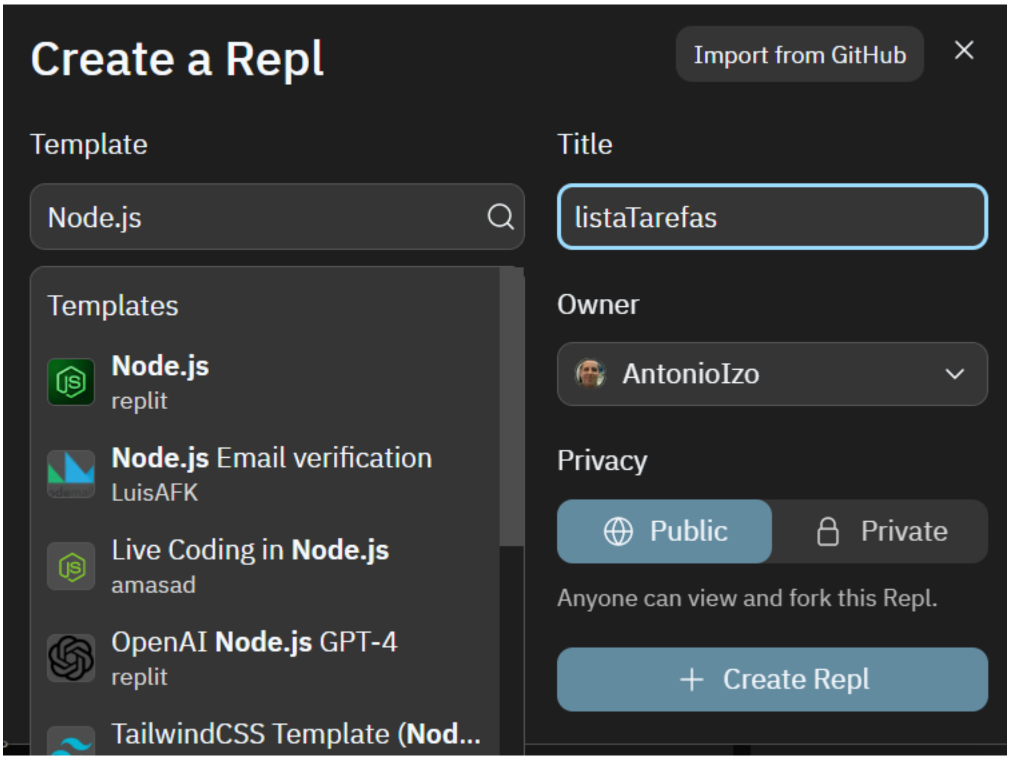
Criando a estrutura do servidor, model e banco de dados.

Nesta aula, vamos criar a estrutura do servidor os models e o banco de dados da API de um sistema de lista de tarefas que utilizaremos para exemplificar o uso de cookies.

Vamos começar criando uma API com a estrutura que já vimos em aulas passadas. Comece criando um novo projeto chamado **listaTarefas**.



Create a Repl

Template: Node.js

Title: listaTarefas

Owner: AntonioIzo

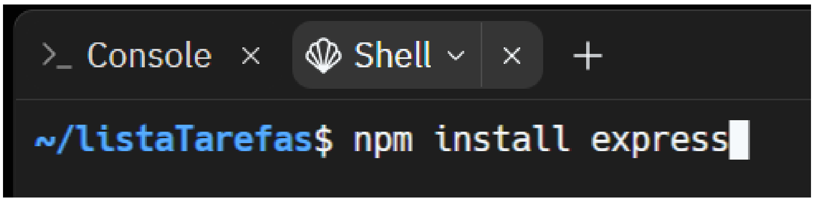
Privacy: Public

Anyone can view and fork this Repl.

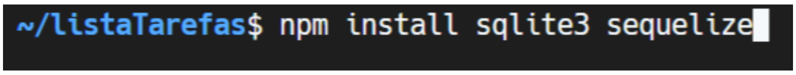
Vamos instalar nosso express, o sequelize e sqlite3.

npm install express

npm install sqlite3 sequelize



```
~/listaTarefas$ npm install express
```

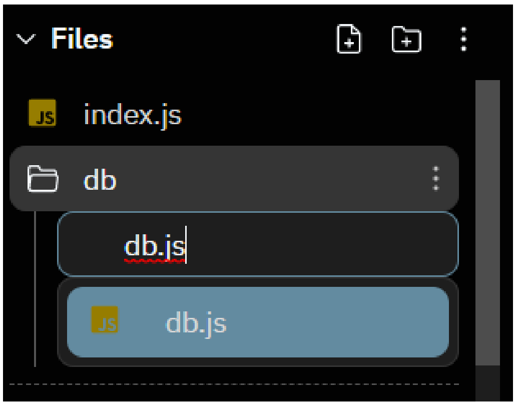


```
~/listaTarefas$ npm install sqlite3 sequelize
```

