Criando a estrutura, conexão com o banco de dados e o servidor

Banco de dados

Dentro da pasta db, vamos criar um arquivo chamado **db.js**. Ele será responsável pela criação e conexão com o banco de dados.

Vamos fazer a criação do banco, da mesma forma que já fizemos durante o módulo.

O código é mostrado abaixo.

db.js

```
1  // BIBLIOTECAS/MODULOS UTILIZADOS
2  const Sequelize = require('sequelize');
3  //CRIANDO A CONFIGURAÇÃO DO BANCO DE DADOS
4  const sequelize = new Sequelize({
5    dialect: 'sqlite',
6    storage: './videoteca.sqlite'
7  })
8  //TRATANDO POSSÍVEIS ERROS E AUTENTICANDO NO BANCO
9  try {
10   sequelize.authenticate();
11   console.log("Banco de dados conectado com sucesso!");
12 }
13 catch (erro) {
14   console.log("Erro ao conectar ao banco",erro);
15 }
```

db.js

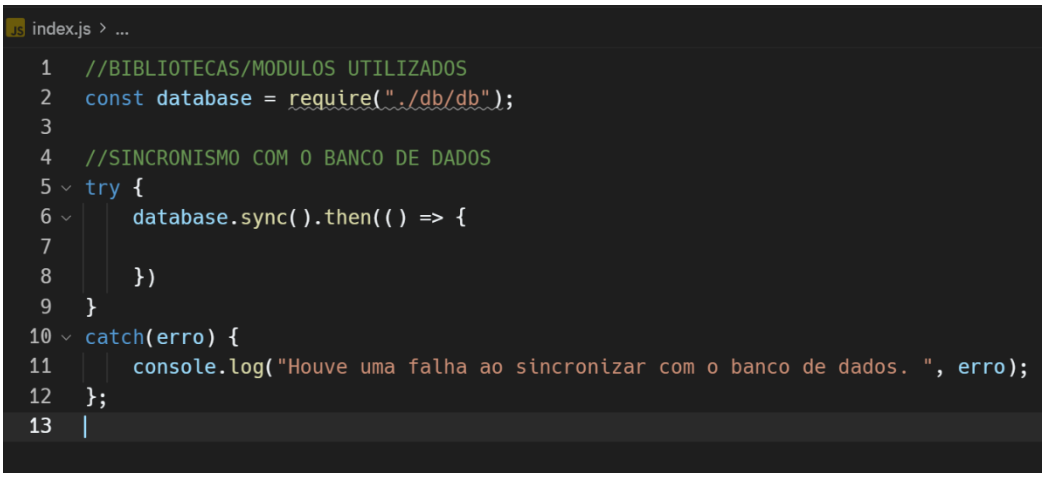
```
// BIBLIOTECAS/MODULOS UTILIZADOS
const Sequelize = require('sequelize');
//CRIANDO A CONFIGURAÇÃO DO BANCO DE DADOS
const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: './videoteca.sqlite'
})
//TRATANDO POSSÍVEIS ERROS E AUTENTICANDO NO BANCO
try {
  sequelize.authenticate();
  console.log("Banco de dados conectado com sucesso!");
}
catch (erro) {
  console.log("Erro ao conectar ao banco",erro);
}
module.exports = sequelize;
```

É bom sempre ressaltar a importância de colocar o try exception. Ele é responsável pela verificação de erros na hora da criação/conexão com o banco de dados.

Temos um comando novo no código que é o **sequelize.authenticate().** O comando é responsável pela autenticação no banco de dados. Ele é muito utilizado para passagem de parâmetros em bancos com autenticação.

Criando o index.js

Vamos iniciar a criação do nosso index.js, importando os módulos que iremos utilizar e logo após nosso sincronismo com o banco de dados.



Criamos um try exception e chamamos nossa constante **database.sync().** O sync realiza o sincronismo de forma síncrona ou seja, em tempo real.

