VALIDACIÓN Y VERIFICACIÓN DE SOFTWARE (INF 732)
GUÍA DE LABORATORIO 2
API RESTful

En este laboratorio, desarrollaremos una API RESTful para la gestión de notas utilizando NestJS y MySQL, aplicando buenas prácticas de desarrollo backend como la estructura modular, el uso de TypeORM para la interacción con la base de datos y la implementación de operaciones CRUD (Crear, Leer, Actualizar y Eliminar). Esta guía te proporcionará los pasos necesarios para configurar el entorno, definir la estructura del proyecto, implementar los endpoints esenciales y validar su funcionamiento mediante pruebas con Thunder Client. Al finalizar, tendrás una API escalable y bien organizada, lista para integrarse con frontends o extenderse con autenticación, filtros avanzados y más.

Objetivos

1. **Desarrollar una API RESTful funcional** utilizando **NestJS** para gestionar notas (CRUD: Crear, Leer, Actualizar y Eliminar).
2. **Configurar una base de datos MySQL** y aprender a integrarla con NestJS mediante **TypeORM** (u otro ORM como Prisma).
3. **Aplicar la estructura modular de NestJS**, organizando el código en **módulos, servicios, controladores y entidades**.
4. **Implementar validación de datos** usando **DTOs (Data Transfer Objects)** y decoradores de class-validator.
5. **Probar los endpoints** manualmente con **Thunder Client** y entender el flujo de solicitudes/respuestas HTTP.
6. **Fomentar buenas prácticas** como inyección de dependencias, código limpio y manejo básico de errores.

Prerrequisitos

- Node.js (v18+) y npm/yarn instalados.
- Conocimientos básicos de TypeScript y REST APIs.
- MySQL instalado o acceso a una base de datos.

1. Crear un Nuevo Proyecto

Con el CLI de NestJS, puedes crear un nuevo proyecto con el siguiente comando:

```
nest new api-notas
```

Esto iniciará un asistente que te preguntará qué administrador de paquetes deseas usar (npm o yarn). Selecciona el que prefieras y espera a que se instalen las dependencias.

Ingresa a la carpeta del proyecto.

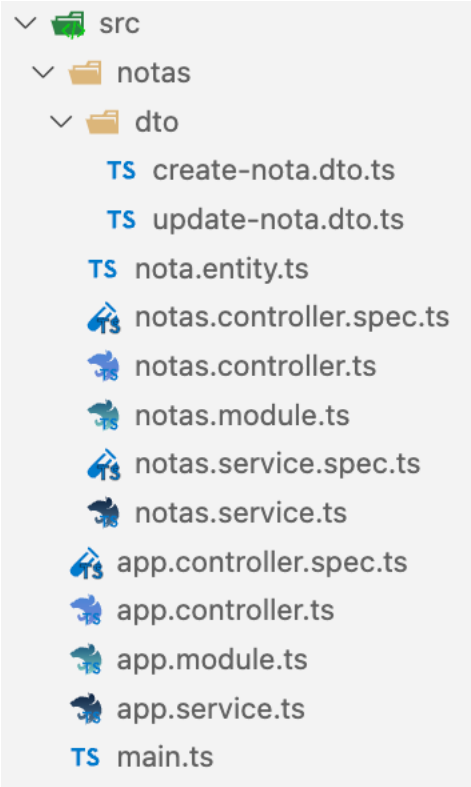
```
cd api-notas
```

Verifica tener instalado o en su caso instala las siguientes dependencias:

1. @nestjs/config - Para manejar variables de entorno (se añade con npm i @nestjs/config).
2. @nestjs/typeorm - Integración con TypeORM (se añade con npm i @nestjs/typeorm typeorm).
3. class-transformer - Usado para transformar objetos (común en DTOs y validación, pero no viene por defecto).
4. class-validator - Validación basada en decoradores (se instala aparte).
5. mysql2 - Driver de MySQL para TypeORM/Node.js (dependencia de base de datos opcional).
6. typeorm - ORM para bases de datos SQL/NoSQL (requiere instalación manual).