No arquivo index.js, crie o servidor da aplicação com a rota principal cadastrada.

```
const express = require("express");
const app = express();
app.get("/",(req, res) =>{
  return res.json({message: "Sistema de Lista de Tarefas"});
})
app.listen(3000);
```

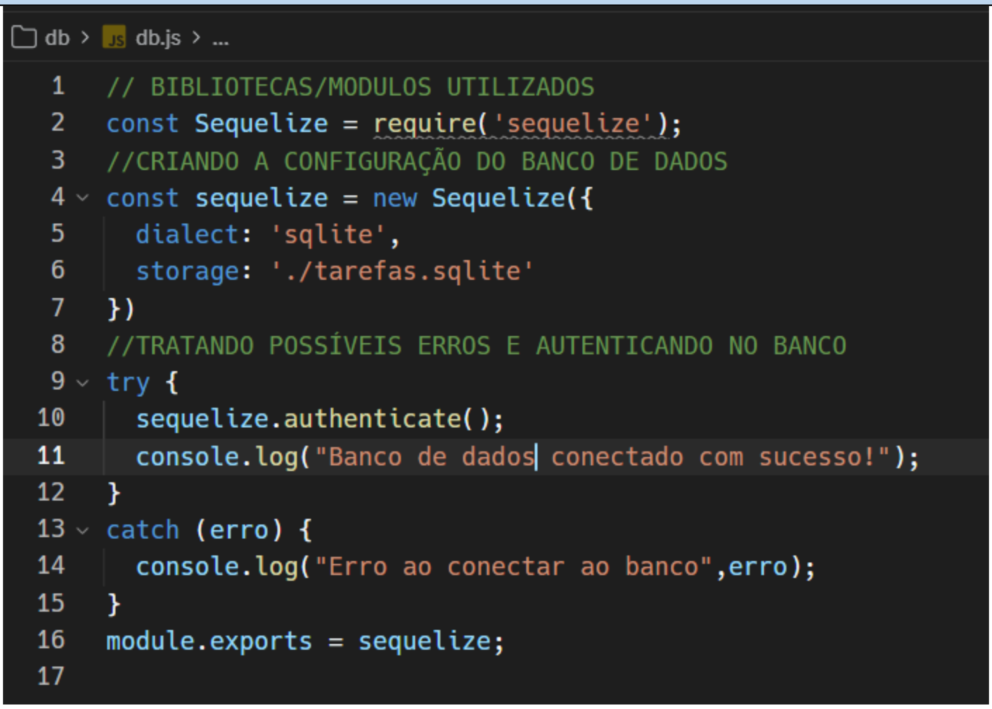
Crie agora a pasta db e dentro dela crie arquivo **db.js**



Vamos criar o banco de dados chamado tarefas.sqlite. O código de criação é mostrado abaixo.

```
// BIBLIOTECAS/MODULOS UTILIZADOS
const Sequelize = require('sequelize');
//CRIANDO A CONFIGURAÇÃO DO BANCO DE DADOS
const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: './mercado.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);
```

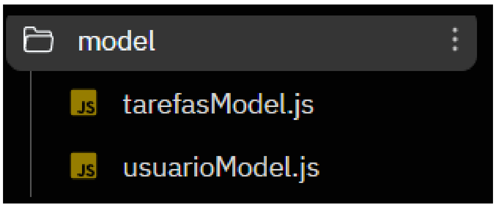
```
}
module.exports = sequelize;
```



```
// BIBLIOTECAS/MODULOS UTILIZADOS
const Sequelize = require('sequelize');
//CRIANDO A CONFIGURAÇÃO DO BANCO DE DADOS
const sequelize = new Sequelize({
  dialect: 'sqlite',
  storage: './tarefas.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;
```

Criando os Models da API

Vamos criar uma pasta chamada model e os models **tarefas** e **usuario** dentro dela para utilização na nossa API.



Model usuarioModel.js

A estrutura do model usuário será:

```
const Sequelize = require('sequelize');
const database = require('../db/db');
const Usuario = database.define('usuario', {
  id_usuario: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
```

```
    allowNull: false,
    primaryKey: true
  },
  nome:{
    type: Sequelize.STRING,
    allowNull: false,
  },
  email:{
    type: Sequelize.STRING,
    allowNull: false,
  },
  senha:{
    type: Sequelize.STRING,
    allowNull:false
  }
}, {database,modelname:'usuario',tableName: 'usuarios'})
module.exports = Usuario;
```

O model usuário terá somente os campos id_usuario, nome, email e senha.

