

INFO282-Color-Ríos: DEPLOY

Requerimientos obligatorios

- Sistema Operativo Linux Ubuntu 22.04 o Windows
- [Linux] Librerías esenciales
- Docker y docker-compose

Instalaciones opcionales

Instalaciones enfocadas en el desarrollo

- Extras
 - [Windows] [scoop](#) para instalar Symfony CLI
 - [Windows] [openssl](#) para generar un jwt
- Herramientas
 - [VsCode](#)
 - [Postman](#)
- Desarrollo
 - PHP & MySQL
 - [Xampp](#)
 - [Composer](#)
 - [Symfony CLI](#)
 - JS
 - [NodeJS](#)

Introducción tecnologías y arquitectura

La aplicación está dividida en

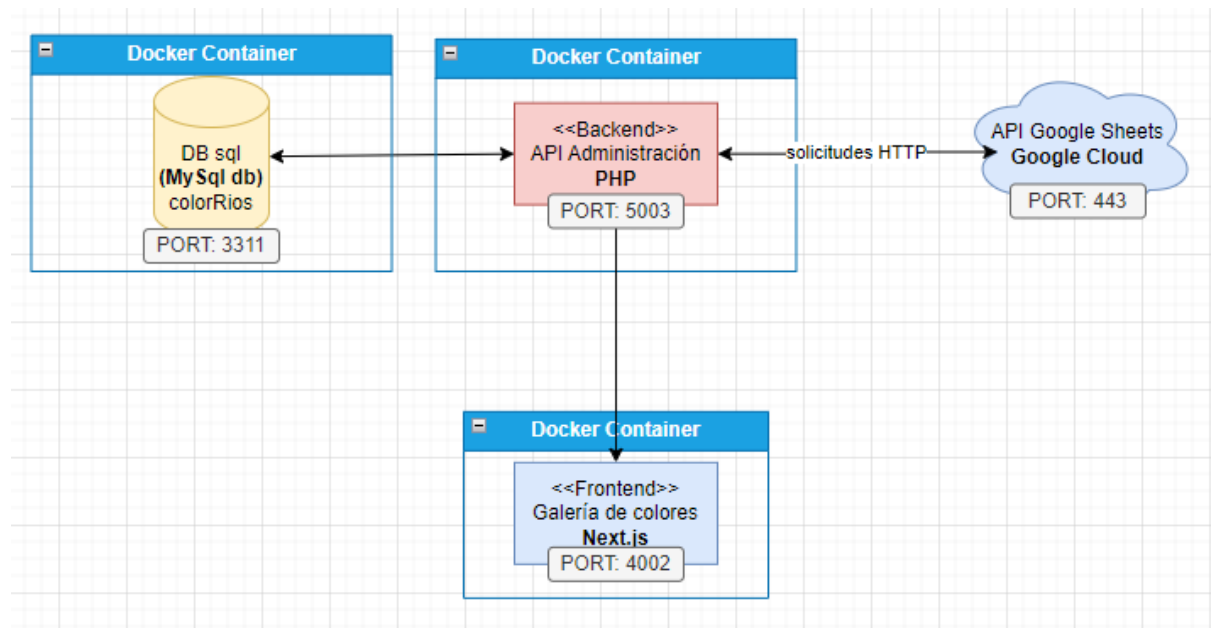
Backend:

- PHP: Framework Symfony. Encargado de los endpoints.
- MySQL. Encargada de la persistencia de datos.

Frontend:

- Javascript: Framework NextJS (React). Encargado de las vistas.