2. Estructura del proyecto

Tu proyecto debe tener la siguiente estructura (Revisa la Guía de Laboratorio 1 para crear el módulo, servicio y controlador, los otros archivos y carpetas puedes crearlos manualmente)



3. Implementación de la API.

Configuración TypeORM en app.module.ts

app.module.ts

```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { NotaModule } from './notas/notas.module';
import { TypeOrmModule } from '@nestjs/typeorm';
import { ConfigModule } from '@nestjs/config';
import { Nota } from './notas/nota.entity';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'mysql',
      host: 'localhost',
      port: 3306,
      username: 'root',
      password: '',
      database: 'notas_db',
      entities: [Nota],
      synchronize: true,
    }),
    ConfigModule.forRoot({ isGlobal: true }),
```

```
    NotaModule,
  ],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

Asegúrese de tener una base de datos con el nombre “notas_db” y de tener configurado los datos del anterior archivo según como tenga configurado MySQL.

Crear la Entidad Nota

nota.entity.ts

```
import { Column, Entity, PrimaryGeneratedColumn } from 'typeorm';

@Entity()
export class Nota {
  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  title: string;

  @Column()
  content: string;
}
```

Crea el DTO para crear notas

create-nota.dto.ts

```
import { IsNotEmpty, IsString } from 'class-validator';

export class CreateNotaDto {
  @IsString()
  @IsNotEmpty({ message: 'The title is required' })
  title: string;

  @IsString()
  @IsNotEmpty({ message: 'The content is required' })
  content: string;
}
```

Crear el DTO para modificar notas

update-nota.dto.ts

```
import { IsNotEmpty, IsString, IsOptional } from 'class-validator';

export class UpdateNotaDto {
  @IsString()
  @IsNotEmpty({ message: 'El título es requerido' })
  @IsOptional()
  title?: string;

  @IsString()
  @IsNotEmpty({ message: 'El contenido es requerido' })
  @IsOptional()
  content?: string;
}
```

Crear el servicio de notas

notas.service.ts

```
import { Injectable, NotFoundException } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Nota } from './nota.entity';
import { Repository } from 'typeorm';
import { CreateNotaDto } from './dto/create-nota.dto';
import { UpdateNotaDto } from './dto/update-nota.dto';

@Injectable()
export class NotasService {
  constructor(
    @InjectRepository(Nota)
    private notasRepository: Repository<Nota>,
  ) {}

  async create(createNotaDto: CreateNotaDto): Promise<Nota> {
    const newNota = this.notasRepository.create(createNotaDto);
    return this.notasRepository.save(newNota);
  }

  async findOne(id: number): Promise<Nota> {
    const nota = await this.notasRepository.findOneBy({ id });
    if (!nota) {
      throw new NotFoundException(`Nota con ID ${id} no encontrada`);
    }
    return nota;
  }

  async findAll(): Promise<Nota[]> {
```

```
    return this.notasRepository.find();
  }

  async update(id: number, updateNotaDto: UpdateNotaDto): Promise<Nota> {
    const updateResult = await this.notasRepository.update(id,
updateNotaDto);
    if (updateResult.affected === 0) {
      throw new NotFoundException(`Nota con ID ${id} no encontrada`);
    }
    return this.findOne(id);
  }

  async remove(id: number): Promise<void> {
    const result = await this.notasRepository.delete(id);
    if (result.affected === 0) {
      throw new NotFoundException(`Nota con ID ${id} no encontrada`);
    }
  }
}
```

Crear el controlador

notas.controller.ts

```
import {
  Controller,
  Get,
  Post,
  Body,
  Param,
  Put,
  Delete,
} from '@nestjs/common';
import { NotasService } from './notas.service';
import { Nota } from './nota.entity';
import { CreateNotaDto } from './dto/create-nota.dto';
import { UpdateNotaDto } from './dto/update-nota.dto';

@Controller('notas')
export class NotasController {
  constructor(private readonly notasService: NotasService) {}

  @Post()
  async create(@Body() createNotaDto: CreateNotaDto): Promise<Nota> {
    return this.notasService.create(createNotaDto);
  }

  @Get(':id')
  async findOne(@Param('id') id: string): Promise<Nota> {
    return this.notasService.findOne(+id);
  }
```

```
  @Get()
  async findAll(): Promise<Nota[]> {
    return this.notasService.findAll();
  }

  @Put(':id')
  async update(
    @Param('id') id: string,
    @Body() updateNotaDto: UpdateNotaDto,
  ): Promise<Nota> {
    return this.notasService.update(+id, updateNotaDto);
  }

