INSTALACIÓN LARAVEL 9, WINDOWS

PRIMEROS PASOS, PROBLEMAS Y SOLUCIONES

LARAVEL

Es un Framework (estructura predefinida que facilita el desarrollo de aplicaciones. Proporciona un conjunto de herramientas, bibliotecas, y componentes reutilizables) MVC escrito en PHP.

CREAR EL PROYECTO

INSTALAR Docker Desktop

Docker es una plataforma de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software. Estos son entornos aislados que contienen todo lo necesario para ejecutar una aplicación: código, bibliotecas, dependencias...etc.

Docker Desktop facilita la creación, despliegue u administración de contenedores Docker.

Enlace de descarga Docker Desktop

INSTALAR WSL

Windows Subsystem for Linux, característica de Windows que permite ejecutar un entorno GNU/Linux sin necesidad de máquina virtual.

Enlace de descarga WSL

COMANDOS EN POWERSHELL

Establecer versión 2 como predeterminada:

- *wsl --set-default-version 2*

Instalar una distribución de Linux:

- *wsl --list --online*
- *Wsl --install -d <distribución a instalar>*

Posteriormente crear un nuevo usuario UNIX, solicitará introducir un usuario y una contraseña. Estos serán necesarios para distintas operaciones.

ABRIR SESIÓN CON WSL

Situados en WSL (escribir wsl en powershell o iniciar la app que se descarga al instalar la distribución anterior), navegar hasta la ubicación donde se quiere crear el proyecto. Ej: cd /Desktop.

Para crear el proyecto, siempre se usará el siguiente comando:

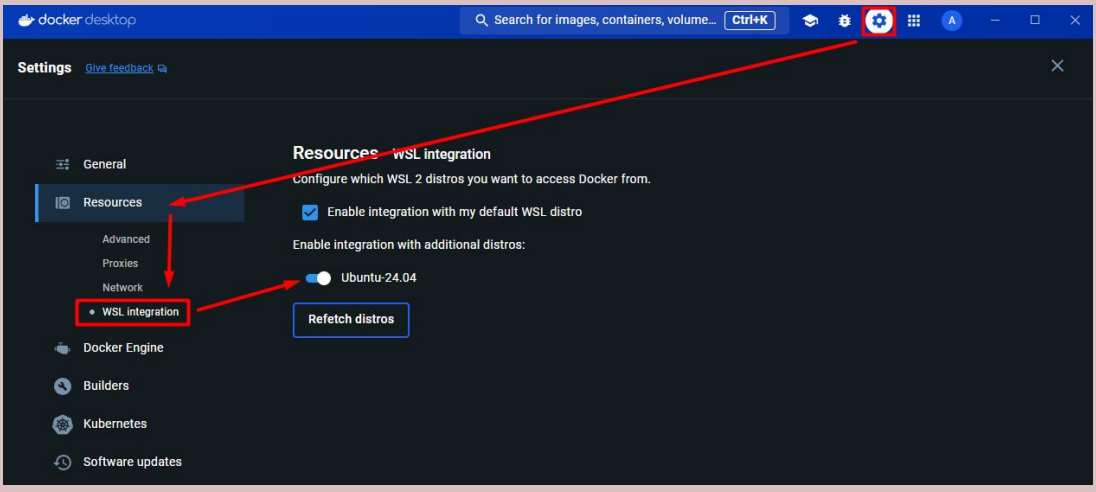
- ✧ *curl -s https://laravel.build/NombreDelProyecto | bash*

DOCKER NO ESTÁ HABILITADO

```
ana@Ani:~$ curl -s https://laravel.build/devstagram | bash
Docker is not running.
ana@Ani:~$ docker --version

The command 'docker' could not be found in this WSL 2 distro.
We recommend to activate the WSL integration in Docker Desktop settings.

For details about using Docker Desktop with WSL 2, visit:

https://docs.docker.com/go/wsl2/
```

SOLUCIÓN

Configurar Docker para integrarse con WSL, habilitando la integración con la distribución WSL existente.



