

CRUD API no Next.Js

Por João Siles

Nesse documento vamos aprender a criar uma API em NEXT.js! Portanto, antes de criar um novo projeto verifique se o Node.js está instalado ou não utilizando o comando:

```
Windows PowerShell
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS C:\Users\joaos> node -v
v20.10.0
PS C:\Users\joaos> |
```

Lembre de sempre manter a versão do node mais atualizada o possível! Tendo isso em mente vamos criar o nosso projeto com as seguintes opções usando o comando para isso:

```
npm install
PS C:\Users\joaos\Desktop> npx create-next-app@latest
Need to install the following packages:
create-next-app@15.3.1
Ok to proceed? (y) y
✓ What is your project named? ... aula-api
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like your code inside a `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to use Turbopack for `next dev`? ... No / Yes
✓ Would you like to customize the import alias (`@/*` by default)? ... No / Yes
Creating a new Next.js app in C:\Users\joaos\Desktop\aula-api.

Using npm.

Initializing project with template: app-tw

Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- @tailwindcss/postcss
- tailwindcss
```

Após criar abra com o VS Code usando os seguintes comandos:

```
PS C:\Users\joaos\Desktop> cd .\aula-api\
PS C:\Users\joaos\Desktop\aula-api> code .
```

Instale também o driver mysql2 com o comando:

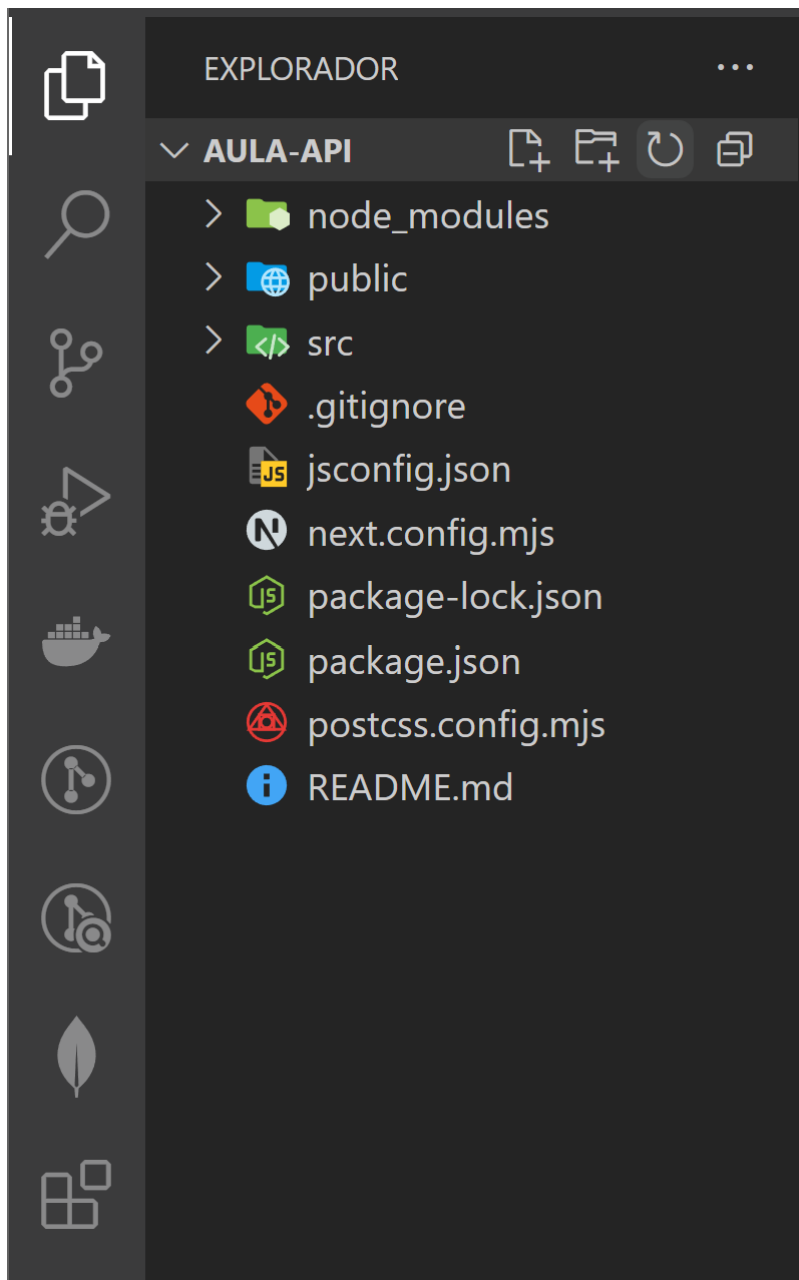
```
PS C:\Users\joaos\Desktop\aula-api> npm install mysql2

added 13 packages, and audited 56 packages in 2s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\joaos\Desktop\aula-api> npm run dev
```

