
Documentación del Proyecto Symfony con Docker

1. Requisitos Previos

Antes de empezar con el proyecto, asegúrate de tener instaladas las siguientes herramientas:

- **PHP 8.2+** o superior
 - **Symfony CLI**
 - **Docker y Docker Compose**
 - **Node.js** (si necesitas manejar tareas de frontend)
 - **Composer** para gestionar dependencias de PHP
 - **npm** para gestionar dependencias de frontend
-

2. Instalación del Proyecto

Clonar el Repositorio

Si aún no tienes el código del proyecto, clónalo desde tu repositorio Git:

```
git clone https://github.com/tu-usuario/tu-proyecto-symfony.git
cd tu-proyecto-symfony
```

Instalar las Dependencias Backend (PHP)

Usa Composer para instalar las dependencias PHP:

```
composer install
```

Instalar las Dependencias Frontend (Node.js)

Si estás usando Webpack o alguna otra herramienta de construcción de frontend:

```
npm install
```

3. Configuración del Entorno Local

Variables de Entorno

Crea un archivo `.env.local` para definir las variables de entorno locales específicas.

Cuando trabajas con proyectos Symfony, las variables de entorno se gestionan a través de archivos `.env`. Este archivo contiene configuraciones específicas que pueden cambiar según el entorno (local, producción, etc.), como la base de datos, el secreto de la aplicación, y otras configuraciones sensibles.

Pasos para configurar el entorno local:

1. Crear el archivo `.env.local`

En tu proyecto Symfony, existe un archivo `.env` que sirve como plantilla para las variables de entorno. Sin embargo, **para no afectar la configuración de producción** o de otros entornos, debes crear un archivo `.env.local` que sobrescriba las configuraciones locales de tu entorno de desarrollo.

En el directorio raíz de tu proyecto, crea el archivo `.env.local`. Puedes hacerlo manualmente o con el siguiente comando:

```
touch .env.local
```

2. Definir las variables de entorno

Dentro de este archivo `.env.local`, puedes definir las variables de entorno necesarias para el desarrollo local. Algunas de las variables más comunes para proyectos Symfony son:

```
# .env.local
APP_ENV=dev                # Define el entorno de desarrollo (puede ser
                             'prod' para producción)
APP_SECRET=tu-secreto-aqui # Un secreto para la aplicación, utilizado
                             para generar tokens, sesiones, etc.

# Configuración de base de datos
DATABASE_URL="postgresql://symfony:secret@db:5432/symfony" # URL de
conexión a la base de datos PostgreSQL

# Configuración de correos electrónicos (si aplicas)
MAILER_DSN=smtp://localhost:1025 # Puedes usar un servicio de correo local
para desarrollo
```

3.

4. Verificar la configuración en Symfony

Puedes verificar si Symfony está utilizando las variables de entorno correctamente utilizando el comando:

```
php bin/console debug:container
```

Este comando te muestra todos los servicios y sus configuraciones actuales. Si todo está bien configurado, podrás ver que las variables del archivo `.env.local` se están usando correctamente.

4. Levantar el Proyecto con Docker

Pero primero....:

- **verifica la existencia del archivo `Dockerfile`**: Asegúrate de que el archivo **Dockerfile** esté presente en la raíz de tu proyecto. Este archivo es necesario para construir la imagen del contenedor de **PHP**.
- **verifica la existencia de un archivo `docker-compose.yml`**. En la raíz de tu proyecto, crea un archivo llamado `docker-compose.yml`. Este archivo contiene la configuración de los contenedores de Docker que necesitas para tu aplicación.
- **verificar si Docker Desktop está corriendo**: Asegúrate de que **Docker Desktop** esté abierto y en funcionamiento en tu máquina. Si no está corriendo, inícialo.

Usar Docker Compose

Para facilitar el levantamiento de todo el entorno (backend, frontend y base de datos), usamos Docker Compose. Si tienes un archivo `docker-compose.yml`, simplemente ejecuta el siguiente comando:

```
docker-compose up --build
```

Este comando:

- **Construye** los contenedores si es la primera vez que los ejecutas.
- **Levanta los servicios** necesarios (PHP, Nginx/Apache, base de datos y Node.js).

