[00:00] Olá! Boas-vindas ao curso de Sequelize e ORM com NodeJS e MySQL. Nesse curso, vamos escrever uma API para um sistema de controle de alunos e turmas de uma escola de inglês. Vamos partir de um diagrama de banco, onde as tabelas já estão dadas, e, a partir desses dados, escreveremos nossa API do zero.

[00:17] A diferença é que, dessa vez, usaremos um ORM, um mapeador de objeto relacional. Sequelize é o nome do ORM que usaremos. O Sequelize abstrai os comandos de operações de SQL, e faz com que possamos usar linguagem de programação que já estamos usando no backend para nos conectarmos e operarmos o banco. Ou seja, no nosso projeto, não iremos inserir queries de SQL dentro da aplicação que está escrita em JavaScript.

[00:40] Utilizaremos apenas JavaScript para nos conectarmos, passarmos instruções para o banco, fazermos as operações de CRUD, etc. Então, o JavaScript, através do Sequelize, vai se encarregar de traduzir as instruções para o SQL, para o banco, e fazer as operações necessárias.

[00:56] Usando o Sequelize, nossa aplicação fica mais legível, elegante, é mais fácil fazer a manutenção, mais padronizada. Então, vamos ver alguns tópicos que iremos abordar no nosso curso. Vamos conhecer o Sequelize, criar uma API do zero com algumas ferramentas que ele tem para isso. Essas ferramentas são uma mão na roda. Iremos ver como utilizar um ORM para conectar aplicação ao banco SQL, para gerenciar essas conexões, etc. utilizando apenas JavaScript.

[01:24] Iremos organizar a nossa aplicação no padrão MVC. Não é apenas organizar a aplicação, organizar os arquivos, mas também como aplicamos o conceito MVC no fluxo da aplicação. Vamos ver como utilizar as ferramentas do Sequelize e para fazer o CRUD junto ao banco e devolver dados no formato JSON.

[01:40] Você vai aproveitar melhor esse curso se já tiver feito os cursos básicos de criação de API com Node e SQL que temos na nossa formação aqui na Alura. Não entraremos em detalhes durante o curso a respeito do JavaScript, do ambiente do Node e como fazer a conexão inicial com banco e o CRUD.

[01:57] Muitas aplicações em Node utilizam Sequelize. Então, depois que você entender como o Sequelize funciona, conhecer alguns dos seus métodos, você já consegue desbravar projetos que utilizam esse ORM, descobrir mais ferramentas conforme a complexidade do projeto que você está trabalhando e também criar suas próprias APIs. Eu sou Juliana Amoasei, vou acompanhar você nesse curso. Então, vamos lá!

[00:00] Antes de efetivamente começarmos a trabalhar no projeto, o subiremos do zero, tem algumas ferramentas que precisam ser instaladas, se você já não as tiver instaladas no seu computador. Então, do que iremos precisar? De um SQL que seja suportado pelo Sequelize. No caso, aqui no curso, iremos usar o MySQL, mas existem alguns outros bancos que se pode usar com o Sequelize. Vou deixar uma lista deles na próxima sessão.

[00:28] Se você quiser usar algum outro, tudo bem. Aqui no curso, vamos usar o MySQL. O ORM, o Sequelize, serve justamente para isso, para podermos abstrair comandos de SQL. A única coisa que faremos direto no banco será criar a database, nos conectarmos com ela, fazer algumas consultas, para acompanhar o que está acontecendo.

[00:49] Nesse curso, precisaremos também saber sobre como criar e relacionar as tabelas do SQL. Se você preferir, pode dar uma conferida nesse curso de introdução ao SQL da Alura e podemos continuar a partir dele.

[1:04] Também criaremos algumas rotas e usar alguns métodos, alguns verbos de HTTP, como get, post, etc. o famoso CRUD. Se você precisa lembrar um pouco como isso funciona, também temos um curso aqui de HTTP que fala de todos esses métodos. Você pode dar uma relembrada se precisar.

[1:23] Faremos as consultas ao banco direto no terminal mesmo. Eu prefiro fazer dessa forma, mas se você quiser, pode usar o MySQL Workbench, que pode ser instalado com as ferramentas do SQL. Senão, você pode fazer no terminal mesmo, que é como eu vou fazer durante o curso. Já que iremos codar essa aplicação em JavaScript, precisaremos do ambiente do Node.

[01:48] Então, no curso estamos usando o Node versão 10.20 e o NPM versão 6.14. Por último, para testarmos as rotas e os métodos HTTP que usaremos para conseguir enviar requisições, receber respostas e testar tudo direito, usaremos o Postman, que está iniciando. Se você não tem o Postman instalado na sua máquina, você pode instalar, eu vou deixar o link. Ele é essa ferramenta que serve justamente para conseguirmos enviar e receber requisições HTTP para testar APIs.

