Instalación y entrega

Este documento explica cómo instalar y ejecutar el proyecto, y contiene notas útiles para generar los entregables solicitados (PDF, script de BD y video demo).

Requisitos previos

- Docker & Docker Compose
- Node.js 18+ (se recomienda Node 20 para compatibilidad con dependencias frontend)
- npm
- psql (cliente) si desea ejecutar el script SQL directamente

1. Levantar servicios con Docker Compose

En la raíz del repositorio:

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

Esto inicia la base de datos PostgreSQL y los servicios (backend, frontend según la configuración). Verifique variables de entorno en `.env` o `.env.example`.

2. Script SQL con usuarios predefinidos

Hay un script SQL preparado en `scripts/init_users.sql` que crea dos usuarios de ejemplo (`alice` / `password1` y `bob` / `password2`). El script usa la extensión `pgcrypto` de Postgres para crear hashes bcrypt.

Ejemplo de uso (ajuste host/usuario/BD según su entorno):

```
psql -h localhost -U postgres -d mydb -f scripts/init_users.sql
```

Si su imagen de Postgres no incluye `pgcrypto`, active la extensión o use la alternativa: ejecutar el seeder de la aplicación o un script Node que inserte usuarios con bcrypt.

3. Alternativa: usar el seeder de la app

El backend incluye una utilidad/servicio de seed (ver `src/seed/seeder.service.ts`). Puede ejecutar los seeds desde el entorno de desarrollo si prefiere que la app gestione el hashing.

4. Ejecutar backend y frontend localmente

Backend (desde la raíz):

```
npm run start:dev
```

Frontend (desde `frontend`):

```
cd frontend
npm install
npm run dev
```

5. Tests

Backend tests (Jest):

```
npx jest --config jest.config.ts
```

Frontend tests (Vitest):

```
cd frontend
npx vitest
```

6. Generar PDF de instalación

Para producir el PDF final, puede convertir este Markdown a PDF con una herramienta como `pandoc` o mediante VS Code: "Markdown: Export (PDF)".

Comando ejemplo con pandoc:

```
pandoc docs/INSTALLATION.md -o entrega_instalacion.pdf
```

7. Script y guion para el video demo

- Duración objetivo: 3-5 minutos.
- Guion recomendado:

1. Breve presentación del proyecto (15s)
2. Levantar Docker Compose y mostrar servicios (30s)
3. Mostrar login con `alice` / `password1` (20s)
4. Crear post con imagen, dar like, mostrar profile (60s)
5. Ejecutar tests rápidamente o mostrar que CI pasa (20s)

Herramientas: OBS Studio, Grabadora de pantalla nativa, o `ffmpeg`.

Comando `ffmpeg` para grabar (ejemplo en Windows PowerShell):

```
ffmpeg -f gdigrab -framerate 25 -i desktop -preset ultrafast demo.mp4
```

8. Entregables a incluir en el repositorio final

- `docs/INSTALLATION.md` (y el PDF generado `entrega_instalacion.pdf`)
- `scripts/init_users.sql`
- Captura de pantalla o `demo.mp4` (no subir archivos grandes; preferir enlace a YouTube privado o Drive)
- README actualizado con instrucciones de ejecución y link a video

Si quieres, puedo convertir este MD a PDF y añadir un `scripts/record_demo.ps1` con pasos automatizados para la demo. Dime si lo quieres ahora.

9. Notas sobre `docker-compose.yml` y seed dentro del contenedor

- El servicio `postgres` expone el puerto `5432` y monta un volumen `pgdata` para persistencia.
- El `healthcheck` usa `pg_isready` — al ejecutar scripts o arrancar tests espere a que el servicio esté sano.

Ejemplo para ejecutar el script SQL dentro del contenedor Postgres (desde la raíz del repo):

```
# copiar el script al contenedor y ejecutarlo con psql en el contenedor
docker-compose up -d postgres
docker exec -i $(docker-compose ps -q postgres) psql -U ${DB_USER:-postgres} -d ${DB_NAME:-socialdb} <
scripts/init_users.sql
```

O usando `psql` local apuntando al contenedor:

```
psql -h localhost -U postgres -d socialdb -f scripts/init_users.sql
```

Si CI necesita esperar a Postgres, use la técnica de "wait-for" (o la acción `wait-for-postgres`) y/o compruebe el `healthcheck` antes de ejecutar tests.