Acceso a la Aplicación

Una vez que los contenedores estén en funcionamiento, puedes acceder a la aplicación desde tu navegador:

- **Backend** (Symfony) está disponible en `http://localhost:8080`.
- **Frontend** (si usa Webpack, React, Vue.js, etc.) puede estar en `http://localhost:3000` dependiendo de tu configuración.

5. Estructura de Archivos y Carpetas

La estructura del proyecto está organizada de la siguiente manera:

```
mi-proyecto-symfony/
├── config/           # Configuración del framework y servicios
├── public/           # Archivos accesibles al público (punto de entrada y
recursos estáticos)
├── src/              # Código fuente del backend (controladores, entidades,
lógica)
├── templates/        # Vistas del frontend con Twig
├── tests/            # Pruebas automatizadas
├── translations/     # Archivos de traducción (opcional)
├── var/              # Archivos temporales y de caché
├── vendor/           # Dependencias de Composer (PHP)
├── assets/           # Archivos fuente del frontend (CSS, JS)
├── node_modules/     # Dependencias de frontend (npm/yarn)
├── .env              # Variables de entorno
├── .env.local        # Configuración local
└── .gitignore        # Archivos a excluir del repositorio
```

— <code>composer.json</code>	# Dependencias PHP (Composer)
— <code>composer.lock</code>	# Bloqueo de versiones de Composer
— <code>package.json</code>	# Dependencias frontend (npm/yarn)
— <code>webpack.config.js</code>	# Configuración de Webpack Encore
— <code>Dockerfile</code>	# Instrucciones para construir la imagen Docker
— <code>docker-compose.yaml</code>	# Configuración de Docker Compose
— <code>README.md</code>	# Documentación del proyecto

6. Explicación de Componentes Clave

Backend (PHP - Symfony)

- **src/**: Código fuente del backend, incluyendo controladores (**Controller/**), entidades (**Entity/**), repositorios (**Repository/**), etc.
- **config/**: Configuración global de Symfony, servicios y rutas.
- **public/**: Punto de entrada para el backend y recursos estáticos accesibles (como el archivo `index.php`).
- **tests/**: Contiene las pruebas automatizadas para el backend.
- **var/**: Archivos temporales generados por Symfony (caché y logs).

Frontend (HTML, CSS, JS)

- **assets/**: Archivos fuente del frontend (CSS, JS, imágenes) antes de compilar.
- **public/**: Archivos estáticos generados después de la compilación (como archivos JS y CSS optimizados).
- **templates/**: Vistas que usan Twig para combinar datos dinámicos del backend con HTML.

Docker

- **Dockerfile**: Instrucciones para construir la imagen del backend (PHP) y herramientas necesarias.
 - **docker-compose.yaml**: Coordina todos los contenedores (backend, frontend, base de datos, etc.).
-

7. Descripción de Servicios en Docker Compose

- **php**: Contenedor para el backend (PHP con Symfony).
 - **webserver**: Servidor web Nginx que gestiona las solicitudes y las redirige al contenedor PHP.
 - **db**: Contenedor para la base de datos (PostgreSQL).
 - **node**: Contenedor para manejar la compilación de recursos frontend (usando Node.js y herramientas como Webpack).
-

8. Comandos Útiles

- **Levantar contenedores:**

```
docker-compose up --build
```

- **Parar los contenedores:**

```
docker-compose down
```

- **Acceder a un contenedor en ejecución (por ejemplo, PHP):**

```
docker exec -it php_app bash
```

- **Ver los registros:**

```
docker-compose logs -f
```

9. Pruebas Automatizadas

Para ejecutar las pruebas, puedes usar el siguiente comando:

```
docker-compose exec php_app vendor/bin/phpunit
```

Esto ejecutará todas las pruebas definidas en el directorio `tests/` dentro del contenedor PHP.

10. Desarrollo y Contribución

Si deseas contribuir o realizar cambios en el proyecto, asegúrate de seguir estas pautas:

- Usa un entorno de desarrollo basado en Docker.
 - Realiza pruebas localmente antes de hacer un commit.
 - Asegúrate de que el código siga los estándares de Symfony.
-

11. Referencias

- [Symfony Documentation](#)
 - [Docker Documentation](#)
 - [Docker Compose Documentation](#)
-