Social Network – Backend (Node.js + TypeScript + Express)

Backend para una red social simple con autenticación JWT, publicaciones, likes y perfil, usando Express, Prisma y PostgreSQL. Incluye pruebas, linting, documentación OpenAPI y orquestación con Docker Compose junto al frontend (Vite + React).

Arquitectura

- Backend: Node.js, Express, TypeScript, Prisma ORM, JWT
- Base de datos: PostgreSQL
- Frontend: Vite + React + TypeScript (carpeta hermana `../connect-circle-93`)

Requisitos

- Node.js 20+
- PostgreSQL 14+ (o usar Docker Compose)

Variables de entorno (backend)

Crear `.env` (puedes copiar `.env.example` si existe) con:

```
 PORT=4001
JWT SECRET=dev secret
JWT EXPIRES IN=3600
DATABASE_URL=postgresql://postgres:postgres@localhost:5432/socialdb?schema=public
```

Desarrollo local (sin Docker)

1. Instalar dependencias
2. Generar Prisma Client y aplicar migraciones
3. (Opcional) Cargar datos semilla
4. Ejecutar en modo desarrollo

Scripts útiles:

- `npm run dev` – Dev server en watch (por defecto en 4001)
- `npm run build` – Compila a `dist/`
- `npm start` – Ejecuta compilado
- `npm test` – Pruebas (Jest + Supertest)
- `npm run db:generate` – Prisma generate

- `npm run db:migrate` – Prisma migrate dev
- `npm run db:seed` – Semilla de usuarios y posts de demo

Docker Compose (DB + Backend + Frontend)

En esta carpeta hay un `docker-compose.yml` preparado para levantar:

- `db`: PostgreSQL 16 (puerto 5432)
- `backend`: Node + Prisma (puerto 4001)
- `frontend`: Vite (puerto 8080) apuntando a `http://localhost:4001`

Uso básico:

```
docker compose up
```

Rutas:

- Frontend: http://localhost:8080
- Backend: http://localhost:4001/health
- Docs API: http://localhost:4001/docs

Semillas dentro del contenedor backend:

```
docker compose exec backend npm run db:seed
```

API y documentación

Swagger UI ligero (sin dependencias locales):

- UI: `GET /docs`
- JSON: `GET /openapi.json`

Endpoints principales:

- `POST /auth/login` – JWT por username/email + password
- `GET /me` – Perfil autenticado
- `GET /me/stats` – Estadísticas (posts, likes dados, likes recibidos)
- `GET /posts` – Listar publicaciones (paginado, más recientes primero)
- `POST /posts` – Crear publicación (<= 280 chars)
- `POST /posts/{id}/like` – Dar like (idempotente)
- `DELETE /posts/{id}/like` – Quitar like (idempotente)

Pruebas

Ejecuta toda la suite:

```
npm test
```

Cubre autenticación, listados, paginación, creación, likes idempotentes, perfil y estadísticas, y exposición de OpenAPI.

Solución de problemas

- Puerto 4001 en uso: cambia `PORT` o cierra procesos en conflicto.
- DB local ocupando 5432: detén tu Postgres local o cambia el mapeo del compose.
- CORS: el backend permite por defecto `http://localhost:8080`.
