

FIC 2 – Módulo III

# Aula 1 - Manipulação de dados com Sequelize

Persistência de dados com  
Sequelize.  
Antonio Izo Júnior

# Sumário

- Mapeando uma tabela simples no sequelize.

# Mapeando uma tabela simples no sequelize.

- Na aula de hoje, aprenderemos a mapear uma tabela simples com sequelize.
- Utilizaremos o banco de dados **empresa.sqlite** mapeado na **aula 4 da semana 1.**



# Mapeando uma tabela simples no sequelize.

- Vamos criar a tabela “setor” no banco de dados via sequelize.
- A tabela “setor” possui quatro campos (idsetor, nome, ramal, email).

IDSETOR	NOME	RAMAL	EMAIL
1	FINANCEIRO	4254	FINANCEIRO@EMPRESA.COM
2	PORTARIA	4253	PORTARIA@EMPRESA.COM
3	SECRETARIA	4237	SECRETARIA@EMPRESA.COM



# Mapeando uma tabela simples no sequelize.

- Na criação da tabela de um banco de dados, precisamos colocar o nome do campo, suas restrições e o tipo de dados que ele receberá.
- As restrições podem ser: campo chave, não nulo ou nulo.
- Os principais tipos de dados incluem: integer (inteiro), varchar (texto) e date (data).



# Mapeando uma tabela simples no sequelize.

- Assim, a tabela “setor” pode ser representada da seguinte forma:

SETOR
idsetor integer not null primary key
nome varchar(40) not null
ramal varchar(10) not null
email varchar(30)

- Em vermelho temos os tipos de dados e, em verde, as restrições.

# Mapeando uma tabela simples no sequelize.

- Além dos campos e restrições, as tabelas possuem as operações para manipulação dos seus dados, que são **inserir registros, excluir registros, alterar registros e realizar pesquisas.**
- Vamos utilizar as regras do **mapeamento de objetos relacional.**
- Assim, o processo de manipulação de dados na tabela será muito mais prático e simples de entendimento.



# Mapeando uma tabela simples no sequelize.

Nossa tabela será criada como se fosse uma classe e terá como método as tarefas de manipulação de dados.

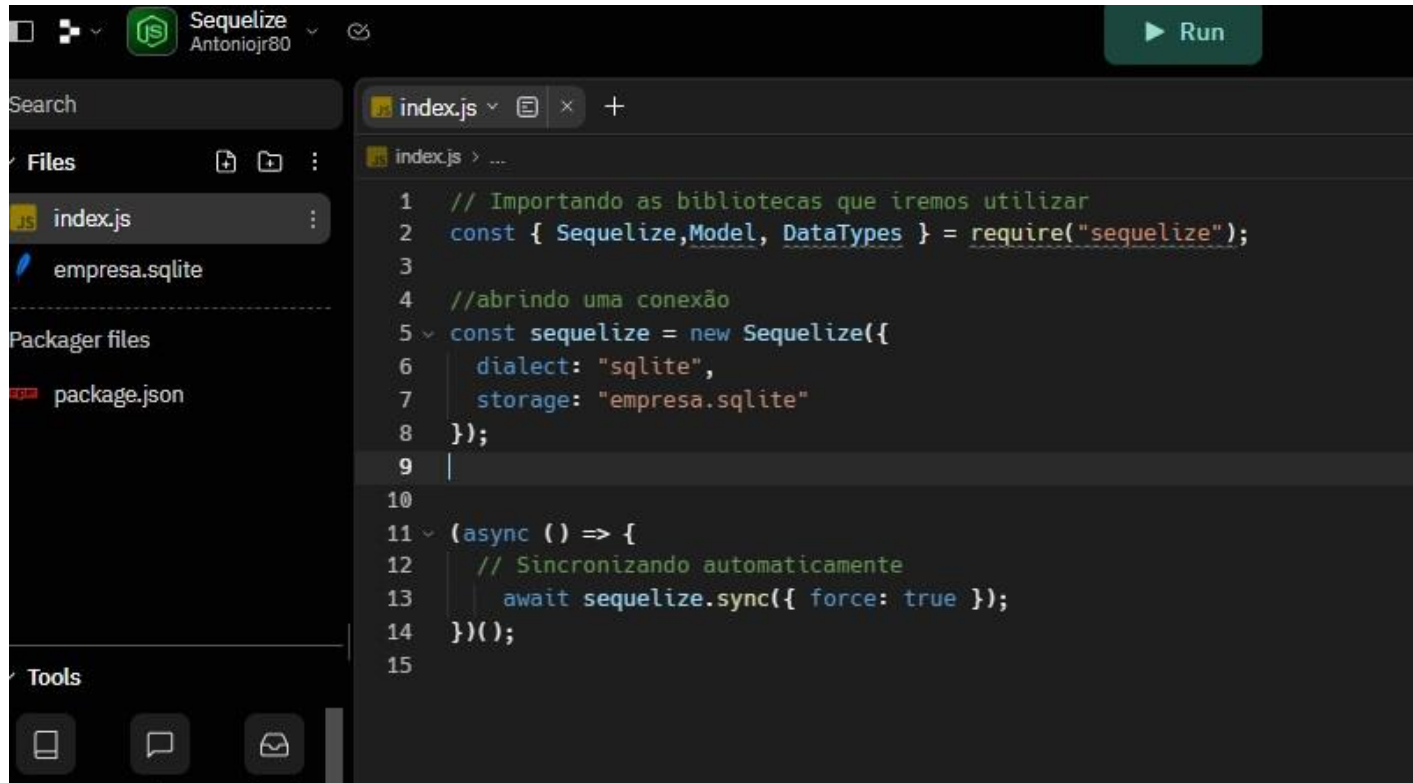
SETOR
idsetor <b>integer not null primary key</b> nome <b>varchar(40) not null</b> ramal <b>varchar(10) not null</b> email <b>varchar(30)</b>
Inserir registro() Excluir registro() Alterar registro()



