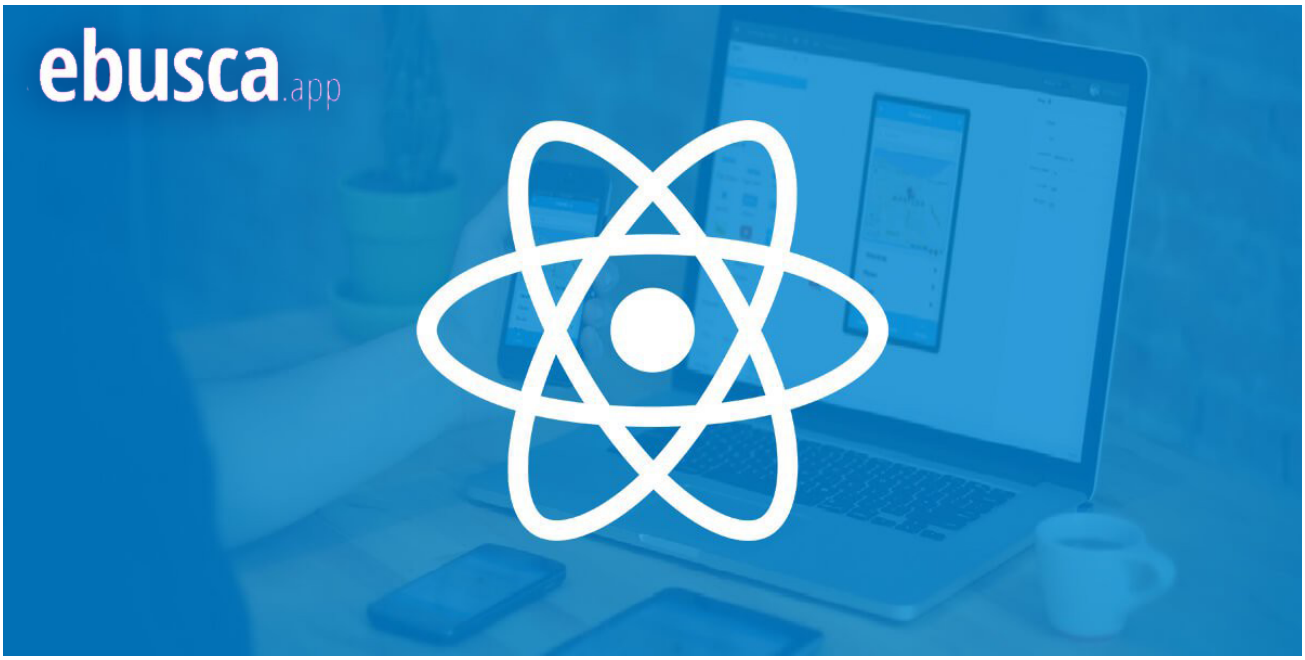


React Native: Como pegar dados do banco de dados MySQL utilizando Express (Rest API)

Publicado em 24 de janeiro de 2019

Saulo Joab Penha Simplicio
Estagiário na Enel Brasil
2 artigos + Seguir

ATENÇÃO: *Esse artigo é um pouco antigo e eu escrevi quando não tinha muito conhecimento sobre NodeJS e afins. Ele ainda funciona normalmente, mas eu não detalho muito as coisas.*

React Native é uma ferramenta maravilhosa que a gente vem usando pra desenvolver o ébusca.app. Uma coisa que temos percebido, porém, é que a comunidade brasileira ainda não é tão grande. Uma das principais fontes de informação sobre RN aqui no Brasil é a Rocketseat, eles são bem ativos tanto no site deles quanto no Discord. Recomendo dar uma olhada.

Porém, devido à essa falta de comunidades brasileiras falando sobre React Native, pode ser bem complicadinho achar tutoriais que ensinam à fazer certas coisas que parecem básicas. Como uma conexão com banco de dados MySQL, por exemplo. Então resolvi escrever esse artigo pra ajudar algumas pessoas com isso. *Allons-y!*

Passo 0: Importando os packages.

