Vielen Dank für die Zusammenfassung. Basierend auf der Beschreibung werde ich Ihnen einen einfachen Code-Beispiel geben, wie man ein User-Feature in NestJS mit TypeORM implementiert.

1. **Datenbankverbindung in app.module.ts**:

typescript

// app.module.ts

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { UserModule } from './user/user.module';

@Module({

imports: [

TypeOrmModule.forRoot({

type: 'mysql',

host: 'localhost',

port: 3306,

username: 'root',

password: 'password',

database: 'test',

entities: [\_\_dirname + '/\*\*/\*.entity{.ts,.js}'],

synchronize: true,

}),

UserModule,

],

})

export class AppModule {}

1. **User Entity**:

typescript

// user.entity.ts

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

@Entity()

export class User {

@PrimaryGeneratedColumn()

id: number;

@Column()

name: string;

@Column()

email: string;

}

1. **User DTOs**:

typescript

// user.dto.ts

// Request DTO

export class CreateUserDTO {

name: string;

email: string;

}

// Response DTO

export class UserResponseDTO {

id: number;

name: string;

email: string;

}

1. **User Module**:

typescript

// user.module.ts

import { Module } from '@nestjs/common';

import { TypeOrmModule } from '@nestjs/typeorm';

import { UserController } from './user.controller';

import { UserService } from './user.service';

import { User } from './user.entity';

@Module({

imports: [TypeOrmModule.forFeature([User])],

controllers: [UserController],

providers: [UserService],

})

export class UserModule {}

1. **User Service**:

typescript

// user.service.ts

import { Injectable } from '@nestjs/common';

import { InjectRepository } from '@nestjs/typeorm';

import { Repository } from 'typeorm';

import { User } from './user.entity';

@Injectable()

export class UserService {

constructor(

@InjectRepository(User)

private userRepository: Repository<User>,

) {}

async findAll(): Promise<User[]> {

return this.userRepository.find();

}

async create(user: User): Promise<User> {

return this.userRepository.save(user);

}

}

1. **User Controller**:

typescript

// user.controller.ts

import { Controller, Get, Post, Body } from '@nestjs/common';

import { UserService } from './user.service';

import { UserResponseDTO, CreateUserDTO } from './user.dto';

import { User } from './user.entity';

@Controller('users')

export class UserController {

constructor(private readonly userService: UserService) {}

@Get()

async findAll(): Promise<UserResponseDTO[]> {

return this.userService.findAll();

}

@Post()

async create(@Body() createUserDto: CreateUserDTO): Promise<UserResponseDTO> {

const user: User = new User();

user.name = createUserDto.name;

user.email = createUserDto.email;

return this.userService.create(user);

}

}

Dies ist ein einfaches Beispiel, wie man einen User-Service in NestJS mit TypeORM implementiert. Sie können es weiter ausbauen, indem Sie zusätzliche Funktionen, Validierungen und Fehlerbehandlungen hinzufügen.