
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. Algunas de las variables comunes que podrías necesitar son:

```
APP_ENV=dev
APP_SECRET=tu-secreto-aqui
DATABASE_URL="postgresql://symfony:secret@db:5432/symfony"
```

---

4. Levantar el Proyecto con Docker

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.yaml`, 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
├── composer.json        # Dependencias PHP (Composer)
├── composer.lock        # Bloqueo de versiones de Composer
├── package.json         # Dependencias frontend (npm/yarn)
├── webpack.config.js    # Configuración de Webpack Encore
├── Dockerfile           # Instrucciones para construir la imagen Docker
├── docker-compose.yaml  # Configuración de Docker Compose
└── README.md            # 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