Primeiro, no Prompt de Comando, navegue até a pasta do seu pr

React Native, e use os seguintes comandos:

```
npm install express
npm install body-parser
npm install mysql
```

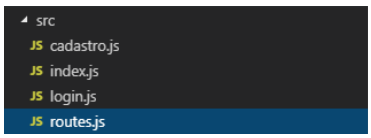
Talvez isso demore um pouquinho, mas é assim mesmo. E não se preocupe, eu vou explicar o que cada package desse faz.

Passo 1: Crie seu banco MySQL.

Essa parte é a mais *óbvia* de todas, não tem muito o que falar. Primeiro, crie o banco de dados MySQL localmente no seu computador (caso não saibam fazer isso, eu posso fazer um outro artigo explicando). Para isso, eu uso o Wamp. Sim, eu uso Windows, btw. Não me julguem :p

Passo 2: Routes.js.

No seu projeto **React Native**, crie um arquivo chamado *'routes.js'*. Esse arquivo irá gerenciar nossas requisições POST e GET, que utilizaremos pra **puxar** ou **inserir** dados no nosso banco de dados. Será nesse arquivo também que faremos a conexão com o banco. Eu recomendo criar uma pasta '*src*' e colocar ele lá dentro. Essa é minha pasta:



Passo 3: Conexão com o banco de dados.

Agora o negócio começa a ficar **doido** (num bom sentido, claro). Primeiro, vou mostrar o código e depois explico o que cada coisinha tá fazendo.

```
// Isso aqui é no arquivo routes.js!

const express = require('express');
const bodyParser = require('body-parser');
const mysql = require('mysql');

const connection = mysql.createPool({
  host     : 'localhost',
  user     : 'me',
  password : 'secret',
  database : 'my_db'
});
```

Ok. Bonito, né não? Vamos por partes. As três primeiras linhas são linhas de importação. Na primeira, estamos importando o Express. Ele é um framework Node, usado no **back-end**, que faz esse negócio das rotas que eu venho falando. Ele é ótimo pra interagir com API's.

Na segunda linha, importamos o Body Parser. De forma resumid

que vai deixar a gente acessar o Body das nossas requisições e usar os dados que forem inseridos.

Da terceira linha em diante, a gente tá importando o módulo do MySQL e estamos criando uma conexão com o banco de dados. Aí você vai colocar os dados do seu banco de dados (no pun intended). Siga o exemplo abaixo:

```
const connection = mysql.createPool({
  host     : 'localhost', // O endereço da conexão (localhost).
  user     : 'root',     // O nome de usuário do banco.
  password : '',        // A senha do banco.
  database : 'my_db'   // O nome do seu database.
});
```

Passo 4: Pegando dados de uma tabela.

Ok, agora pode ser que fique um pouco mais complicadinho (eu espero que não). Basicamente agora vamos pegar os dados da uma tabela do banco de dados. Primeiro, vou mostrar o código e depois digo o que tá acontecendo.

```
// Isso ainda é no routes.js! Logo abaixo dos outros códigos.
// Iniciando o app.
const app = express();

// Criando uma rota GET que retorna os dados da tabela usuários.
app.get('/users', function (req, res) {
    // Conectando ao banco.
    connection.getConnection(function (err, connection) {

    // Executando a query MySQL (selecionar todos os dados da tabela usuár
    connection.query('SELECT * FROM users', function (error, results, fiel
      // Caso ocorra algum erro, não irá executar corretamente.if (error)

      // Pegando a 'resposta' do servidor pra nossa requisição. Ou seja, a
      res.send(results)
    });
  });
});

// Iniciando o servidor.
app.listen(3000, () => {
 console.log('Vai no navegador e entra em http://localhost:3000/users pra
});
```

Eu comentei o código pra facilitar um pouco. Primeiro, iniciamos nosso **App Express**. Depois, a gente cria uma rota **GET** chamada de '*users*', e essa rota vai executar uma query MySQL que vai selecionar todos os dados da tabela '*users*' pra gente. O nome da tabela e da rota não precisam ser iguais, só coloquei pra facilitar. Depois disso, a gente inicia nosso servidor na porta 3000.