USUARIO NO INTEGRADO EN GRUPO DOCKER

```
ana@Ani:~$ docker run hello-world
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head "http
://%2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
```

SOLUCIÓN

Agregar usuario a Grupo Docker

Crear el grupo (si no existe):

- *sudo groupadd docker*

Agregar tu usuario al grupo docker:

- *sudo usermod -aG docker $USER*

Actualizar la sesión para aplicar los cambios

CONFIGURACIÓN ARCHIVOS .env Y docker-compose.yml

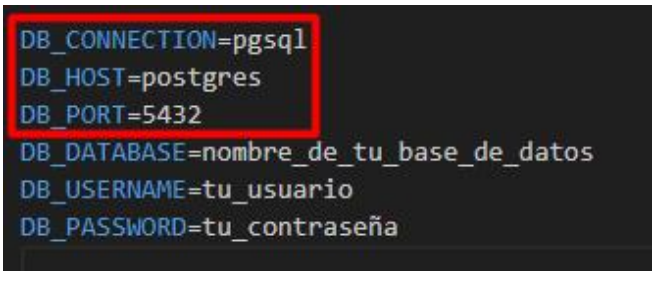
El archivo .env es crucial para la configuración segura de aplicaciones, en el se almacenan las variables de entorno que suelen incluir configuraciones sensibles como credenciales de base de datos, claves...etc.

Con este archivo se puede configurar la aplicación en diferentes entornos (producción, desarrollo, pruebas) sin tener que cambiar el código fuente.

Al principio se encuentra el archivo como **.env.example**. Esta es una plantilla para ver que configuraciones son necesarias sin tener acceso a datos reales. Es una buena práctica mantenerlo a modo de ejemplo.

A tener en cuenta que .env no es un archivo que se sube a sistemas de control de versiones como Git por los mismos motivos de seguridad que se han mencionado anteriormente.

CONFIGURACIÓN DE EJEMPLO PARA BASE DE DATOS POSTGRES

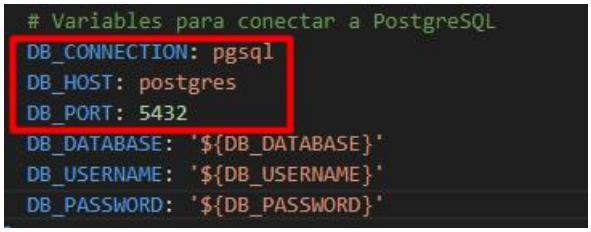


En el archivo **docker-compose.yml** se define que contenedores se van a utilizar y que contiene cada uno de ellos .

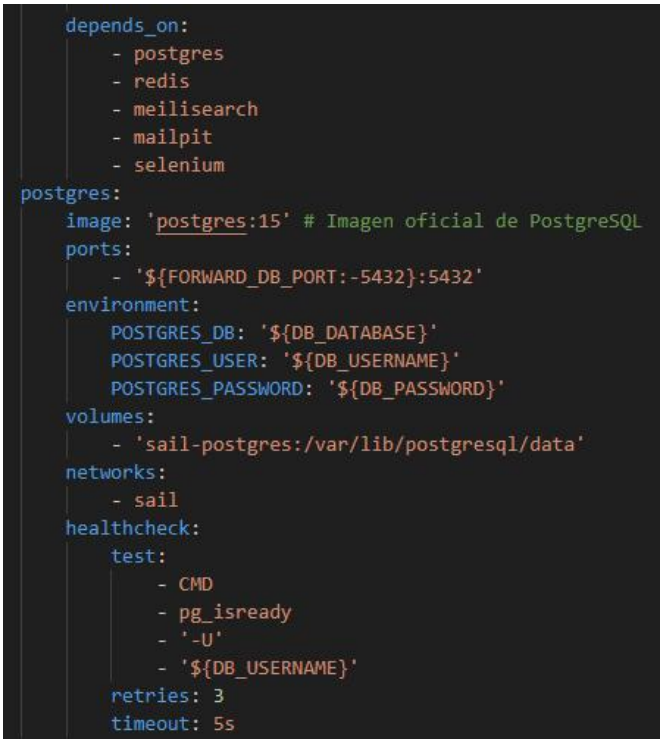
Contiene las versiones de la sintaxis utilizada, los servicios que forman parte de la aplicación, redes, volúmenes...etc.

