Passo a Passo - API

Criar Projeto da API

Após fazer o *Open Folder* no VSCode, é necessário criar o projeto NodeJS com o comando abaixo. O comando cria um arquivo *package.json* responsável por orientar as configurações do projeto NodeJS.

```
npm init -y
```

Package.json

É necessário configurar o arquivo *package.json* adicionando dois campos: **type** e **start**:

```
{
  "name": "api",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "set PORT=3030 && nodemon ./src/index.js"     // START AQUI
  },
  "type": "module",   // TYPE AQUI
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "mysql2": "^2.3.0",
    "nodemon": "^2.0.12",
    "sequelize": "^6.6.5",
    "sequelize-auto": "^0.8.4"
  }
}
```

Configuramos o servidor web (api) para rodar na porta 3030. O comando **npm start** executará o projeto iniciando pelo arquivo *./src/index.js*.

A configuração **Type: "Module"** diz que será utilizado o mais novo recurso da linguagem Javascript para importação e exportação de objetos entre arquivos.

Instalar NPM's

```
npm i express cors nodemon sequelize sequelize-auto mysql2
```

Express: Permite a criação de um servidor web.
CORS: Habilita chamadas/requisições de domínios diferentes.
Nodemon: Permite que os arquivos sejam alterados enquanto o projeto estiver startado refletindo as alterações sem que seja necessário reiniciar o servidor.
Sequelize: ORM para comunicação com Banco de Dados pelo Javascript.
Sequelize-Auto, MySql2: Para mapear o banco de dados no Javascript.

Mapear Banco de Dados

O comando abaixo mapeará as tabelas do banco de dados em arquivos javascript. Esses arquivos ficarão em uma pasta chamada **models**.

```
npx sequelize-auto -h localhost -d NOME_DB -u NOME_USU -x SENHA -e mysql -o
models -l esm
```

Configurar Estrutura de Pastas

Para padronizar a comunicação dos objetos de nosso projeto, seguiremos a seguinte estrutura de pasta:

- src/
  - models/
  - db.js
  - index.js
- package.json

Programar o Arquivo db.js

A configuração consiste em importar o mapeamento gerado anteriormente e configurar as credenciais para acessar o banco de dados.

Ao final serão exportados os objetos que representam as tabelas, já preparados para uso.

```
import initdb from './models/init-models.js'

import Sequelize from 'sequelize';
const sequelize = new Sequelize(
  'NOME_BD',
  'NOME_USU',
  'SENHA', {
    host: 'ENDERECO_SERVIDOR',
    dialect: 'mysql',
    logging: false
});

const db = initdb(sequelize);
export default db;
```

Programar o Servidor WEB

A configuração consiste em importar o express, cors e db. A função express cria o objeto que representa o servidor web. Em seguida, atribuímos o CORS ao servidor web. Ao final, abrimos a porta do servidor web apontando para a porta configurada no arquivo *package.json*.

```
import db from './db.js';
import express from 'express'
import cors from 'cors'

const app = express();
app.use(cors());

//
// endpoints virão aqui
//

app.listen(process.env.PORT,
           x => console.log(`Server up at port ${process.env.PORT}`))
```

Programar os Endpoints

O primeiro endpoint será o verbo **GET** associado ao endereço **/chat** e receberá um parâmetro de rota **:salaId**. Esse parâmetro será utilizado para fazer o filtro no banco de dados que retornam mensagens de apenas uma sala.

A chamada é feita pela função **db.tb_chat.findAll(...)** que recebe um parâmetro contendo o filtro (where) e uma opção que indica que deverá fazer JOIN com outras tabelas e retornar seus campos também (include)

A função **resp.send(...)** é a responsável por ***retornar*** os registros recuperados do banco de dados para quem chamou a api (frontend/react).

```
//...

app.get('/chat/:salaId', async (req, resp) => {
    try {
        let mensagens = await
            db.tb_chat.findAll({
                where: {
                    id_sala: req.params.salaId
                },
                include: ['tb_usuario', 'tb_sala'],
            });

        resp.send(mensagens);
    } catch (e) {
        resp.send(e.toString())
    }
})

//...
```