[2:27] Se você já tem instalado, é só usar normalmente. Os links para instalação você encontra na próxima sessão. A instalação é de preparação de ambiente. Então, agora é a hora de colocar o nosso projeto em pé, criar os arquivos base dele e fazer algumas instalações.

[02:41] Eu já estou aqui na pasta juliana:~/Documents/alura/orm-sequelize$ onde vou criar o meu projeto. Então você pode fazer a mesma coisa do seu lado, que é navegar pelo terminal até a pasta do projeto e dar o comando npm init -y, que já vai criar um arquivo package.json com esqueleto base, do arquivo base de configurações do nosso projeto.

[03:01] Então já podemos deixar aberto inclusive o editor de código. Mas antes de começarmos a codar, existem algumas dependências que precisamos instalar de cara para começar a fazer qualquer coisa. Uma delas é o express, uma das bibliotecas que mais usamos no Node. No nosso caso, a usaremos para subir o servidor local e também para gerenciar as rotas que criaremos e usaremos na aplicação. A outra biblioteca que usaremos, por enquanto, é a Body Parser.

[03:37] Parser é converter, então a Body Parser converterá os dados que chegarão no corpo, no body das requisições, por exemplo, em uma requisição do tipo post. Então, em requisições do tipo post, receberemos alguns dados e essa biblioteca será responsável por converter para um tipo de dado que iremos trabalhar em nossa aplicação.

[03:58] No caso, estaremos trabalhando com dados do tipo Json. Eu falo json (pronúncia acentuada no “j”), tem gente que fala json (pronúncia acentuada no “son”), mas o importante é saber que sem a biblioteca, não conseguimos converter os dados para um tipo que consigamos usar; no caso, tipo Json.

[04:13] O próximo passo é criar um arquivo que seja o ponto de entrada da nossa aplicação. Vou criar uma pasta chamada API para separar os arquivos que trabalharemos. Os arquivos de configuração, da pasta de módulos, etc. Dentro da pasta API, criarei a pasta index.js. Podemos, inclusive, no pack de Json, mudar o caminho do nosso main de “main”: “index.js”, para “main”: “./api/index.js”,.

[04:44] Começaremos a subir nosso servidor local com express, começando por importar a biblioteca e também já importar o Body Parser.

[05:07] Para começarmos a usar o express, vou criar a const app const app =express ()e nela iniciaremos oexpress, deixando através daconst app` os métodos da biblioteca disponíveis para o resto do código.

[05:23] O primeiro que usaremos é o método use. Passaremos como parâmetro para ele o próprio Body Parser, e o método do Body Parser chamado Json, pelo comando app.use(bodyParser.json()). O método use serve para avisarmos o express que vai ter alguma biblioteca ou algum texto de código que vai, digamos assim, fazer um meio de campo entre as requisições e o próprio express.

[05:50] No caso desse comando, é o Body Parser que pegará todos os dados que chegarão via requisições do tipo post, já convertendo para Json e passando em diante para ser usado pelo restante da aplicação.

[06:05] Nosso servidor local precisará de uma porta. Estou passando na constante porta a porta 3.000, por meio do comando const port = 3000. É um número que normalmente usamos como padrão nas aplicações que subimos localmente com o express.

[06:19] Agora, podemos criar uma rota de teste para fazermos um primeiro teste com nosso servidor. O método .get() do express se refere ao mesmo get do método de HTTP, tem o mesmo nome, e passaremos para ele dois parâmetros. O primeiro parâmetro é uma string com a rota que testaremos. Vamos dar para ela o nome de teste mesmo. E o segundo parâmetro é uma função callback que receberá requisição e resposta como parâmetros. O comando é app.get(‘/teste’, (req, res) => ).

[06:54] Então o express vai receber uma requisição e vai passar para frente uma resposta com algumas informações. Uma dessas informações será o status da requisição. Passaremos o status 200, que é o ok.

[07:11] Em seguida, vamos também enviar uma informação de volta. Essa informação no método .send, de envio, pode ser uma array, pode ser um objeto. Vamos passar um objeto então que carregue .send( {mensagem: ‘boas-vindas à API’ }), por exemplo, só para testarmos e ver se está tudo ok.

[07:44] Vamos pedir também para o express para ele ficar ouvindo, listening, ouvindo o nosso servidor para nos dizer se está tudo ok, se tem algum erro, se o servidor está de pé, etc. O método listening ele vai receber dois parâmetros. O primeiro é o número da porta onde ele tem que ficar ouvindo, e o segundo parâmetro é mais um callback e nele, passaremos só um console.log, que ficará no terminal nos dizendo se está tudo ok ou não.