La configuración de este archivo debe estar sincronizada con la del archivo .env.

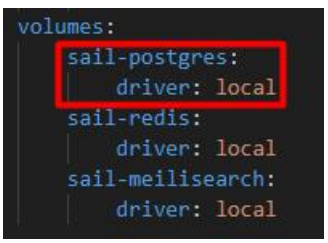
Por ejemplo, la configuración (dentro de docker-compose.yml) para el ejemplo anterior sería agregar el siguiente bloque de variables en: **services / laravel.test / environment**, a continuación de la última linea de código de este bloque.



El siguiente paso es configurar el servicio para ejecutar PostgreSQL. La configuración podría verse así:



Por último crear el volumen para la base de datos PostgreSQL:

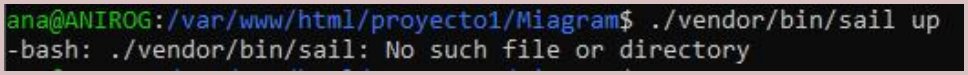


LEVANTAR SERVIDOR, INICIAR TODOS LOS SERVICIOS DE LARAVEL EN DOCKER

En la carpeta del proyecto, ejecutar el siguiente comando:

- *./vendor/bin/sail up*

DEPENDENCIAS DE COMPOSER NO INSTALADAS



En este caso, puede ser por dos motivos:

Composer no está correctamente instalado. En este caso ejecutar los siguientes comandos para instalar de nuevo las dependencias necesarias y verificar la presencia de Sail.

Sail es una herramienta de línea de comandos incluida en Laravel que permite un desarrollo sencillo con Docker sin necesidad de instalar servicios manualmente como PHP, MySQL....

- *composer install*
- *ls vendor/bin/sail*
- *./vendor/bin/sail up*

PHP no está instalado. Es la opción más probable si no se ha trabajado antes con PHP, ya que no suele estar instalado por defecto.

Para instalarlo, ejecutar los siguientes comandos:

- *sudo apt update*
- *apt-cache policy php-cli (Buscar versiones disponibles)*
- *sudo apt install php8.3-cli (última versión a Julio 2024)*
- *php -v*

Con PHP instalado, el siguiente paso sería instalar Composer con los pasos anteriores.

Es posible que aparezca un error debido a que no se han instalado las extensiones 'xml' y 'dom' de PHP, o no están correctas. Para solucionarlo, verificar la versión que se está usando de PHP, actualizar los paquetes e instalar las extensiones apropiadas a la versión que se está usando.

- *php -v*
- *sudo apt-get update*
- *sudo apt-get install php8.3-xml*

Solucionado el problema, entrar al contenedor de laravel:

- *docker exec -it proyecto-laravel.test-1 bash*

Ejecutar las migraciones (son clases que definen cómo se realiza una modificación en la estructura de la base de datos, creación, modificación o eliminación):

- *php artisan migrate*

Volver a levantar el servidor con el comando*./vendor/bin/sail up*

La web ya debería ser visible en localhost

ATAJOS

Agregar un alias para cualquier comando:

Dentro de la carpeta del proyecto ejecutar el siguiente comando para editar el archivo bashrc:

- *sudo nano ~/.bashrc*

Dentro de este archivo (mucha precaución con él), navegar hasta el final del mismo e introducir la siguiente línea:

- *alias="./vendor/bin/sail" (este es solo un ejemplo)*

Para refrescar los cambios:

- *source ~/.bashrc*

INSTALAR NODE.JS Y NPM

Aún que Laravel es un framework de PHP, es beneficioso usar Node.js y NPM ya que permiten integrar tecnología del lado del cliente de manera mucho mas sencilla y eficaz, además de facilitar el mantenimiento.