  @Delete(':id')
  async remove(@Param('id') id: string): Promise<void> {
    return this.notasService.remove(+id);
  }
}
```

4. Ejecución y pruebas

Ejecute con el comando:

```
npm start
```

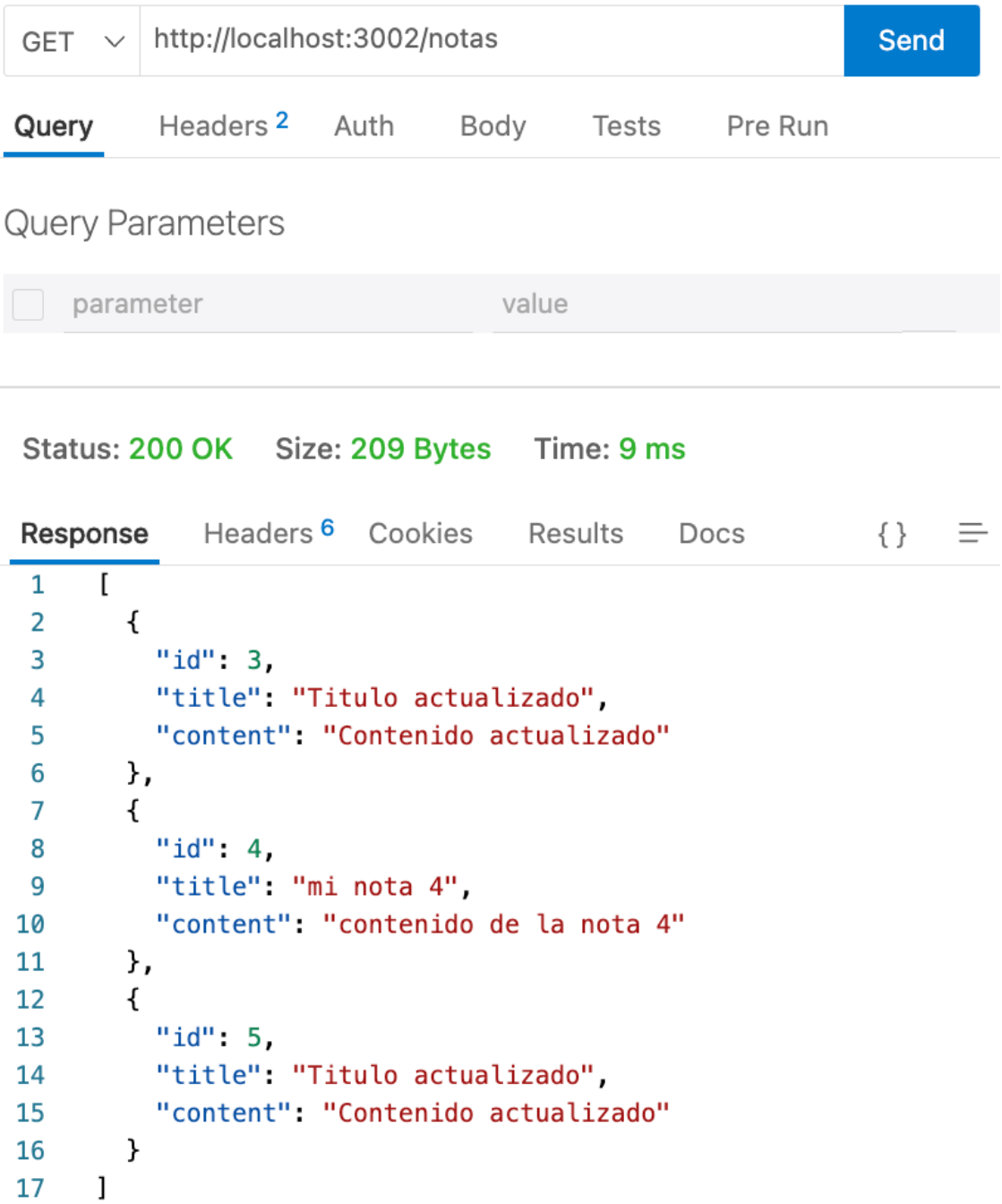
La terminal deberá indicar los endpoints en ejecución:

