Aula - Knex.js

Materiais Complementares

- PT
- EN

Materiais de Aula

- Slides
- Codando Junto
- Template para os exercícios propostos
- Exercícios feitos em aula
  - exercicio-knex.zip 14649.4KB
- Gravação da aula

  vimeo.com

  https://vimeo.com/groups/766170/videos/746704561

- Inicie o projeto por aqui

Enunciado

Nesta semana, trabalharemos na combinação dos conceitos de SQL, banco de dados e integração com o Backend. Para isto, iremos construir ao longo da semana um sistema de registros básicos de funcionários e também de projetos de uma dada empresa de desenvolvimento.

No exercício de hoje vamos iniciar a integração do banco de dados ao Backend. Para isto, trabalharemos na construção de endpoints de usuários (funcionários) cadastrados nas aulas anteriores.

Exercícios Propostos

Nessa atividade utilizaremos as tabelas criadas na segunda e terça.

Utilizem o template a seguir para iniciar a atividade:

template-knex.zip 24.8KB

Exercícios de Desenvolvimento de Código

Utilizem o template como base de estruturação do seu código e desenvolva os endpoints definidos a seguir, aplicando os conceitos de Knex vistos em aula.

Observação 1: Não se esqueça de rodar npm install e verificar se todas as dependências estão corretas.

Observação 2: Não se esqueça de inserir seus dados de acesso ao banco no arquivo .env .

Exercício 1

Crie um endpoint de busca de usuários cadastrados. Este endpoint deve permitir a busca por partes do nome. Caso nenhum valor de busca seja recebido, retornamos todos os usuários.

Entradas → 'search' é uma variável opcional de busca.

Validação de Input → Nenhuma.

Regras de Negócios → Nenhuma.

Saídas → Erro de requisição, lista de usuários selecionados ou toda a lista.

Dicas:

- Utilize o connection para estabelecer a conexão com o banco de dados.
- 'search' é uma query opcional.

Exercício 2

Desenvolva um endpoint que cria um novo usuário e retorna as informações do mesmo. A id do usuário deve ser gerada automaticamente pela API.

Entradas → name e email do usuário.

Validação de Input:

- name e email devem existir.
- name e email devem ser do tipo string.
- O email deve possuir o caractere @

Regras de negócio:

- O email é único para cada usuário.
- O nome do usuário deve ter ao menos 4 caracteres.
- A id é gerada automaticamente.

Saídas possíveis:

- Cada erro deve retornar o seu respectivo status code e uma mensagem descrevendo a situação.
- Para sucesso, deve retornar o status de criação, mensagem de sucesso e o usuário atualizado.

Dicas:

- Utilize o connection para estabelecer a conexão com o banco de dados.
- Para validar um e-mail no formato definido, você pode utilizar o método .includes() para verificar se existem os caracteres "@"(arroba) no e-mail enviado pelo client. Se estiver confortável tente utilizar um RegEx (Regular Expression) de validação de e-mail.
- Para não se preocupar em contar o tamanho da lista ao criar uma id automatizada, utilize o Date.now().
- Para conferir o registro do novo usuário, utilize a requisição do exercício 1.

Exercício 3

Crie um endpoint que edita o e-mail de um determinado usuário.

Entradas → id e novo email do usuário.

Validação de Input:

- O email deve existir.
- O email deve ser do tipo string.
- O e-mail deve possuir o caractere @

Regras de negócio:

- Se o id fornecido não corresponder a um usuário existente, um erro deverá ser exibido.
- O email é único para cada usuário.

Saídas possíveis:

- Cada erro deve retornar o seu respectivo status code e uma mensagem descrevendo a situação.
- Para sucesso, deve retornar o status de edição e mensagem de sucesso.

Dicas:

- Utilize o connection para estabelecer a conexão com o banco de dados.
- Para validar um e-mail no formato definido, você pode utilizar o método .includes() para verificar se existem os caracteres "@"(arroba) no e-mail enviado pelo client. Se estiver confortável tente utilizar um RegEx (Regular Expression) de validação de e-mail.
- Para conferir o registro da alteração de dados, utilize a requisição do exercício 1.

Exercício 4

Construa um endpoint que deleta um determinado usuário.

Entradas → id do usuário a ser deletado.

Validação de Input → Nenhuma.

Regras de negócio:

- Se o id fornecido não corresponder a um usuário existente, um erro deverá ser exibido.

Saídas possíveis:

- Cada erro deve retornar o seu respectivo status code e uma mensagem descrevendo a situação.
- Para sucesso, deve retornar o status de remoção e mensagem de sucesso.

Desafio

Caso tenha conseguido concluir todas as atividades propostas anteriores, crie uma documentação para a API, explicitando informações relevantes para uso e aplicação.

Como documentar APIs no Postman

https://vimeo.com/727469710/a763bea649