Nest

CRUD

CertiDevs

Índice de contenidos

1. Introducción

A continuación, se presenta un ejemplo de cómo implementar un CRUD de tareas en NestJS utilizando TypeORM y controladores.

Este ejemplo asume que ya tienes una base de datos configurada y lista para usar con TypeORM.

2. Crear la entidad Task

```
// src/tasks/task.entity.ts
import { Entity, PrimaryGeneratedColumn, Column } from 'typeorm';

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

    @Column()
    title: string;

    @Column()
    description: string;

    @Column({ default: false })
    completed: boolean;
}
```

3. Crear el módulo TasksModule

```
// src/tasks/tasks.module.ts
import { Module } from '@nestjs/common';
import { TasksController } from './tasks.controller';
import { TasksService } from './tasks.service';
import { TypeOrmModule } from '@nestjs/typeorm';
import { Task } from './task.entity';

@Module({
    imports: [TypeOrmModule.forFeature([Task])],
    controllers: [TasksController],
    providers: [TasksService],
})
export class TasksModule {}
```

4. Crear el servicio TasksService

```
// src/tasks/tasks.service.ts
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { Task } from './task.entity';

@Injectable()
export class TasksService {
    constructor(@InjectRepository(Task) private tasksRepository: Repository<Task>) {}

    findAll(): Promise<Task[]> {
        return this.tasksRepository.find();
    }

    findOne(id: number): Promise<Task> {
        return this.tasksRepository.findOne(id);
    }

    async create(task: Partial<Task>): Promise<Task> {
        const newTask = this.tasksRepository.create(task);
        return this.tasksRepository.save(newTask);
    }

    async update(id: number, task: Partial<Task>): Promise<Task> {
        await this.tasksRepository.update(id, task);
        return this.tasksRepository.findOne(id);
    }

    async remove(id: number): Promise<void> {
        await this.tasksRepository.delete(id);
    }
}
```

5. Crear el controlador TasksController

```
// src/tasks/tasks.controller.ts
import { Controller, Get, Post, Put, Delete, Param, Body } from '@nestjs/common';
import { TasksService } from './tasks.service';
import { Task } from './task.entity';

@Controller('tasks')
export class TasksController {
constructor(private readonly tasksService: TasksService) {}

  @Get()
```

```
  findAll(): Promise<Task[]> {
    return this.tasksService.findAll();
  }

  @Get(':id')
  findOne(@Param('id') id: number): Promise<Task> {
    return this.tasksService.findOne(id);
  }

  @Post()
  create(@Body() task: Partial<Task>): Promise<Task> {
    return this.tasksService.create(task);
  }

  @Put(':id')
  update(@Param('id') id: number, @Body() task: Partial<Task>): Promise<Task> {
    return this.tasksService.update(id, task);
  }

  @Delete(':id')
  remove(@Param('id') id: number): Promise<void> {
    return this.tasksService.remove(id);
  }
}
```

6. Importar TasksModule en AppModule:

```
// src/app.module.ts
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { TasksModule } from './tasks/tasks.module';

@Module({
    imports: [
        TypeOrmModule.forRoot({
            type: 'mysql',
            host: 'localhost',
            port: 3306,
            username: 'your_username',
            password: 'your_password',
            database: 'your_database',
            entities: [Task],
            synchronize: true,
        }),
        TasksModule,
    ],
})
export class AppModule {}
```

7. Archivo main.ts

Asegúrate de que tu archivo `main.ts` esté configurado correctamente para iniciar la aplicación NestJS:

```
// src/main.ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  await app.listen(3000);
}
bootstrap();
```

Con esta implementación, tendrás un **CRUD** completo de tareas usando NestJS y TypeORM.

Puedes probar las rutas con una herramienta como Postman o curl para asegurarte de que estén funcionando correctamente.

- Para **obtener todas las tareas**: GET http://localhost:3000/tasks
- Para **obtener una tarea** específica: GET http://localhost:3000/tasks/:id
- Para **crear una tarea**: POST http://localhost:3000/tasks con un objeto JSON en el cuerpo de la solicitud que contenga las propiedades title y description.
- Para **actualizar una tarea**: PUT http://localhost:3000/tasks/:id con un objeto JSON en el cuerpo de la solicitud que contenga las propiedades que desees actualizar.
- Para **eliminar una tarea**: DELETE http://localhost:3000/tasks/:id

Recuerda reemplazar `:id` con el ID de la tarea que desees obtener, actualizar o eliminar.