VALIDACIÓN Y VERIFICACIÓN DE SOFTWARE (INF 732)
GUÍA DE LABORATORIO 6
Pruebas de E2E para el módulo Notas

Las pruebas End-to-End (E2E) son un tipo de prueba de software que verifica el funcionamiento de una aplicación de principio a fin, tal como lo haría un usuario real.

El objetivo principal de una prueba E2E es asegurarse de que todos los componentes del sistema funcionen correctamente juntos.

En una prueba E2E, no solo se comprueba una función aislada (como ocurre en las pruebas unitarias), sino que se simula el flujo completo de trabajo, incluyendo:

Solicitudes HTTP (por ejemplo, crear, actualizar o eliminar un recurso), Validación de datos, Procesamiento en el servidor y Respuestas enviadas al cliente.

Objetivos

- Comprender y aplicar pruebas E2E usando Jest, Supertest y ValidationPipe en una API REST construida con NestJS.
- Validar los flujos completos de creación y validación de datos en un endpoint /notas.

Prerrequisitos

Haber desarrollado las Guías 1, 2, 3, 4 y 5 de la asignatura.

1. Ejemplo parcial

test/notas.e2e-spec.ts

```
import { Test, TestingModule } from '@nestjs/testing';
import { INestApplication, ValidationPipe } from '@nestjs/common';
import * as request from 'supertest';
import { AppModule } from '../src/app.module';
import { getRepositoryToken } from '@nestjs/typeorm';
import { Nota } from '../src/notas/nota.entity';
```

```
import { DataSource, Repository } from 'typeorm';
import { CreateNotaDto } from 'src/notas/dto/create-nota.dto';

describe('NotasController (e2e)', () => {
  let app: INestApplication;
  let notaRepository: Repository<Nota>;

  beforeAll(async () => {
    const moduleFixture: TestingModule = await Test.createTestingModule({
      imports: [AppModule],
    }).compile();

    app = moduleFixture.createNestApplication();
    app.useGlobalPipes(
      new ValidationPipe({
        whitelist: true,
        forbidNonWhitelisted: true,
        transform: true,
      }),
    );
    notaRepository = moduleFixture.get<Repository<Nota>>(
      getRepositoryToken(Nota),
    );

    await app.init();
  });

  afterAll(async () => {
    await app.close();
    const dataSource = app.get(DataSource);
    if (dataSource.isInitialized) {
      await dataSource.destroy();
    }
  });

  afterEach(async () => {
    await notaRepository.clear();
  });

  describe('/notas (POST)', () => {
    it('debería crear una nueva nota', async () => {
      const createNotaDto = {
        title: 'Nota de prueba',
        content: 'Este es el contenido de prueba',
      };

      const response = await request(app.getHttpServer())
        .post('/notas')
        .send(createNotaDto)
        .expect(201);
```

```
      expect(response.body).toHaveProperty('id');
      expect(response.body.title).toEqual(createNotaDto.title);
      expect(response.body.content).toEqual(createNotaDto.content);
    });

    it('debería fallar si no se proporciona el título', async () => {
      const invalidDto = {
        content: 'Content without title',
      } as CreateNotaDto;

      const response = await request(app.getHttpServer())
        .post('/notas')
        .send(invalidDto);

      expect(response.status).toBe(400);
      expect(response.body.message).toContain('The title is required');
    });

    it('deberia fallar si no se proporciona el contenido', async () => {
      const invalidDto = {
        title: 'Title without content',
      } as CreateNotaDto;

      const response = await request(app.getHttpServer())
        .post('/notas')
        .send(invalidDto);

      expect(response.status).toBe(400);
      expect(response.body.message).toContain('The content is required');
    });
  });
});
```

Finalmente, para ejecutar los test, utilice el siguiente comando:

```
npm run test:e2e
```

Recuerde que el servidor de base de datos debe estar en ejecución.

Trabajo independiente.

Complete el resto de pruebas necesarios para cada endpoint del controlador.