Model tarefasModel.js

A estrutura do model tarefas será:

```
const Sequelize = require('sequelize');
const database = require('../db/db');
const Usuario = require('./usuarioModel');

const Tarefa = database.define('tarefa', {
  id_tarefa: {
    type: Sequelize.INTEGER,
    autoIncrement: true,
    allowNull: false,
    primaryKey: true
  },
  titulo:{
    type: Sequelize.STRING,
    allowNull: false,
  },
  descricao:{
    type: Sequelize.STRING,
```

```
    allowNull: false,
  }
}, {database,modelname:'tarefa',tableName: 'tarefas'})
module.exports = Tarefa;
```

Teremos no model tarefas os seguintes campos:

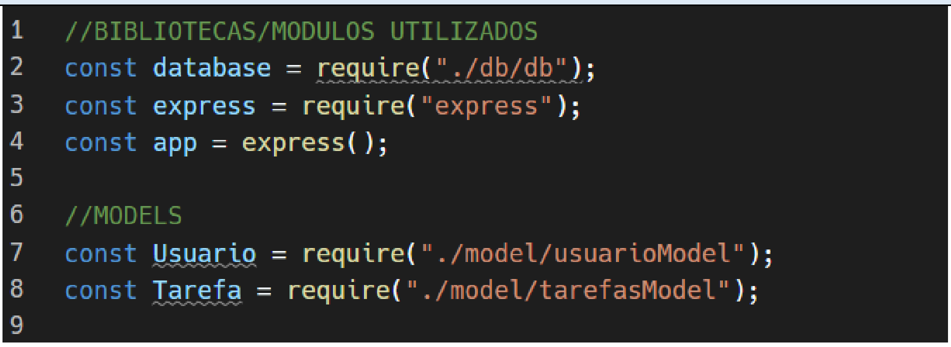
Id_tarefa, titulo e descrição.

O campo **id_usuario** é uma **chave estrangeira** da tabela usuário, iremos utilizar para saber qual usuário criou a tarefa.

Vamos incluir no index.js nossas models o módulo db.js e o sincronismo com o banco.

```
Const database = require(“./db/db”);
//MODELS
const Usuario = require("./model/usuarioModel");
const Tarefa = require("./model/tarefasModel");
```

```
try {
    database.sync().then(() => {
    })
}
catch(erro) {
    console.log("Houve uma falha ao sincronizar com o banco de dados. ", erro);
};
```

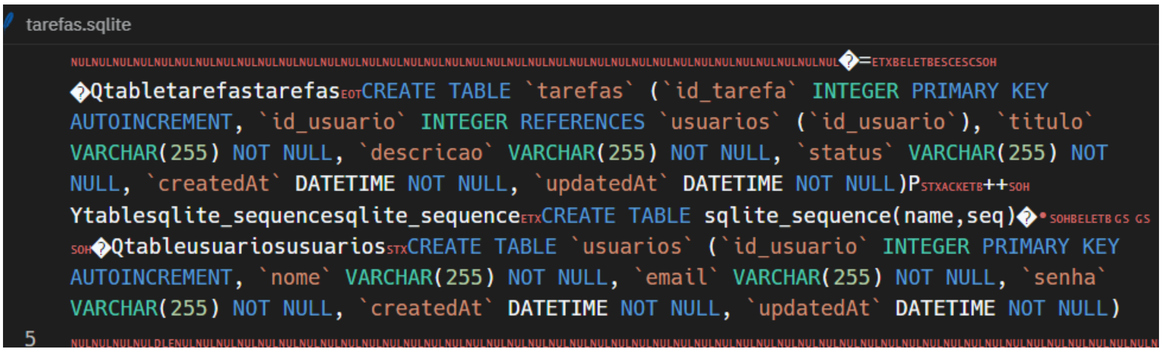


```
//BIBLIOTECAS/MODULOS UTILIZADOS
const database = require("./db/db");
const express = require("express");
const app = express();

//MODELS
const Usuario = require("./model/usuarioModel");
const Tarefa = require("./model/tarefasModel");

```

Executando nossa API teremos a criação do banco e das nossas tabelas usuários e tarefas.



```
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL�=ETXBELETBBESCESCSOH
�QtabletarefastarefasEOTCREATE TABLE `tarefas` (`id_tarefa` INTEGER PRIMARY KEY
AUTOINCREMENT, `id_usuario` INTEGER REFERENCES `usuarios` (`id_usuario`), `titulo`
VARCHAR(255) NOT NULL, `descricao` VARCHAR(255) NOT NULL, `status` VARCHAR(255) NOT
NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL)PSTXACKETB++SOH
Ytablesqlite_sequencesqlite_sequenceETXCREATE TABLE sqlite_sequence(name,seq)�•SOHBELETBGS GS
SOH�QtableusuariosusuariosSTXCREATE TABLE `usuarios` (`id_usuario` INTEGER PRIMARY KEY
AUTOINCREMENT, `nome` VARCHAR(255) NOT NULL, `email` VARCHAR(255) NOT NULL, `senha`
VARCHAR(255) NOT NULL, `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL)
NULNULNULNULDLENULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULN
```

Na próxima aula, vamos realizar a criação do arquivo de routes e dos controllers .

até a próxima aula.