# Vamos ver na prática como funciona?

# Mapeando uma tabela simples no sequelize.

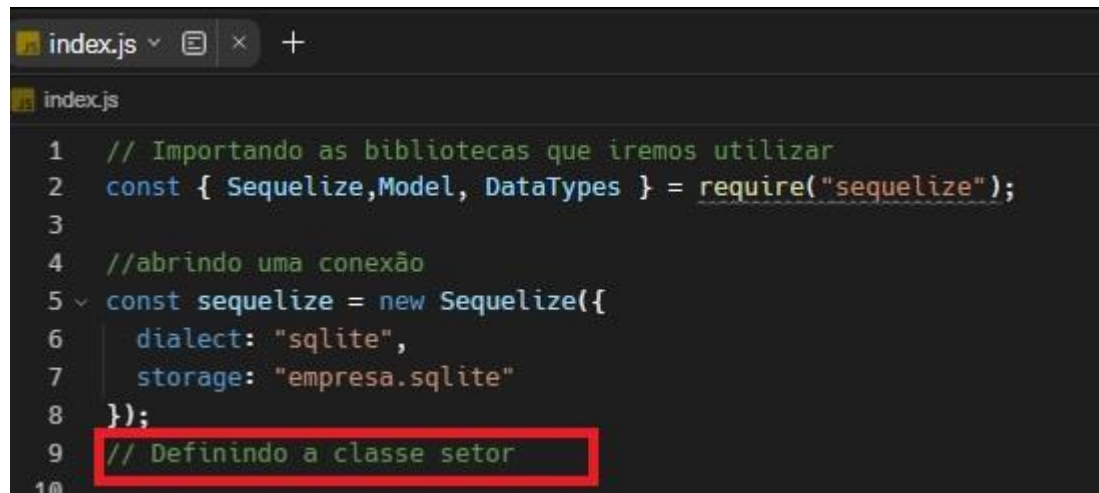
- **Passo 1:** Abra o projeto sequelize, criado na aula 4 da semana 1.



```
1 // Importando as bibliotecas que iremos utilizar
2 const { Sequelize, Model, DataTypes } = require("sequelize");
3
4 //abrindo uma conexão
5 const sequelize = new Sequelize({
6   dialect: "sqlite",
7   storage: "empresa.sqlite"
8 });
9
10
11 (async () => {
12   // Sincronizando automaticamente
13   await sequelize.sync({ force: true });
14 })();
15
```

# Mapeando uma tabela simples no sequelize.

- **Passo 2:** Após a criação do banco de dados e antes do async, começaremos a criar nossa classe Setor. No meu caso será na linha 9.



```
index.js
index.js
1 // Importando as bibliotecas que iremos utilizar
2 const { Sequelize, Model, DataTypes } = require("sequelize");
3
4 //abrindo uma conexão
5 const sequelize = new Sequelize({
6   dialect: "sqlite",
7   storage: "empresa.sqlite"
8 });
9 // Definindo a classe setor
10
```

# Mapeando uma tabela simples no sequelize.

- **Passo 3:** Iniciamos a declaração da nossa classe Setor.

class Setor extends Model {

```
1 // Importando as bibliotecas que iremos utilizar
2 const { Sequelize, Model, DataTypes } = require("sequelize");
3
4 //abrindo uma conexão
5 const sequelize = new Sequelize({
6   dialect: "sqlite",
7   storage: "empresa.sqlite"
8 });
9 // Definindo a classe setor
10 class Setor extends Model {
11
```

Criamos uma classe Setor estendida da biblioteca Model.