Node.js es un entorno de ejecución de **JavaScript** del lado del servidor, esto quiere decir que proporciona una plataforma para ejecutar programas escritos en JavaScript fuera del navegador web.

Existen otros lenguajes que permiten realizar las mismas acciones en el servidor, pero integrar JavaScript en todo el stack (cliente-servidor) es beneficioso por varios motivos: reduce la curva de aprendizaje, es ideal para web que requieren muchas acciones entrada/salida debido a su arquitectura no bloqueante y orientada a eventos que permite gestionar conexiones simultáneamente, escalabilidad...etc.

NPM es un sistema de gestión de paquetes para Node.js que facilita la re utilización de código. Es una herramienta para la gestión de librerías y módulos externos en un proyecto Node.js que permite instalar, actualizar y gestionar dependencias de software de forma eficiente.

Enlace de descarga Node.js

El proceso de instalación es simple, marcar siguiente hasta llegar a la casilla para la instalación de herramientas necesarias (se recomienda marcarla) y finalmente instalar.

Automáticamente se abrirá una ventana de PowerShell con la instalación de las herramientas y para verificar la versión instalada, tanto de Node.js como de NPM, se puede ejecutar el comando *node -v* y *npm -v*

MIGRACIONES

Como se ha mencionado anteriormente, las migraciones son clases que definen cómo se realizan los cambios en una base de datos. El uso de las migraciones permite que los cambios sean aplicados de manera ordenada y manteniendo el desarrollo en sincronía con diferentes entornos de desarrollo (producción, pruebas o desarrollo).

Permite tener un control de versiones, revertir cambios de forma sencilla (con el comando *php artisan migrate:rollback* o *php artisan migrate:rollback --step=4* para especificar cuantos pasos debe retroceder, entre otros comandos), automatizado de actualización de la base de datos

¿Como funcionan?

Cuando se ejecuta el comando php artisan migrate, Laravel busca las migraciones que aún no se han aplicado en la base de datos. Estas migraciones se ejecutan en el orden que fueron creadas, basándose en la fecha y hora de su archivo.

Cada migración modificada se registra en una tabla especial llamada 'migrations' en la base de datos, lo que previene que una misma migración se ejecute más de una vez.

Para realizar cambios en la base de datos, se debe crear un nuevo archivo de migración que contenga los cambios específicos y las instrucciones de cómo revertir esos cambios si es necesario.

Nombres de los archivos

El nombre de estos archivos es importante y sigue un patrón específico para asegurar el orden de ejecución.

Primero fecha y hora en formato *"aaaa_mm_dd_hhmmss"*, seguido de la descripción del cambio y finalmente la extensión *.php*.

Un ejemplo completo del nombre de un archivo de migración podría ser *2024_07_15_123456_create_users_table.php.* Este formato asegura que las migraciones se apliquen en el orden correcto y facilita la comprensión rápida de los cambios que cada archivo introduce en la base de datos.

SAIL

Laravel Sail es una herramienta de desarrollo basada en Docker que utiliza contenedores para asegurar que el entorno de desarrollo sea consistente y portátil en cualquier máquina.

Sail simplifica la configuración inicial de un nuevo proyecto Laravel al proporcionar servicios pre-configurados como bases de datos y servidores web. Ofrece comandos sencillos que facilitan la gestión de estos servicios.

Para utilizar Sail, es necesario tener Docker instalado, ya que Sail depende de contenedores para su funcionamiento. Sin embargo, Docker puede ser intensivo en recursos, y no todos los ordenadores pueden manejarlo eficientemente. En esos casos, aunque es posible trabajar sin Docker utilizando comandos alternativos, esto puede requerir más tiempo y esfuerzo para configurar y administrar manualmente el entorno de desarrollo.

Los comandos Sail siempre se ejecutan dentro del contenedor laravel.

Esto se debe a que cuando se ejecuta un comando con Sail, éste utiliza Docker Compose para determinar qué contenedor debe ejecutar el comando.
Esta configuración se debe realizar de forma manual si no se trabaja con Sail.