# Manual de Instalación – Periferia Social

## 1. Prerrequisitos

- Node.js 20 o superior
- npm 10 o superior
- Docker y Docker Compose
- Acceso a PowerShell (Windows) o Bash (Linux/Mac)

## 2. Clonado del repositorio

```
git clone https://github.com/markus993/pruebaFullstackPeriferia.git
cd pruebaFullstackPeriferia
```

## 3. Configuración de variables de entorno

1. Duplica el archivo de variables de entorno de ejemplo (env.example) y renombra el archivo nuevo a ".env"
2. Verifica los siguientes valores clave:
    - `POSTGRES_PASSWORD` : contraseña de PostgreSQL.
    - `API_PORT` y `FRONTEND_PORT` : puertos expuestos.
    - `JWT_SECRET` : clave segura para firmar JWT.
    - `DATABASE_URL` : cadena de conexión que utilizará Prisma (por defecto apunta al servicio `postgres-db` ).

> *Para el despliegue con Docker basta con dejar los valores por defecto.*

## 4. Despliegue con Docker Compose

1. Construye y levanta la solución completa:

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

2. Espera a que el contenedor `api` muestre en logs "Executing seed" y "Iniciando servidor Nest".
3. Accede a los servicios:
    - Frontend: `http://localhost:5173`
    - Backend: `http://localhost:3000/api`

## 5. Credenciales de prueba

| Usuario | Alias | Contraseña |
|---|---|---|
| ana.romero@periferia.it | `@anar` | `Periferia123!` |
| carlos.mendez@periferia.it | `@carlitos` | `Periferia123!` |
| laura.castillo@periferia.it | `@lauca` | `Periferia123!` |

# 6. Pruebas unitarias

## Backend (Jest)

### Ejecución local (backend)

1. Instala dependencias (solo la primera vez):

```
cd backend
npm install
```

2. Ejecuta los tests:

```
npm run test
```

3. Para un reporte de cobertura:

```
npm run test:cov
```

### Ejecución con Docker Compose (backend)

1. Asegúrate de tener los servicios arriba:

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

2. Corre las pruebas dentro del contenedor `api`:

```
docker compose exec api npm run test
```

3. Para obtener cobertura en Docker:

```
docker compose exec api npm run test:cov
```

## Frontend (Vitest)

### Ejecución local (frontend)

1. Instala dependencias (solo la primera vez):

```
cd frontend
npm install
```

2. Ejecuta los tests (modo una sola corrida):

```
npm run test -- --run
```

3. Para cobertura:

```
npm run test:coverage
```

**Ejecución con Docker Compose (frontend)**

1. Asegúrate de tener el servicio levantado:

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

2. Corre las pruebas dentro del contenedor `frontend`:

```
docker compose exec frontend npm run test -- --run
```

3. Para cobertura:

```
docker compose exec frontend npm run test:coverage
```

## 7. Endpoints principales

- `GET /api/health` – health check del backend.
- `POST /api/auth/login` – inicio de sesión, devuelve JWT + perfil.
- `GET /api/users/me` – perfil del usuario autenticado.
- `GET /api/posts` – feed de publicaciones de otros usuarios.
- `POST /api/posts` – crear publicación.
- `POST /api/posts/{id}/like` – enviar like idempotente.
- Swagger UI: `http://localhost:3000/api/docs` (o `http://localhost:${API_PORT}/api/docs` si personalizaste el puerto).

## 8. Resolución de problemas

- **Docker no arranca servicios**: Verifica que los puertos 3000/5173 estén libres.
- **Seeds fallan**: borra los volúmenes de PostgreSQL ( `docker compose down -v` ) y vuelve a levantar.
- **Frontend responde**: si acabas de inciar el servicio, espera unos 5-10 segundos, aun esta terminando el inicio del servicio Front.
- **Frontend no llega al backend**: confirma que `VITE_API_URL` (o la URL configurada en Docker) apunta a `http://localhost:3000` .