07

# Faça como eu fiz: Integrando com o banco de dados

Primeiramente vamos refatorar o projeto, tornando-o específico para uma Livraria.

Feche o Visual Studio Code.

No terminal, dentro da pasta do nosso projeto vamos digitar o seguinte comando para sair e acessar a pasta que está um nível acima:

```
cd ..
```

COPIAR CÓDIGO

Agora vamos renomear a pasta `produtos-api` para `livraria-api` utilizando o comando:

```
mv produtos-api livraria-api
```

COPIAR CÓDIGO

Agora vamos entrar novamente na pasta do projeto, que acabou de ser renomeada:

```
cd livraria-api
```

COPIAR CÓDIGO

Podemos utilizar o comando `code .` para abrir o Visual Studio Code com a pasta do nosso projeto.

Dentro da pasta `src` vamos renomear os seguintes arquivos:

- produto.model.ts -> livro.model.ts
- produtos.controller.ts -> livros.controller.ts
- produtos.service.ts -> livros.service.ts

No arquivo `livro.model.ts` vamos renomear a classe `Produto` para `Livro`.

No arquivo `livros.service.ts`, vamos:

- Renomear a classe `ProdutosService` para `LivrosService`.

- Renomear a propriedade `produtos` para `livros`.
- No método `criar`, renomear o parâmetro `produto` para `livro`.
- No método `alterar`, renomear o parâmetro `produto` para `livro`.

No arquivo `livros.controller.ts`, vamos:

- Renomear a classe `ProdutosController` para `LivrosController`.
- Alterar o parâmetro do decorator `@Controller` de `produtos` para `livros`.
- Alterar o parâmetro do método `constructor` de `produtosService` para `livrosService`.
- No método `criar`, renomear o parâmetro `produto` para `livro`.
- No método `alterar`, renomear o parâmetro `produto` para `livro`.

No arquivo `package.json`, alterar o valor da propriedade `name` de `produtos-api` para

`livraria-api .`

No Insomnia também alterar em todas as URLs o path `/produtos` para `/livros .`

No terminal do programa "MySQL 8.0 Command Line Client":

Informar a senha do usuário `root .`

Digitar o seguinte comando para listar os esquemas de bancos de dados existentes:

```
show databases;
```

COPIAR CÓDIGO

Agora vamos criar o banco de dados para o nosso projeto:

```
create database livraria;
```

COPIAR CÓDIGO

Se digitarmos novamente o comando para listar os bancos de dados, o banco de dados `livraria` deve ser exibido:

```
show databases;
```

COPIAR CÓDIGO

Para usar o banco de dados, deve-se usar o seguinte comando:

```
use livraria;
```

COPIAR CÓDIGO

Para verificar as tabelas existentes, deve-se usar o comando:

```
show tables;
```

COPIAR CÓDIGO

Agora precisamos conectar o nosso projeto Nest com o banco de dados, para isso vamos utilizar uma ferramenta ORM (Mapeamento objeto-

relacional), ou seja algo que “traduza” o nosso modelo Produto em uma tabela e vice-versa. Iremos utilizar o Sequelize que é um ORM popular escrito em JavaScript que possui um “Wrapper” que permite utilizar anotações TypeScript.

Antes de utilizar precisamos instalar alguns pacotes npm, podemos fazer isso com os seguintes comandos:

```
npm install --save @nestjs/sequelize se
```

COPIAR CÓDIGO

```
npm install --save-dev @types/sequelize
```

COPIAR CÓDIGO

Com as dependências instaladas, podemos importar o módulo Sequelize no módulo principal da aplicação. No arquivo app.module.ts, vamos

adicionar o seguinte no array imports dentro do decorator @Module.

```
@Module({
  imports: [
    SequelizeModule.forRoot({
      dialect: 'mysql', // dialeto do b
      host: 'localhost', // endereço do
      port: 3307, // porta do banco de
      username: 'root', // usuário do M
      password: 'root', // senha do usu
      database: 'livraria', // nome do
      models: [] // por enquanto vamos
    })
  ],
```

COPIAR CÓDIGO

Uma vez que o módulo do Sequelize foi importado no módulo principal da aplicação, vamos injetá-lo na classe AppService para que ele fique disponível, para isso vamos criar um construtor no arquivo app.service.ts.

```
import { Injectable } from '@nestjs/com
import { Sequelize } from 'sequelize-ty

@Injectable()
export class AppService {
  constructor(private sequelize: Sequel
}
```

COPIAR CÓDIGO

Atenção: O Sequelize deve ser importado do módulo sequelize-typescript.

Com isso, quando nossa aplicação subir ela vai configurar o sequelize deixando-o pronto para ser usado.

No Sequelize um modelo define uma classe no banco de dados, para isso vamos fazer algumas anotações na classe `Livro`.

A primeira anotação será diretamente na classe, utilizaremos o decorator `@Table` para indicar que a classe `Livro` deve ser representada por uma

tabela no banco de dados. Este decorator é importado do módulo "sequelize-typescript".

Os modelos do Sequelize devem herdar da classe Model, então vamos adicionar isso à nossa classe extends `Model<Livro>`.

Como agora estamos herdando de `Model`, podemos eliminar a propriedade `id`, que será herdada e também não precisaremos mais de um construtor.

Além de marcar a classe como uma tabela, também devemos marcar as propriedades como colunas da tabela. Para isso utilizamos o decorator `@Column` que permite definir propriedades da coluna como, por exemplo, o tipo de dado, se o preenchimento obrigatório ou opcional, entre outras.

Nossa classe Livro ficou assim:

```
import { Column, DataType, Model, Table
```

```
@Table
export class Livro extends Model<Livro>
    @Column({
        type: DataType.STRING(60),
        allowNull: false,
    })
    codigo: string;


    @Column({
        type: DataType.STRING,
        allowNull: false,
    })
    nome: string;


    @Column({
        type: DataType.DECIMAL(10, 2),
        allowNull: false,
    })
    preco: number;
}
```

COPIAR CÓDIGO

Agora o nosso modelo está devidamente mapeado, precisamos avisar o Sequelize, para isso vamos adicioná-lo no array “models” nas configurações do SequelizeModule no arquivo app.module.ts.

- Antes: `models: [] // por enquanto vamos deixar em branco`
- Agora: `models: [ Livro ]`

O vscode está reclamando que a classe `Livro` não é do tipo esperado pelo array models do SequelizeModule.

Isso é causado por uma incompatibilidade entre o sequelize-typescript e a última versão do Sequelize, portanto, vamos fixar a versão do Sequelize como 5. Para isso, no arquivo package.json, dentro de "dependencies", vamos remover a linha "sequelize": "^6.3.5" e no terminal executar `npm install --save sequelize@5` .

Com isso o vscode para de reclamar. Obs: Algumas vezes pode ser necessário fechar o

vscode e abrir novamente para que de fato pare de reclamar.

Agora precisamos deixar a funcionalidade do Sequelize para o modelo Livro disponível para a aplicação, então vamos importá-lo no módulo principal. Nota: Em projetos maiores teríamos um módulo de livros e seria lá que essa importação seria feita, mas neste projeto didático temos apenas um módulo então é nele mesmo que faremos a importação, apesar de ficar um pouco estranho. Dentro do decorator `@Module`, vamos adicionar a seguinte linha:

```
SequelizeModule.forFeature([ Livro ])
```

COPIAR CÓDIGO