# Mapeando uma tabela simples no sequelize.

- **Passo 4:** Criamos a declaração do **init** com sequelize.

```
static init(sequelize) {  
    super.init({
```

```
// Definindo a classe setor  
class Setor extends Model {  
    static init(sequelize) {  
        super.init({
```

O comando **init** inicia o módulo sequelize **Create table.**

# Mapeando uma tabela simples no sequelize.

- **Passo 5:** Declaramos os nossos campos, os tipos de cada um e suas restrições.

## **Campo idsetor**

```
idsetor:{  
  type: DataTypes.INTEGER, //informamos o tipo do campo.  
  autoIncrement: true, // Campo auto incremento.  
  allowNull: false, // Quando false, o campo não aceitará nulo.  
  primaryKey: true // informa que o campo será chave primária.  
},
```



# Mapeando uma tabela simples no sequelize.

- **Passo 5:** Declaramos os nossos campos, os tipos de cada um e suas restrições.

## **Campo nome**

```
nome:{  
  type: DataTypes.STRING(40), //informamos o tipo do campo.  
  allowNull: false // Quando false, o campo não aceitará nulo.  
},
```



# Mapeando uma tabela simples no sequelize.

- **Passo 5:** Declaramos os nossos campos, os tipos de cada um e suas restrições.

## **Campo ramal**

```
ramal:{  
    type: DataTypes.STRING(10), //informamos o tipo do campo.  
    allowNull: false // Quando false, o campo não aceitará nulo.  
},
```





# Mapeando uma tabela simples no sequelize.

- **Passo 5:** Declaramos os nossos campos, os tipos de cada um e suas restrições.

## **Campo email**

```
email:{  
  type: DataTypes.STRING(30), //informamos o tipo do campo.  
  allowNull: false // Quando false, o campo não aceitará nulo.  
},
```

# Mapeando uma tabela simples no sequelize.

Nossa classe ficará conforme a figura abaixo.

```
// definindo a classe setor
class Setor extends Model{
  static init(sequelize){
    super.init({
      idsetor:{
        type: DataTypes.INTEGER,
        autoIncrement: true,
        allowNull:false,
        primaryKey: true
      },
      nome:{
        type: DataTypes.STRING(40),
        allowNull: false
      },
      ramal:{
        type: DataTypes.STRING(10),
        allowNull: false
      },
      email:{
        type: DataTypes.STRING(30)
      }
    }, {sequelize, modelName: 'setor', tableName: 'setores'})
  }
}
```

# Mapeando uma tabela simples no sequelize.

- **Passo 6:** Após, criamos o nome do nosso model e o nome da nossa tabela no banco de dados.

```
    { sequelize, modelName: 'setor', tableName: 'setores' })  
  }  
}
```

```
    email:{  
      type: DataTypes.STRING(30)  
    }, {sequelize, modelname: 'setor', tableName: 'setores'})  
  }  
}
```

# Mapeando uma tabela simples no sequelize.



# Mapeando uma tabela simples no sequelize.

- **Passo 6:** Para finalizar nossa criação da tabela, precisamos inicializar o **create table**.

Setor.init(sequelize);

```
    }, { sequelize, modelName: 'setor', tableName: 'setores' })  
  }  
}  
  
// inicializando o modelo create table  
Setor.init(sequelize);
```

# Mapeando uma tabela simples no sequelize.

- Agora é só executarmos o nosso arquivo e teremos nossa tabela “setor” criada.
- Na próxima aula, criaremos uma tabela fazendo relacionamento com outra tabela.

*Até lá...*



# Referências Bibliográficas

- ZHAO, Alice. **SQL Guia Prático: Um guia para o uso de SQL**. Editora Novatec, 2023.
- NEWMAN, Chris. **SQLite**. 1ª. Editora Sams, 2004.



**INSTITUTO  
FEDERAL**  
Espírito Santo