Tá. Agora, como fazemos pra rodar isso? Então, pra fazer isso, você **não precisa rodar seu App do React Native ainda**. Você vai abrir o Prompt de Comando, navegar até o arquivo routes.js e executar somente ele. Pra fazer isso, use o comando 'node routes.js', igual eu fiz abaixo:

Não foi fornecido texto alternativo para esta imagem

Como você pode ver, abaixo apareceu uma mensagem mandando

entrar em http://localhost:3000/users. Se você acessar essa URL, poderá ver que a nossa requisição GET **está funcionando**!

Não foi fornecido texto alternativo para esta imagem

Bacana, né? Agora, como fazemos pra acessar esses dados no nosso App React Native? É muito simples, utilizamos a função **fetch**. Para fazer essa conexão, ao invés de usar 'localhost:3000', você terá que **colocar diretamente o ip do seu computador**. Se usar 'localhost', o App vai entender que você tá tentando acessar o localhost do seu celular/emulador. E não é isso que queremos. Siga o exemplo abaixo:

```
teste(){
    fetch('http://seuip:3000/users')
      .then(response => response.json())
      .then(users => console.warn(users))
  }
```

Os dados estarão armazenados na variável users. Eu coloquei essa função em um 'onPress' de um botão meu. E como vocês podem ver, já está puxando os dados pro App React Native! Agora podemos manipular esses dados a vontade! Dá até uma emoção...

Não foi fornecido texto alternativo para esta imagem

Então é isso. Esse foi meu primeiro artigo aqui no linkedin. Caso tenha alguma dúvida ou sugestão de artigo (fazer um CRUD completo, por exemplo), manda um inbox pra mim que eu irei ajudar! Espero que tenha ajudado.

REFERÊNCIAS:

- https://github.com/mysqljs/mysql/issues/1213 <Acesso em 24/01/19>
- https://medium.com/@febatista107/como-converter-os-dados-de-uma-requisição-com-o-body-parser-2b5b93100f00 <Acesso em 24/01/19>

Adicione um comentário...

10 m ···

fabricio correa • 3º+
Tec. informática na Detran-PA

Amigo, desculpe a ignorância, mas essa função que tem o fatch fica dentro da página Routes ou em outra página...como ficaria? Desde já, obrigado.

· 1 gostou | · 3 respostas

Carregar respostas anteriores

fabricio evangelista correa • 3º+ 9 m ···
Técnico em sistemas de informática na Detran-PA

eu consegui fazer . Obrigado. Mas estou precisando fazer muito um CRUD, ajude-me por favor.

· 1 gostou |

Saulo Joab Penha Simplicio • 3º+ 9 m ···
Estagiário na Enel Brasil

O CRUD é um pouquinho mais complicado, se quiser eu posso te recomendar um material pra estudo. Tô um pouco sem tempo pra escrever outro artigo, mas assim que puder eu escrevo.

Bernardo Junior • 3º+ 7 m ···
Desenvolvedor Web (PHP, HTML, CSS, Javascript, React Native, Node.js).

Olá Saulo, estou com a dificuldade de fazer o CRUD, teria como ajudar?

· 1 gostou | · 3 respostas

Carregar respostas anteriores

Bernardo Junior • 3º+ 7 m ···
Desenvolvedor Web (PHP, HTML, CSS, Javascript, React Native, Node.js).

essa parte de criar servidor eu já aprendi, to mesmo focado eh como inserir dados, por exemplo, tenho um banco node e uma tabela teste, lá no connection.query uso tava tentando usar template literals para tentar colocar variaveis no insert ao inves de valores manuais e não tava inserindo, sendo que se colocar valores sem variaveis, tipo "INSERT INTO teste VALUES(4,3)" funciona mas se eu usar `INSERT INTO teste VALUES(${id}, ${fkId})` não funciona, não retorna nada.

Bernardo Junior • 3º+ 7 m ···
Desenvolvedor Web (PHP, HTML, CSS, Javascript, React Native, Node.js).

tava usando mongodb com mongoose, só que é um desafio muito grande no momento, por n motivos, por isso agora to tentando mysql. Usando mongodb que aprendi a criar API REST, servidor Node etc.

Carregar mais comentários

Saulo Joab Penha Simplicio
Estagiário na Enel Brasil

+ Seguir

Mais de Saulo Joab Penha Simplicio

Hackathons: Minhas primeiras experiências.
Saulo Joab Penha Simplicio no Lin...

Mensagens


Mensagens