02

Preparando o ambiente

Para esse curso, vamos continuar de onde paramos no curso anterior (https://cursos.alura.com.br/course/orm-nodejs-api-sequelize-mysql). Se você já fez o curso anterior e quiser continuar direto com o mesmo projeto, pode pular essa parte; mas se precisar ou preferir começar com um projeto novo, baixe o repositório (https://github.com/alura-cursos/1862-sequelize/tree/master) com a seguinte estrutura e arquivos iniciais:

```
.
├── api
│   ├── config
│   │   └── config.json
│   ├── controllers
│   │   ├── NivelController.js
│   │   ├── PessoaController.js
│   │   └── TurmaController.js
│   ├── index.js
│   ├── migrations
│   │   ├── 20200505131114-create-pessoas.js
│   │   ├── 20200526194618-create-niveis.js
│   │   ├── 20200526194804-create-turmas.js
│   │   └── 20200526194858-create-matriculas.js
│   ├── models
│   │   ├── index.js
│   │   ├── matriculas.js
│   │   ├── niveis.js
│   │   ├── pessoas.js
│   │   └── turmas.js
│   ├── routes
│   │   ├── index.js
│   │   ├── niveisRoute.js
│   │   ├── pessoasRoute.js
│   │   └── turmasRoute.js
```

```
│   └── seeders
│       ├── 20200505161755-demo-pessoa.js
│       ├── 20200601170039-demo-nivel.js
│       ├── 20200601170107-demo-turmas.js
│       └── 20200601170115-demo-matriculas.js
├── diagrama de banco - descricao da imagem.txt
├── Diagrama Relacional - escola de inglês.pdf
├── .gitignore
├── package.json
├── package-lock.json
├── requisitos.md
└── .sequelizerc
```

COPIAR CÓDIGO

Agora siga os seguintes passos de instalação:

Navegue pelo terminal até o diretório do projeto e instale as dependências com o comando `npm install`.

Se você ainda não criou um banco de dados local para trabalhar nesse projeto, vai precisar fazer isso agora. No curso usamos o MySQL; caso precise pode seguir as instruções que usamos no curso anterior (https://cursos.alura.com.br/course/orm-nodejs-api-sequelize-mysql) para criar um novo banco de dados chamado `escola_ingles` e conectar-se nele. Durante os cursos vamos fazer as consultas direto no terminal do MySQL, mas se você quiser pode usar algum cliente, como o MySQL Workbench.

Uma vez criado o banco, confira os dados no arquivo `api/config/config.json`:

```
{
 "development": {
   "username": "alura",  //utilize seu nome de user
   "password": "admin123", //sua senha, se existir
   "database": "escola_ingles",
```

```
    "host": "127.0.0.1",
    "dialect": "mysql",
    "operatorsAliases": false
  },
//restante do código
}
```

COPIAR CÓDIGO

Se for o caso, troque o nome e senha de usuário para os que você estiver usando em seu banco local. Se quiser usar outro banco que o Sequelize dê suporte, como SQLite, MSSQL Server, MariaDB ou PostgreSQL, modifique também essa informação conforme a documentação do Sequelize:

```
dialect: 'mysql' | 'mariadb' | 'postgres' | 'mssql'
```

COPIAR CÓDIGO

E faça a instalação da dependência:

```
$ npm install --save pg pg-hstore # Postgres
$ npm install --save mysql2
$ npm install --save mariadb
$ npm install --save sqlite3
$ npm install --save tedious # Microsoft SQL Server
```

COPIAR CÓDIGO

Rode os comandos de migração do Sequelize no terminal para criar as tabelas no banco: `npx sequelize-cli db:migrate` Você pode conferir se as tabelas foram criadas com sucesso através do comando `show tables;` no terminal do MySQL.

Você pode usar os arquivos de seed que estão na pasta `seeders` do projeto para popular as tabelas com dados de teste. Vamos usar bastante esses dados

durante o curso, então é super recomendável que você faça este passo para que o seu banco tenha dados pra serem trabalhados no projeto. Você pode usar o comando no terminal `npx sequelize-cli db:seed:all` .

Rode na pasta raiz do projeto o comando de terminal `npm start` para subir o servidor local. O Express fará a conexão em `localhost:3000/` , caso queira modificar a porta você pode alterar no arquivo `api/index.js` , que é o ponto de entrada da aplicação.

Faça o teste no Postman das rotas, por exemplo GET `localhost:3000/pessoas` para trazer do banco os registros da tabela Pessoas, além das demais rotas que estão na pasta `api/routes` .

Neste projeto utilizamos um linter para cuidar do estilo do código. Caso não queira utilizar, basta deletar a dependência e o arquivo `.eslintrc` da raiz do projeto:

```
"devDependencies": {
  "eslint": "^7.4.0", //delete esta linha
  "nodemon": "^2.0.4"
}
```

COPIAR CÓDIGO

Caso queira utilizar o Linter, faça a instalação e confira se o script do eslint está adicionado no `package.json` : "scripts": { "lint": "eslint api --fix", "start": "nodemon ./api/index.js "npm run lint"" },

Com o ambiente configurado, podemos seguir em frente!