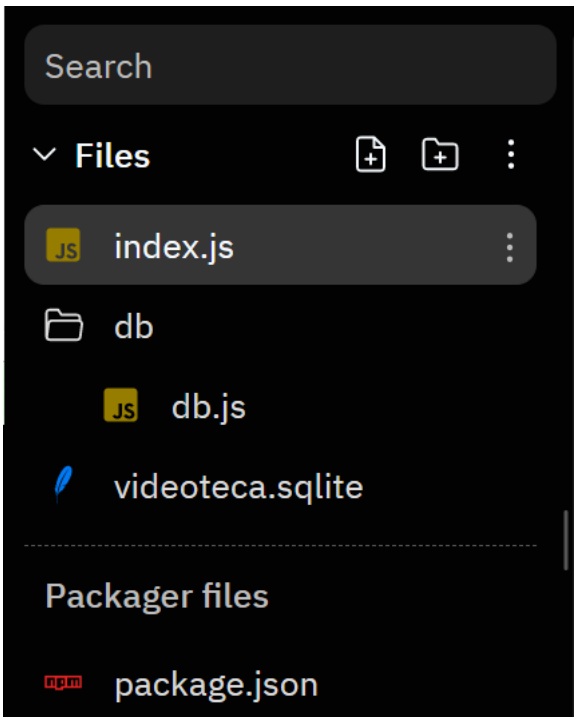
Codificação completa do index.js até o momento é mostrada abaixo.

```
//BIBLIOTECAS/MODULOS UTILIZADOS
const database = require("./db/db");

//SINCRONISMO COM O BANCO DE DADOS
try {
    database.sync().then(() => {

    })
}
catch(erro) {
    console.log("Houve uma falha ao sincronizar com o banco de dados. ", erro);
};
```

Vamos testar nosso código para ver se o banco está sendo criado.



Nosso banco foi criado. Até aqui nossa estrutura está funcionando de acordo com nosso projeto.

Criando o servidor Node.

Vamos aproveitar e criar nosso servidor.

Comece incluindo as duas linhas marcadas abaixo.

```
//BIBLIOTECAS/MODULOS UTILIZADOS
const database = require("./db/db");
const express = require("express");
const app = express();
//SINCRONISMO COM O BANCO DE DADOS
```

```
const express = require("express");
const app = express();
```

Criamos a constante express e informamos que utilizaremos o módulo express.

Após criamos uma constante app e instanciamos o express.

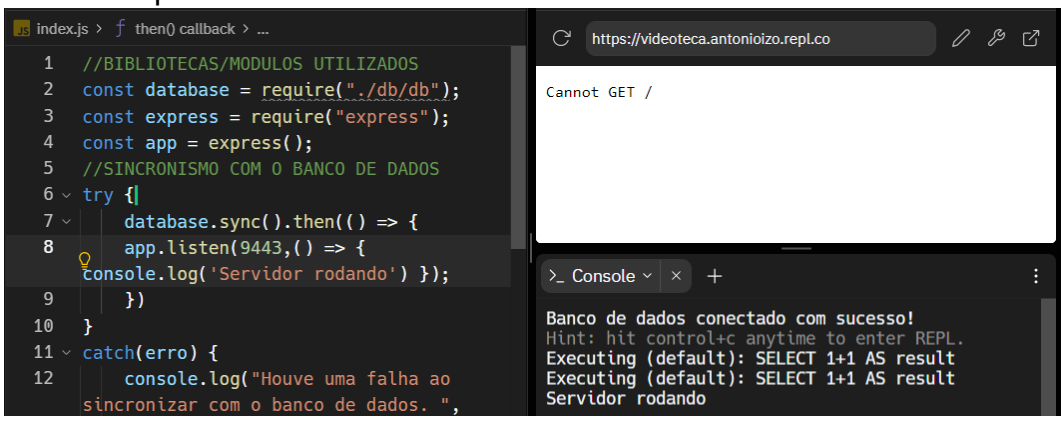
Agora vamos colocar nosso servidor para rodar dentro do nosso database.sync.

Assim toda vez, que realizarmos o sincronismo, ou seja enquanto estivermos executando nossa aplicação o servidor estará rodando.

```
try {
    database.sync().then(() => {
    app.listen(9443,() => { console.log('Servidor rodando') });
    })
}
```

```
app.listen(9443,() => { console.log('Servidor rodando') });
```

Já utilizamos algumas vezes o comando acima, porém agora estou colocando o valor da porta direto no listem sem criar uma constante.



Pronto, nosso servidor está funcionando.
Na próxima aula vamos criar a estrutura do model.

Até lá...