Arquitectura implementada

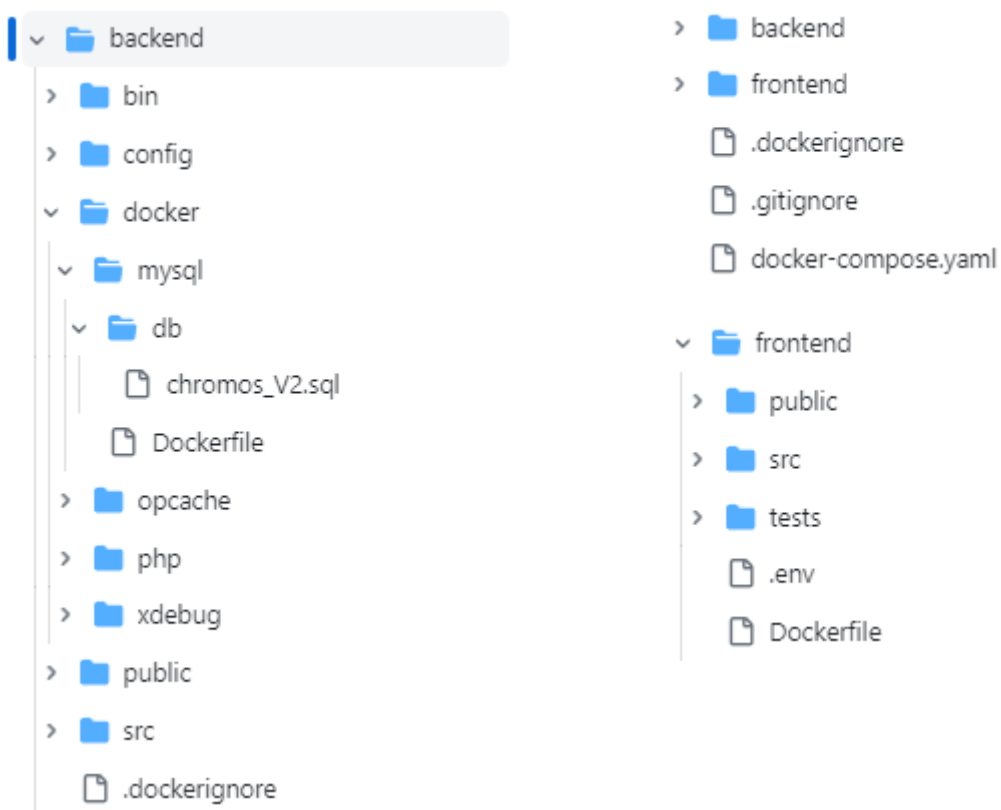


Pasos necesarios para desplegar la aplicación

#1 Clonar repositorio

```
$ git clone https://github.com/GeoCurquan/INFO282-ColorRios
$ cd /INFO282-ColorRios
```

Estructura de carpetas



2 Configuración Frontend

Se requiere un archivo **.env** ubicado en **frontend/**

```
NEXT_PUBLIC_IP=http://localhost:3000
BACKEND_IP=http://146.83.216.166:5003 # IP BACKEND endpoints
```

BACKEND_IP

Se encarga de referenciar la IP a la que irán dirigidas las solicitudes para así:

- Loguearse, registrarse, solicitar datos, etc.

3 Configuración Backend

NOTA: Algunos comandos no serán ejecutables sin los programas opcionales, es por esto que en el paso **#4** se explicará cómo acceder al **contenedor de docker** con tal de instalar dependencias u otros.

Se requiere un archivo `.env` ubicado en `backend/`

```
###> symfony/framework-bundle ###
APP_ENV=dev
APP_SECRET=84c2afb135cd001c03e06f7c6907908a
###< symfony/framework-bundle ###

DATABASE_URL=mysql://root:1234@colors-mysql:3306/chromosV2?serverVersion=mariadb-10.4.11
CORS_ALLOW_ORIGIN=http://146.83.216.166:4002 # IP FRONTEND

JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_PASSPHRASE=secret
```

DATABASE_URL

Conexión a MySQL con el usuario, contraseña y dirección dadas por nuestro docker-compose. La URL “colors-mysql” corresponde al nombre del servicio en docker-compose.

CORS_ALLOW_ORIGIN

Aquí indicaremos la URL del frontend, con tal de que pueda efectuar peticiones hacia el backend.

JWT_SECRET_KEY, JWT_PUBLIC_KEY, JWT_PASSPHRASE

Llaves secretas JWT para el manejo de sesiones del usuario utilizando [LexikJWTAuthenticationBundle](#).

Para generar las llaves deberemos correr el siguiente comando:

```
$ php bin/console lexik:jwt:generate-keypair
```

Esto debería generarnos una carpeta `backend/config/jwt` con los archivos `private.pem` y `public.pem`

Verificar la existencia de la DB `backend/docker/mysql/db`

Debe existir un archivo llamado `chromos_V2.sql` que será el inicializador de nuestra base de datos. El nombre de este archivo y su ruta es editable en el docker ubicado en la carpeta `backend/docker/mysql`

Instalación de dependencias

Deberemos instalar las dependencias necesarias para correr nuestro backend symfony. La razón de esto es que compartiremos los cambios con un volúmen del contenedor de docker.

```
$ composer install
```

4 Una vez con todo listo ejecutamos docker-compose

Nos ubicamos en la raíz de nuestro proyecto y ejecutamos el docker-compose up para levantar nuestros servicios

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

En caso de no haber podido ejecutar comandos como:

```
$ php bin/console lexik:jwt:generate-keypair
$ composer install
```

Debemos ingresar a nuestro contenedor de php con una terminal interactiva

```
$ docker exec -it info282-colorrios-mysql-1 bash
```

