del proyecto)

> $ **chown -R www-data:www-data cmd-laravel**
>
> **Explicación:** Cuando creas el proyecto eres el propietario de todos sus subdirectorios, pero Apache se ejecuta como usuario www-data y grupo www-data y necesita escribir, entre otros, en los subdirectorios
>
> storage/logs, storage/framework/sessions y storage/framework/views
>
> Puedes optar por dar permiso a todos haciendo, por ejemplo (no recomendable)
>
> $ **chmod -R o+w storage/**
> $ **chmod -R o+w bootstrap/cache/**
>
> pero es más seguro cambiar el propietario de los siguientes directorios del proyecto.
> $ **chown -R www-data:www-data storage/**
> $ **chown -R www-data:www-data bootstrap/cache**

2. Realizar los siguiente cambios en el fichero **.env** de la raíz del proyecto:
    1. Configurar el acceso a la B.D actualizando las constantes DB_* con los valores adecuados a la b.d. creada anteriormente.
    2. Comprobar que la APP_KEY fue creada por composer
    3. Configurar las constantes de .env APP_NAME y APP_URL.
3. Revisar el fichero config/app.php para adaptarlo a nuestra aplicación, en particular los siguientes items: *name*, *url*, *timezone*(1) 'Europe/Madrid' y *locale (es)*. (*name* y *url* intentará leerlos del fichero **.env**)

Notas:
(1) Para las zonas horarias admitidas consultar

> https://www.php.net/manual/es/timezones.php

Referencias:
https://laravel.com/docs/10.x/installation

Configuración de hosting virtual (Apache2) en local

Paso 1.- Modificar el fichero de ***hosts*** para que reconozca el site local, p.e., cmd-laravel.local como dirección url del proyecto.

```
127.0.0.1     localhost
127.0.0.1     ninja.local
127.0.0.1     laravel.local
127.0.0.1     cmd-laravel.local
```

Comprobación: ejecuta

ping cmd-laravel.local

Observación: Si no hay respuesta, es posible que tengas filtrados los paquetes icmp (ping es un tipo de mensaje icmp)

Paso 2.- Modificar en Apache los *sites-enabled,* fichero ***000-default.conf;*** añadiremos un hosting virtual. Para ello, configuraremos nuestro servidor con el ***DocumentRoot*** apuntando a la carpeta ***public*** de nuestro proyecto (que es donde se encuentra el index.php).

```
<VirtualHost *:80>
        ServerName cmd-laravel.local
        DocumentRoot /var/www/cmd-laravel/public
</VirtualHost>
```

Habilitar https en nuestro servidor Apache, si aún no lo está.

Como root

a2enmod ssl
#ln -s /*etc*/apache2/sites-available/default-ssl.conf /*etc/apache2/sites-enabled/000-default-ssl.conf*

Con lo anterior ya tenemos https, pero si queremos utilizar certificados firmados por una autoridad certificadora reconocida, **letsencrypt.org** nos facilitará la tarea por medio de 'certbot', que es específico para un servidor web y sistema operativo concreto.

Pretty url y problemas habituales que impiden su correcto funcionamiento

Para que el servidor Apache2 cargue el ***.htaccess*** que está en la carpeta ***public*** del framework, añadiremos el siguiente código en el fichero /etc/apache2/apache2.conf

```
<Directory /var/www/cmd-laravel>
        Options Indexes FollowsymLinks
        AllowOverride  All          ←Esta es la línea que lo permite. Normalmente configurada a None
        Require all granted
</Directory>
```

El .htaccess de laravel contiene reglas de reescritura para que no sea necesario indicar en la url el fichero index.php; ni la extensión '.php', para que se ejecute un fichero php.

Importante. Asegúrate de que el módulo rewrite esté cargado en el apache, o no se leerá el fichero .htaccess

Para comprobar si el módulo está cargado

```
sudo apache2ctl -M | grep rewrite
```

Debes ver

```
rewrite_module(shared)
```

en casos contrario tienes que cargar el módulo

```
sudo a2enmod rewrite
```

Creación de hosting virtual en remoto (Ionos o similar).

Necesitamos una cuenta en un proveedor en donde instalar una máquina que ejecute nuestra aplicación. No recomiendo un hosting virtual por internet (es preferible una

máquina virtual, debido a que las opciones de B.D. lenguajes etc. son más limitadas y puede que no trabajen con los productos que nos interesan.

Crear un nombre de dominio amigable, p.e., con *noip.com* (si fuese necesario) que apunte a esa máquina, p.e.

cmd-laravel.ddns.net

Instalación de nuestro IDE y registro de nuestro proyecto Laravel (Netbeans 15+, PHPStorm, etc.)

Configurar GIT, como sistema de control de versiones. Primer commit.

Evaluación de los pasos anteriores.

Configuraciones opcionales

Autenticación

Laravel 10 viene con dos kits de autenticación: Breeze y Jetstream.
Breeze es más sencillo de instalar, pero suficiente para la mayoría de nuestros proyectos.
Jetstream incluye autenticación de dos factores, creación de equipos de usuarios, etc.

Ver documentación oficial.
Getting started -> Starter Kits

Authentication

Observaciones:
Es recomendable hacer un 'git commit' antes de instalar un paquete, y otro commit al finalizar la instalación. De este modo podemos ver los cambios que se producen con la instalación de un nuevo paquete.

La instalación de jetstream modifica el fichero config/session.php, en concreto el driver de las 'sessions', cambia **File** por **Database**.

Envío de correos (en pruebas)

Reinicio de password y verificación de correo requiere que podamos enviar emails a través de un servidor smtp. Desde una cuenta gmail es posible siguiendo las indicaciones del enlace
https://medium.com/@agavitalis/how-to-send-an-email-in-laravel-using-gmail-smtp-server-53d962f01a0c

Nota: El enlace anterior tiene una omisión. Tenemos que editar también la entrada

MAIL_FROM_ADDRESS=randion@cifpfbmoll.eu
en el fichero '.env' con el email del emisor, p.e., el mismo que MAIL_USERNAME

Ejemplo de valores:

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.googlemail.com
MAIL_PORT=465
MAIL_USERNAME=randion@cifpfbmoll.eu
MAIL_PASSWORD=MiClave
MAIL_ENCRYPTION=ssl
MAIL_FROM_ADDRESS=randion@cifpfbmoll.eu
MAIL_FROM_NAME="${APP_NAME}"
```

Creación de formularios

Para cada formulario que creemos, realizaremos los siguientes pasos, no necesariamente ejecutados en este orden De hecho, haremos iteraciones entre ellos.

1. Recogida de información sobre el formulario que vamos a crear: recibos, pantallazos, facturas, entrevistas, consulta sobre BD existentes, etc.
2. Maquetado inicial del formulario. (Html5, css, bootstrap5+)
3. Creación de controles y el html necesario para su elaboración.
4. Crear las migraciones con los cambios a realizar en el esquema de la BD.
5. Añadir validaciones al formulario.
6. Tratamiento de errores desde el lado del servidor
7. Añadir reinserción de los datos para que el usuario no teclee de nuevo la información correcta, en caso de que la validación falle: {{old('..')}}
8. Añadir soporte multilingüe
9. Soporte para usuarios con discapacidad (de momento no incluirla)
10. Documentación del formulario.

Tutoriales sobre APIRest

Propuestos por alumnos

Laravel API Tutorial: Building & Testing a RESTful API (Versión de Laravel 5.4)

laravel-8-rest-api-crud-mysql/