Readme:

# Training App — NestJS + TypeORM + PostgreSQL

API de ejemplo para gestionar:

\*\*Usuarios\*\* con NestJS.

Incluye:

- CRUD real con \*\*TypeORM\*\* y \*\*PostgreSQL\*\*

- \*\*Validaciones\*\* con DTOs (`class-validator`) y `ValidationPipe` global

- \*\*Hash de contraseña\*\* con `bcrypt`

- \*\*Swagger\*\* en `/docs` para explorar la API

Requisitos:

- Node.js (LTS recomendado por Nest)

- Nest CLI

```bash

npm i -g @nestjs/cli

* Docker Desktop (para levantar PostgreSQL)
* Postman

Instalación:

1.- Clonar e instalar las siguientes dependencias:

npm i

2.- Crear archivo. env en la raíz con lo siguiente:

PORT=3000

DB\_HOST=localhost

DB\_PORT=5432

DB\_USER=nest

DB\_PASS=nest

DB\_NAME=training

3.- Levantar la base de datos con Docker:

docker compose up -d

docker ps # Debe verse un contenedor postgres "Up"

Ejecución:

Desarrollando en modo recarga automática:

npm run start:dev

# Application is running on: <http://localhost:3000>

# Docs: <http://localhost:3000/docs>

Swagger (Documentación interactiva)

<http://localhost:3000/docs>

4.- Endpoints Principales:

Base URL:

<http://localhost:3000>

Será nuestra Url donde mostraremos en local la página.

Recurso:

/users

DTO de entrada (Validado):

Json

{

"name": "string (min:2)",

"lastName": "string (min:2)",

"email": "email válido",

"password": "string (min:8, letras y números)"

}

Crear usuario:

En Postman:

Post /users

Body (json)

{

"name": "Willy",

"lastName": "Fester",

"email": "willyfester1997@gmail.com",

"password": "abc12345"

}

Lo que nos arrojara una respuesta 201/200 :

{

"id": 1,

"name": "Willy",

"lastName": "Fester",

"email": "willyfester1997@gmail.com",

"createdAt": "2025-09-06T12:34:56.000Z",

"updatedAt": "2025-09-06T12:34:56.000Z"

}

* El Password no se devuelve (esta se guarda hasheada)

cURL:

curl -X POST http://localhost:3000/users \

- H "Content-Type: application/json" \

- d ‘{“name”: ”Willy”,”lastName”:”Fester”,”email”:”willyfester1997@gmail.com”,”password”:”abc12345”}

Listar Usuarios:

En postman:

Get / User

[

{

"id": 1,

"name": "Willy",

"lastName": "Fester",

"email": "willyfester1997@gmail.com,

"createdAt": "2025-09-06T12:34:56.000Z",

"updatedAt": "2025-09-06T12:34:56.000Z"

}

]

cURL:

curl: <http://localhost:3000/users>

Obtener usuario por Id:

En postman:

GET /users/: id

curl <http://localhost:3000/users/1>

Editar parcialmente:

PATCH /users/:id

Body (JSON, cualquier campo del DTO):

{ "name": "Willy F." }

Curl:

curl -X PATCH http://localhost:3000/users/1 \

-H "Content-Type: application/json" \

-d '{"name":"Willy F."}'

Eliminar:

En postman

DELETE /users/:id

Respuesta 200 :

{ "message": "Usuario eliminado" }

cURL:

curl -X DELETE <http://localhost:3000/users/1>

Validaciones (Errores Comunes):

En postman

**POST inválido** (ej.):

{

"name": "A",

"email": "no",

"password": "123",

"hacker": true

}

Respuesta 400 (ej.):

{

"statusCode": 400,

"message": [

"name debe tener al menos 2 caracteres.",

"email debe tener formato válido.",

"password debe tener al menos 8 caracteres.",

"password debe incluir letras y números.",

"property hacker should not exist"

],

"error": "Bad Request"

}

Scripts útiles:

{

"start": "nest start",

"start:dev": "nest start --watch",

"build": "nest build",

"test": "jest"

}

Troubleshooting:

* /docs devuelve 404 → revisa imports/config de Swagger en main.ts.
* No conecta a Postgres → confirma docker compose up -d, .env correcto y puerto (5432).
* “client password must be a string” → DB\_PASS vacío o mal leído; corrige .env.
* Campos extra o formato inválido → el ValidationPipe los bloquea (ver sección Validaciones).