[08:16] Vamos passar app.listen (port, () => console. Log(servidor está rodando na porta ${port}). Estou usando os recursos do template string para conseguir colocar essa variável dentro da string usando crases, e não aspas simples.

[08:41] Antes de testarmos, vamos exportar module. exports = app para ficar disponível para o restante da aplicação.

[08:49] E agora podemos vir no nosso terminal chamar o Node. E para rodar com o Node, a API/index, e vamos ver o que acontece.

[09:01] Servidor está rodando na porta 3.000. Isso significa que a parte app.listen(port, () => console.log(servidor está rodando na porta ${port})) está ok, está ouvindo e está tudo bem. E agora vamos no Postman para testarmos a rota. Essa rota do tipo get. Então, uma rota que está rodando localmente na porta 3000. Eu já tinha testado antes aqui no Postman, então vamos fazer um novo teste. Então <http://localhost:3000/teste> que é a rota que estamos testando. mensagem: “boas-vindas à API”. Está tudo funcionando, por enquanto, perfeitamente. Tem várias alterações que podemos fazer.

[09:37] Vamos dar uma melhorada para deixar nosso servidor um pouco mais prático. Então, vamos ver em seguida com o restante das instalações.

Para este curso, precisaremos instalar algumas ferramentas - caso você ainda não tenha feito isso anteriormente:

**MySQL**: é o banco SQL que vamos utilizar para guardar os dados da nossa API. Temos um [tutorial com o passo a passo para instalação no **Windows**](https://www.alura.com.br/artigos/mysql-do-download-e-instalacao-ate-sua-primeira-tabela) e alguns comandos iniciais. Para [instalação no MacOS, baixe o arquivo DMG](https://downloads.mysql.com/archives/community/) e [siga as instruções de instalação](https://dev.mysql.com/doc/mysql-osx-excerpt/8.0/en/osx-installation-pkg.html). Se estiver usando Linux (Debian/Ubuntu), você pode [baixar direto via terminal](https://ubuntu.com/server/docs/databases-mysql) com sudo apt install mysql-server.

**MySQL Workbench**: durante o curso faremos as consultas ao banco direto pelo terminal, porém se você quiser, pode fazer isso usando um programa como o [MySQL Workbench](https://dev.mysql.com/downloads/workbench/).

**NodeJS**: Se você já fez outros cursos dessa formação, o NodeJS já deve estar instalado. você pode conferir com o comando node --version ou node -v no terminal, caso retorne um número de versão, como por exemplo, v10.16.0, o NodeJS já está instalado. Caso precise instalar, os links estão na [página oficial do NodeJS](https://nodejs.org/en/download/). Este curso foi desenvolvido usando a versão 10.20.0.

**Postman**: É o ambiente onde faremos os testes da nossa API, enviando requisições e recebendo respostas. Caso não tenha instalado, pode baixar na [página oficial do Postman](https://www.postman.com/downloads/). Neste curso utilizamos a versão 7.21.2.

O restante das instalações vamos fazer durante o desenvolvimento do projeto.

Além das instalações, deixamos alguns arquivos prontos, que serão utilizados durante o projeto: o diagrama relacional que iremos usar como guia para criarmos as tabelas do banco e também alguns arquivos .js. Estes arquivos vão servir para popular as tabelas do banco com dados de teste, e que disponibilizamos para que você possa aproveitar melhor o conteúdo do curso sem ter que se preocupar com isso.

Você pode [baixar a pasta do projeto](https://github.com/alura-cursos/orm-sequelize-projeto/archive/aula-0.zip) e deixar disponível na pasta do seu projeto, e vamos indicar durante o curso quando utilizar os arquivos. O diagrama de banco que vamos utilizar como referência pode ser baixado no mesmo link.

Agora podemos começar!

O processo de instalação e uso do MySQL no Windows é um pouquinho diferente do Linux e MacOs, mas temos um [tutorial](https://www.alura.com.br/artigos/mysql-do-download-e-instalacao-ate-sua-primeira-tabela) que pode te ajudar no processo.

Após a instalação o guia continua com a criação de tabelas direto no MySQL. **Você não precisa fazer essa parte agora para continuar com o curso de ORM com Sequelize**, porém se quiser e tiver curiosidade, vai fundo!

No momento em que escrevemos este curso, a última versão do MySQL é a 8.0. Ao fazer a instalação completa, também será instalado o **MySQL Command Line Client**, um cliente de conexão via terminal do próprio MySQL. A partir dele você pode se conectar ao banco e executar as queries como fazemos no vídeo.