Projeto Nest.JS e Vue.JS

Alunos: Augusto Castejon, Álvaro Dias, Luíz Felippe e Vinicius Dogonski

1. Criação da Classe/Entidade Chaves
   1. No diretório backend. Dê o seguinte comando

→ nest generate module chave

O comando adiciona o módulo chave no módulo de aplicação principal.

* 1. Para criar um controlador básico, usamos o próximo comando:

→ nest generate controller /chave (TALVEZ ASSIM? )

→ nest generate controller /chave/controller/

Nosso controlador foi cadastrado no módulo chave

Um controlador é uma classe simples com o decorador @Controller(‘chave’)que é necessário para definir um controlador e especifica o prefixo chave, o que nos permite agrupar facilmente um conjunto de rotas relacionadas e minimizar o código repetitivo.

Para lidar com os diferentes métodos, o NestJS nos fornece métodos: @Get, @Post, @Put(), @Delete(), @Patch() e há outro decorador que lida com todos eles @All() .

Os serviços são importantes porque são responsáveis ​​​​pelo armazenamento e recuperação dos dados, o serviço foi projetado para ser utilizado pelo controlador.

* 1. Vamos criar um serviço básico com o próximo comando.

→ nest generate service /chave (TALVEZ ASSIM?)

→ nest generate service /chave/service/chave

* 1. Entre no diretório /emprestimo-chave/backend/src/chave

→ mkdir model

Agora dentro de /emprestimo-chave/backend/src/chave/model/ crie um arquivo chave.model.ts. esse arquivo será nossa classe e também será responsável pela integridade do banco garantindo sua estrutura.

→ touch chave.mode.ts

→ code .

Acesse o arquivo criado no passo anterior, crie a classe e configure o banco:

import \* as mongoose from 'mongoose';

export const ChaveSchema = new mongoose.Schema({

nome : {

type : String,

unique : true,

required: true

},

situacao : {

type : String,

required : true

},

status : Boolean

});

export interface Chave extends mongoose.Document {

nome: string;

situacao: string;

status: Boolean;

}

1. Criação da Classe/Entidade Servidor
   1. No diretório backend. Dê o seguinte comando

→ nest generate module servidor

O comando adiciona o módulo servidor no módulo de aplicação principal.

* 1. Para criar um controlador básico, usamos o próximo comando:

→ nest generate controller /servidor (TALVEZ ASSIM)

→ nest generate controller /servidor/controller

Nosso controlador foi cadastrado no módulo servidor

* 1. Vamos criar um serviço básico com o próximo comando.

→ nest generate service /servidor (TALVEZ ASSIM)

→ nest generate service /servidor/service/servidor

* 1. No diretório servidor crie um arquivo servidor.module.ts (TALVEZ NAO PRECISA)
  2. No diretório /emprestimo-chave/backend/src/servidor

→ mkdir model

Agora dentro de /emprestimo-chave/backend/src/servidor/model/ crie um arquivo servidor.model.ts. esse arquivo será nossa classe e também será responsável pela integridade do banco garantindo sua estrutura.

→ touch servidor.mode.ts

No VSCode acesse o arquivo criado no passo anterior, crie a classe e configure o banco:

import \* as mongoose from 'mongoose';

export const ServidorSchema = new mongoose.Schema({

nome: String,

cpf: String,

contato: String,

nascimento: Date,

status: Boolean,

});

export interface Servidor extends mongoose.Document {

nome: string;

cpf: string;

contato: string;

nascimento: Date;

status: Boolean;

}

1. Criação da Classe/Entidade Empréstimo
   1. No diretório backend. Dê o seguinte comando

→ nest generate module emprestimo

O comando adiciona o módulo emprestimo no módulo de aplicação principal.

* 1. Para criar um controlador básico, usamos o próximo comando:

→ nest generate controller /emprestimo (TALVEZ ASSIM)

→ nest generate controller /emprestimo/controller

Nosso controlador foi cadastrado no módulo emprestimo

* 1. Vamos criar um serviço básico com o próximo comando.

→ nest generate service /emprestimo (TALVEZ ASSIM)

→ nest generate service /emprestimo/service/emprestimo

* 1. No diretório /emprestimo-chave/backend/src/emprestimo

→ mkdir model

Agora dentro de /emprestimo-chave/backend/src/emprestimo/model/ crie um arquivo emprestimo.model.ts. esse arquivo será nossa classe e também será responsável pela integridade do banco garantindo sua estrutura.

→ touch emprestimo.mode.ts

No VSCode acesse o arquivo criado no passo anterior, crie a classe e configure o banco.

Em empréstimo, à lógica vai mudar um pouco, já que iremos relacionar o schema emprestimo a outros schemas, chave e servidor:

import \* as mongoose from 'mongoose';

import { Chave } from 'src/chave/chave.model/chave.model';

import { Servidor } from 'src/servidor/servidor.model/servidor.model';

export const EmprestimoSchema = new mongoose.Schema({

datahoraEmprestimo: Date,

datahoraDevolucao: Date,

status: Boolean,

chave: { type: mongoose.Schema.Types.ObjectId, ref: 'Chave' },

servidorRetirou: { type: mongoose.Schema.Types.ObjectId, ref: 'Servidor' },

servidorDevolvel: { type: mongoose.Schema.Types.ObjectId, ref: 'Servidor' },

});

export interface Emprestimo extends mongoose.Document {

datahoraEmprestimo: Date;

datahoraDevolucao: Date;

chave: Chave;

servidorRetirou: Servidor;

servidorDevolvel: Servidor;

status: Boolean;

}