

React Social Network Project

Este proyecto es una aplicación full-stack de una red social simple, diseñada con una arquitectura moderna y desacoplada de cliente-servidor.

Core Technologies

- **Frontend:** React, TypeScript, Vite, React Router, Zustand
- **Backend:** Node.js, Express, TypeScript, Prisma, PostgreSQL
- **Containerization:** Docker, Docker Compose

Arquitectura del Proyecto

El proyecto sigue un modelo cliente-servidor:

- **frontend/:** Una Single Page Application (SPA) desarrollada con React que consume la API del backend. Se encarga de toda la lógica de la interfaz de usuario y la interacción con el usuario.
- **backend/:** Una API RESTful desarrollada con Node.js y Express. Gestiona la lógica de negocio, la autenticación de usuarios y la comunicación con la base de datos PostgreSQL.

Getting Started

A continuación se describen los pasos para levantar el proyecto completo.

Prerrequisitos

- Node.js (v18 o superior)
- Docker y Docker Compose

1. Método Recomendado: Usando Docker Compose

Este es el método más sencillo para levantar toda la aplicación, ya que gestiona la base de datos, el backend y el frontend automáticamente.

1. Clonar el repositorio:

```
git clone <URL-DEL-REPOSITORIO>
cd react-social-network-project
```

2. Crear el archivo de variables de entorno:

Crea un archivo `.env` en la raíz del proyecto copiando el archivo de ejemplo `.env.example` y llenando los valores.

```
cp .env.example .env
```

3. Levantar los servicios:

Este comando construirá las imágenes y levantará los tres contenedores (db, backend, frontend).

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

La aplicación estará disponible en:

- **Frontend:** `http://localhost:3000`
- **Backend:** `http://localhost:3001`

2. Método Alternativo: Ejecución Local (Manual)

Si prefieres no usar Docker, puedes ejecutar cada servicio por separado.

Backend Setup

1. Navega a la carpeta del backend: `cd backend`
2. Instala las dependencias: `npm install`
3. Crea un archivo `.env` con las variables de entorno (puedes usar el `.env.example` como guía).
4. Asegúrate de tener una instancia de PostgreSQL corriendo y que la `DATABASE_URL` en tu `.env` apunte a ella.
5. Ejecuta las migraciones de la base de datos: `npx prisma migrate dev`
6. Inicia el servidor de desarrollo: `npm run dev`

Frontend Setup

1. En otra terminal, navega a la carpeta del frontend: `cd frontend`
2. Instala las dependencias: `npm install`
3. Inicia el servidor de desarrollo: `npm run dev`

Estructura de Carpetas

```
.
├── backend/           # Código fuente de la API (Node.js)
│   ├── prisma/
│   └── src/
├── frontend/         # Código fuente de la App (React)
│   └── src/
├── .env.example      # Ejemplo de variables de entorno
├── .gitignore
├── docker-compose.yml # Orquestación de los servicios
└── README.md
```

API Endpoints Principales

La API base se encuentra en `/api`.

Método	Endpoint	Descripción	Requiere Auth
POST	<code>/auth/register</code>	Registra un nuevo usuario.	No
POST	<code>/auth/login</code>	Inicia sesión y devuelve un token JWT.	No
GET	<code>/auth/profile</code>	Obtiene el perfil del usuario autenticado.	Sí
GET	<code>/posts</code>	Obtiene todas las publicaciones.	Sí
POST	<code>/posts</code>	Crea una nueva publicación.	Sí
PATCH	<code>/posts/:id/like</code>	Incrementa los "me gusta" de una publicación.	Sí
PATCH	<code>/posts/:id</code>	Edita una publicación propia.	Sí
DELETE	<code>/posts/:id</code>	Elimina una publicación propia.	Sí

Scripts Disponibles

Frontend (`/frontend`)

- `npm run dev`: Inicia el servidor de desarrollo.
- `npm run build`: Construye la aplicación para producción.
- `npm run lint`: Ejecuta el linter para detectar errores.

Backend (`/backend`)

- `npm run dev`: Inicia el servidor de desarrollo con hot-reload.
- `npm start`: Inicia el servidor en modo producción (requiere `npm run build` primero).
- `npm run lint`: Ejecuta el linter para detectar errores.
- `npm test`: Ejecuta las pruebas automatizadas con Jest.