```
huascarfedor@192 lab3-notas-app % npm start

> lab3-notas-app@0.0.1 start
> nest start

[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [NestFactory] Starting Nest application...
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [InstanceLoader] TypeOrmModule dependencies initialized +37ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [InstanceLoader] ConfigHostModule dependencies initialized +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [InstanceLoader] AppModule dependencies initialized +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [InstanceLoader] ConfigModule dependencies initialized +1ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [InstanceLoader] TypeOrmCoreModule dependencies initialized +104ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [InstanceLoader] TypeOrmModule dependencies initialized +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [InstanceLoader] NotaModule dependencies initialized +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RoutesResolver] AppController {/}: +3ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RouterExplorer] Mapped {/, GET} route +1ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RoutesResolver] NotasController {/notas}: +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RouterExplorer] Mapped {/notas, POST} route +1ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RouterExplorer] Mapped {/notas/:id, GET} route +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RouterExplorer] Mapped {/notas, GET} route +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RouterExplorer] Mapped {/notas/:id, PUT} route +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [RouterExplorer] Mapped {/notas/:id, DELETE} route +0ms
[Nest] 19305  - 03/30/2025, 11:25:25 PM     LOG [NestApplication] Nest application successfully started +1ms
```

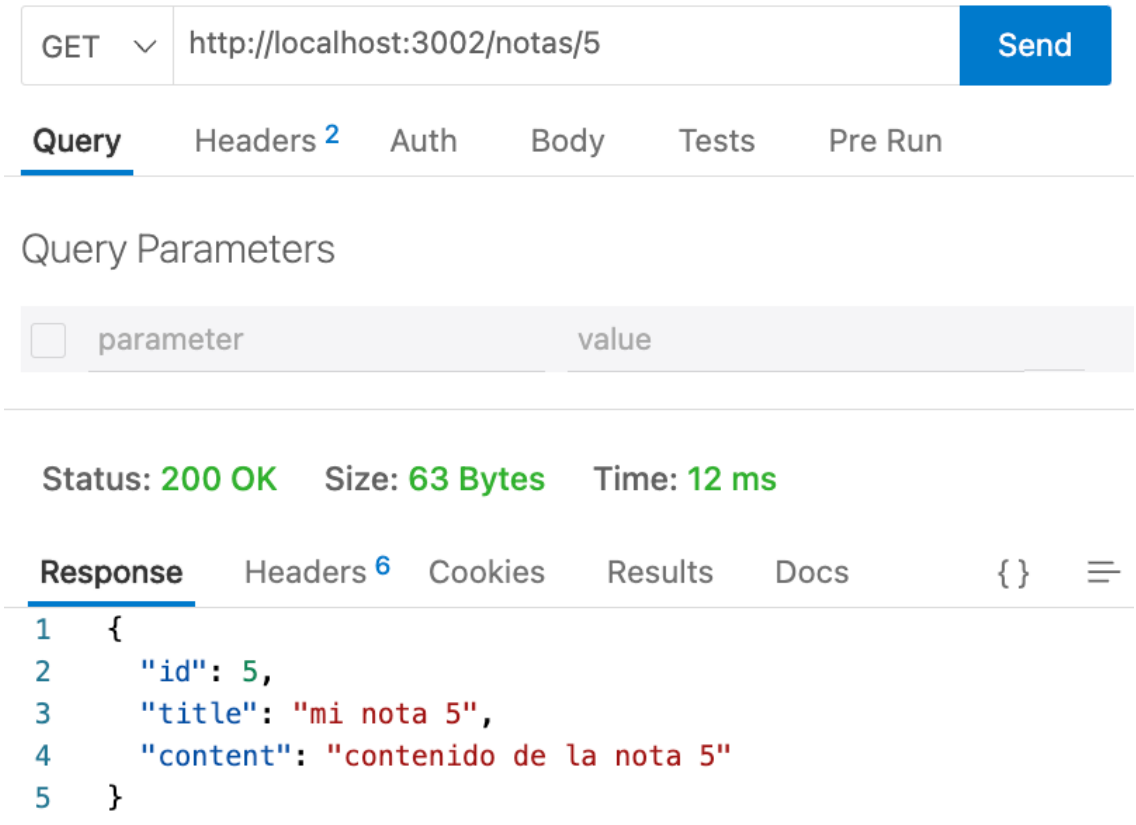
Los endpoints a probar son:

- POST /notas → Crear nota.

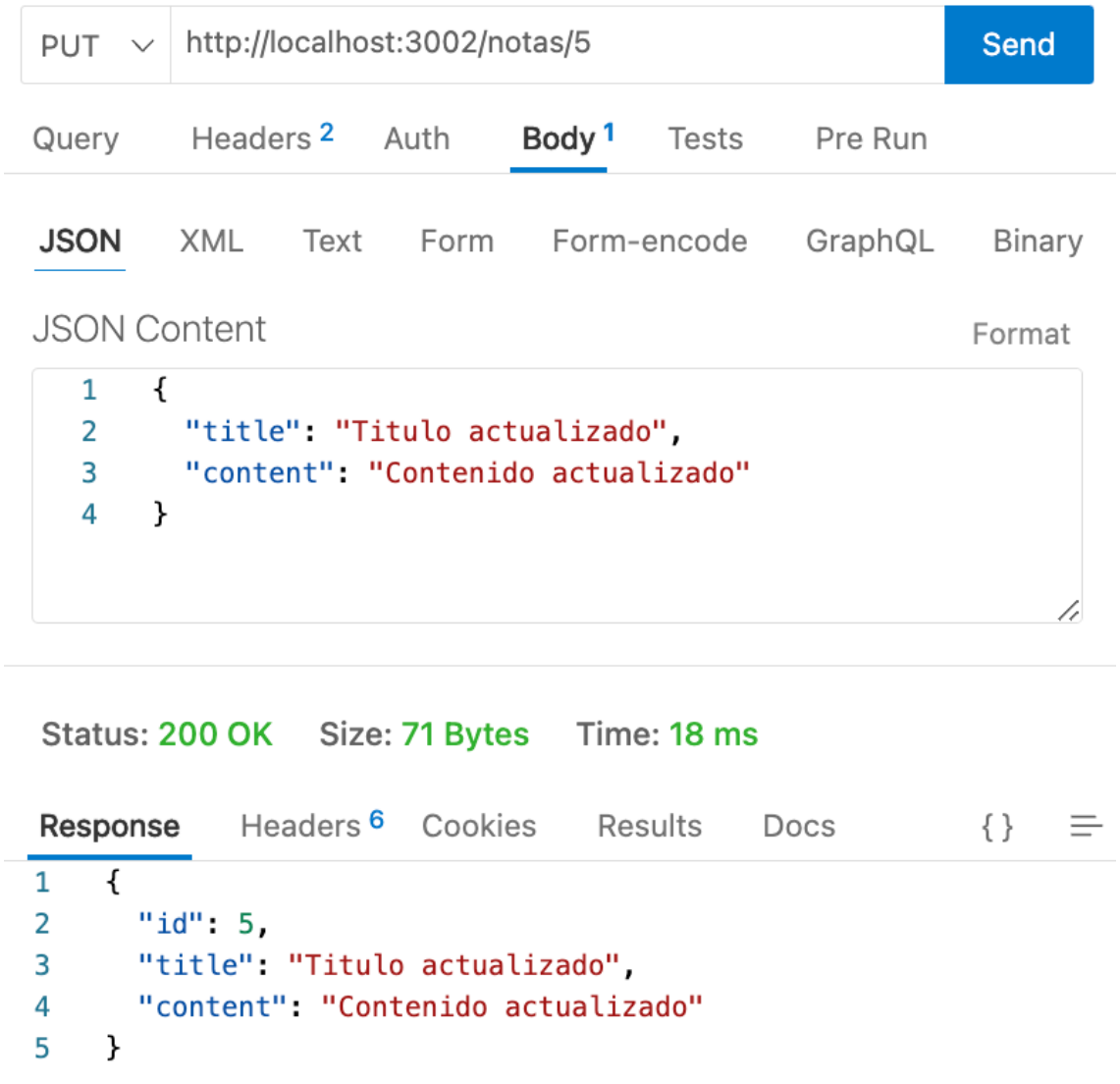
- GET /notas → Obtener todas.



- GET /notas/:id → Obtener por ID.



- PUT /notas/:id → Actualizar.



- DELETE /notas/:id → Eliminar.