Deberíamos estar situados dentro del contenedor en `/var/www/symfony` lo que es equivalente a `/backend` (en local)

```
$ ~/var/www/symfony
```

Ahora solo bastará con correr los comandos

```
$ php bin/console lexik:jwt:generate-keypair
$ composer install
```

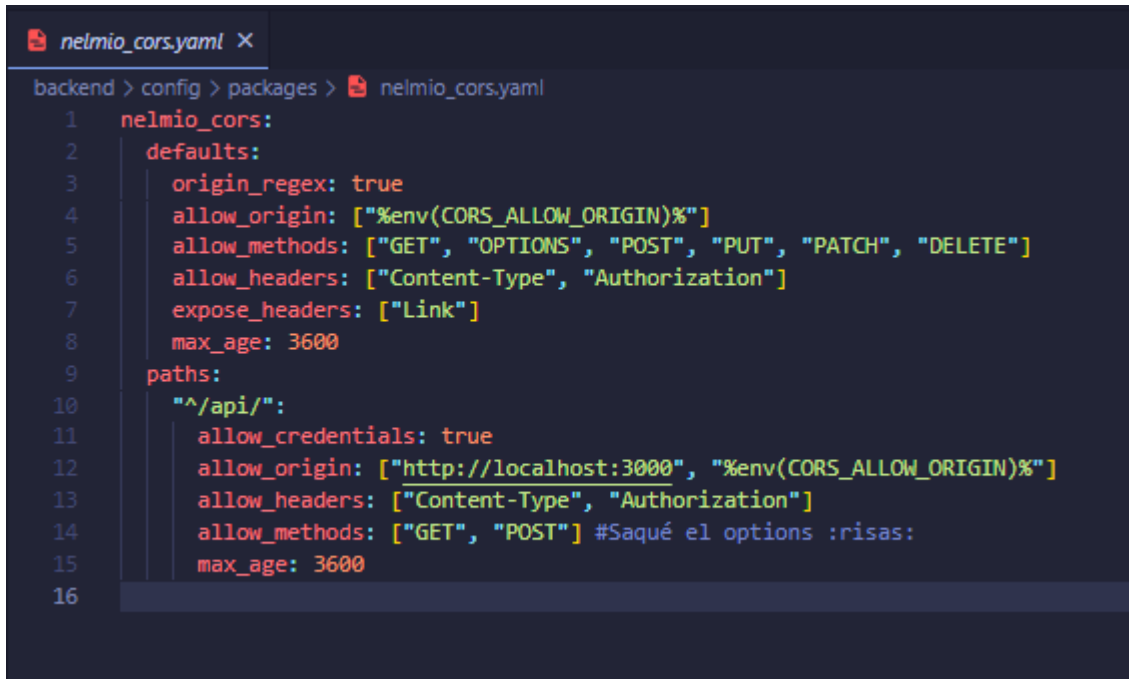
Y salir del contenedor

```
$ exit
```

Consideraciones o problemas

CORS backend/config/packages

Solo para conocimiento, hay variables como las de CORS que se manejan en el archivo `nelmio_cors.yaml`, acá se especifican quiénes pueden realizar peticiones a nuestro backend.



```
backend > config > packages > nelmio_cors.yaml
1  nelmio_cors:
2    defaults:
3      origin_regex: true
4      allow_origin: ["%env(CORS_ALLOW_ORIGIN)%"]
5      allow_methods: ["GET", "OPTIONS", "POST", "PUT", "PATCH", "DELETE"]
6      allow_headers: ["Content-Type", "Authorization"]
7      expose_headers: ["Link"]
8      max_age: 3600
9    paths:
10     "/api/":
11       allow_credentials: true
12       allow_origin: ["http://localhost:3000", "%env(CORS_ALLOW_ORIGIN)%"]
13       allow_headers: ["Content-Type", "Authorization"]
14       allow_methods: ["GET", "POST"] #Saqué el options :risas:
15       max_age: 3600
16
```

Composer install error JWT

Un error que podría ocurrir es por las dependencias de jwt, en donde deberemos configurar el archivo `php.ini` ubicado en `C:\xampp\php`.

Ahí deberemos añadir una línea que contenga `extension=php_sodium.dll`

Extra steps:

Create Admin [POST] /api/register

```
{
  "username": "admin",
  "job": "Estudiante",
  "password": "admin",
  "image": "https://i.imgur.com/fpGnrz4.png",
  "region": "Arica y Parinacota",
  "gender": "Masculino",
  "commune": "Arica",
  "roles": ["ROLE_ADMIN"]
}
```

Insert_colors (using bearer token)

[POST] /api/insertar_color

Insert_colors_stat (using bearer token)

[POST] /api/createColorStat