Documentação Criação de Apis NestJs

1. Configurações Globais

Instalação do client NestJs

npm install -g @nestjs/cli

Criação do projeto

nest new nomeprojeto

Deletar controller e service generico inicial.

Criar pasta Modules: (Onde serão inseridos todos modulos do projeto)

Criar module de sessão

nest generate module backoffice

Importar libs padroes do projeto

npm install --save @nestjs/mongoose mongoose

npm install --save @nestjs/passport passport passport-http-bearer

npm install --save @nestjs/jwt passport-jwt

npm install --save md5-typescript

npm install --save cache-manager

npm i --save compression

npm install --save @nestjs/swagger swagger-ui-express

Importar Pastas Shared comuns dos projetos

common/enviropnment.ts = Variaveis globais do projeto

contracts/contracts.ts = Interface para implementar contratos de validacao

dtos/result.dto.ts = Resposta padrão da api

guards/auth.guards.ts = Padrão para guardar as rotas do sistema

interceptors/role.interceptors.ts = responsável por autorizar os usuários de

acordo com as regras de acesso (‘admin’ , ‘user’)

interceptors/validators.interceptors.ts = responsavel por implementar os contratos de validação

interfaces/jwt-payload.interface.ts = contem os dados que formam um token

services/auth.service.ts = criação e validacao to token

services/custom-logger.service.ts = logger personalizado para ver as funcoes do server

stategies/jwt-strategy.ts = Estrategia de validação do token

Importar pasta Utils

flunt.ts = Validações básicas do projeto ex: isEmail, maior q, menor q) usado nos contracts;

Começando o projeto…

main.ts

Colocando o Custom logger:

**const** app **=** **await** NestFactory.create(AppModule, {

logger: **new** CustomLogger(),

});

app.use(compression()); //Compressão de dados response mais rápida

Para gerar uma documentação automática da api:

**const** options **=** **new** DocumentBuilder()

.setTitle('PetShop API')

.setDescription('Api responsável por XPTO')

.setVersion('1.0.0')

.addTag('petshop')

.build();

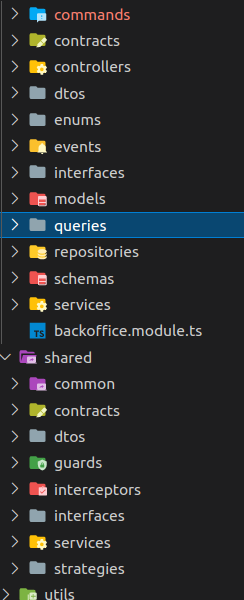
**const** document **=** SwaggerModule.createDocument(app, options);

SwaggerModule.setup('docs', app, document);

Trocando a porta do sistema estática pela variável global

**await** app.listen(environment.server.port);

CRIAR ESTRUTURA DE PASTAS DENTRO DO MODULO BACKOFFICE



IMPORTS DO BANCO DE DADOS

DEFINIR SCHEMA (IMPORTAR NO IMPORTS DO SCHEMA)

DEFINIR MODEL

CRIAR REPOSITORY (IMPORTAR PROVIDERS)

CRIAR DTOS

CRIAR CONTRACTS

CRIAR COMMAND E HANDLER (IMPORTAR PROVIDERS)

CRIAR SERVICE (IMPORTAR PROVIDERS)

CRIAR CONTROLLER RESPONSAVEL (IMPORTAR CONTROLLERS)

IMPORTAR NO CONTROLLER (CONTRATOS,DTOS,SERVICE,USE GUARDS, INTERCEPTOR RULES)