

Projeto Nest.JS e Vue.JS

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

1. Criação da Classe/Entidade Chaves

1.1. No diretório backend. Dê o seguinte comando

→ `nest generate module chave`

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

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

→ `nest generate controller /chave`

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.

1.3. Vamos criar um serviço básico com o próximo comando.

→ `nest generate service /chave`

1.4. 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;
}
```

2. Criação da Classe/Entidade Servidor

2.1. No diretório backend. Dê o seguinte comando

- nest generate module servidor

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

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

- nest generate controller /servidor

Nosso controlador foi cadastrado no módulo servidor

2.3. Vamos criar um serviço básico com o próximo comando.

→ `nest generate service /servidor`

2.4. No diretório servidor crie um arquivo `servidor.module.ts` (TALVEZ NAO PRECISA)

2.5. 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.model.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;
}
```

3. Criação da Classe/Entidade Empréstimo

3.1. No diretório backend. Dê o seguinte comando

→ `nest generate module emprestimo`

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

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

→ `nest generate controller /emprestimo`

Nosso controlador foi cadastrado no módulo emprestimo

3.3. Vamos criar um serviço básico com o próximo comando.

→ `nest generate service /emprestimo`

3.4. 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.model.ts`

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

Em empréstimo, a 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;
}
```