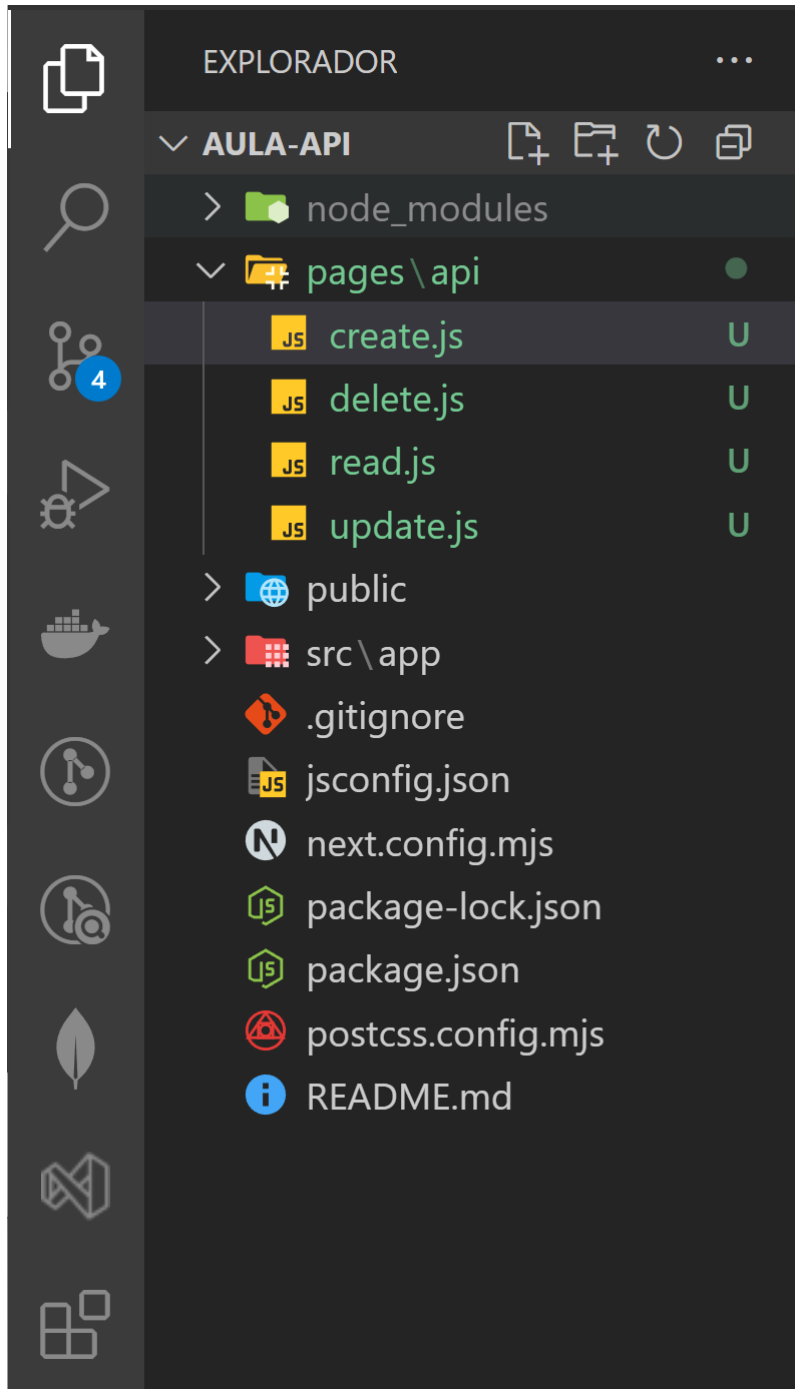
Com isso você verá a seguinte estrutura de pastas:



Agora, estamos prontos para criar uma API. Para criar uma API, primeiro crie uma pasta cujo nome é **pages** e, em seguida, crie mais uma pasta cujo nome deve ser **api**.

Dentro do diretório api, crie quatro arquivos:

- create.js
- read.js
- update.js
- delete.js



Vamos iniciar o XAMPP para o uso de banco de dados no projeto e criar um banco de dados chamado **teste-api**.

Bancos de dados

Criar banco de dados

Criar

☐ Marcar todos

Eliminar

Procurar

	Banco de dados	Colação	Ação
<input type="checkbox"/>	information_schema	utf8_general_ci	<div>Verificar privilégios</div>
<input type="checkbox"/>	mysql	utf8mb4_general_ci	<div>Verificar privilégios</div>
<input type="checkbox"/>	performance_schema	utf8_general_ci	<div>Verificar privilégios</div>
<input type="checkbox"/>	phpmyadmin	utf8_bin	<div>Verificar privilégios</div>
<input type="checkbox"/>	test	latin1_swedish_ci	<div>Verificar privilégios</div>

Total: 5

Vamos criar a seguinte tabela:

phpMyAdmin

Recente Favoritos

Novo

information_schema

mysql

performance_schema

phpmyadmin

test

teste-api

Nova

users

Servidor: 127.0.0.1 » Banco de dados: teste-api » Tabela: users

Visualizar

Estrutura

SQL

Procurar

Inserir

Exportar

Importar

Mais

Estrutura da tabela

Visão de relação(ões)

#	Nome	Tipo	Colação	Atributos	Nulo	Padrão	Comentários	Extra	Ação
<input type="checkbox"/>	1 id	int(11)			Não	Nenhum		AUTO_INCREMENT	<div>Alterar</div>
<input type="checkbox"/>	2 name	varchar(60)	utf8mb4_general_ci		Não	Nenhum			<div>Alterar</div>
<input type="checkbox"/>	3 email	varchar(80)	utf8mb4_general_ci		Não	Nenhum			<div>Alterar</div>

☐ Marcar todos

Com marcados:

Visualizar

Alterar

Eliminar

Primária

Único

Índice

Espacial

Texto completo

Adicionar coluna(s) central(is)

Remover da(s) coluna(s) central(is)

Imprimir

Propor estrutura de tabela

Monitorar tabela

Mover campo(s)

Normalizar

Adicionar

1

campo(s)

após email

Executar

Vamos agora editar o arquivo **read.js** com o seguinte código:



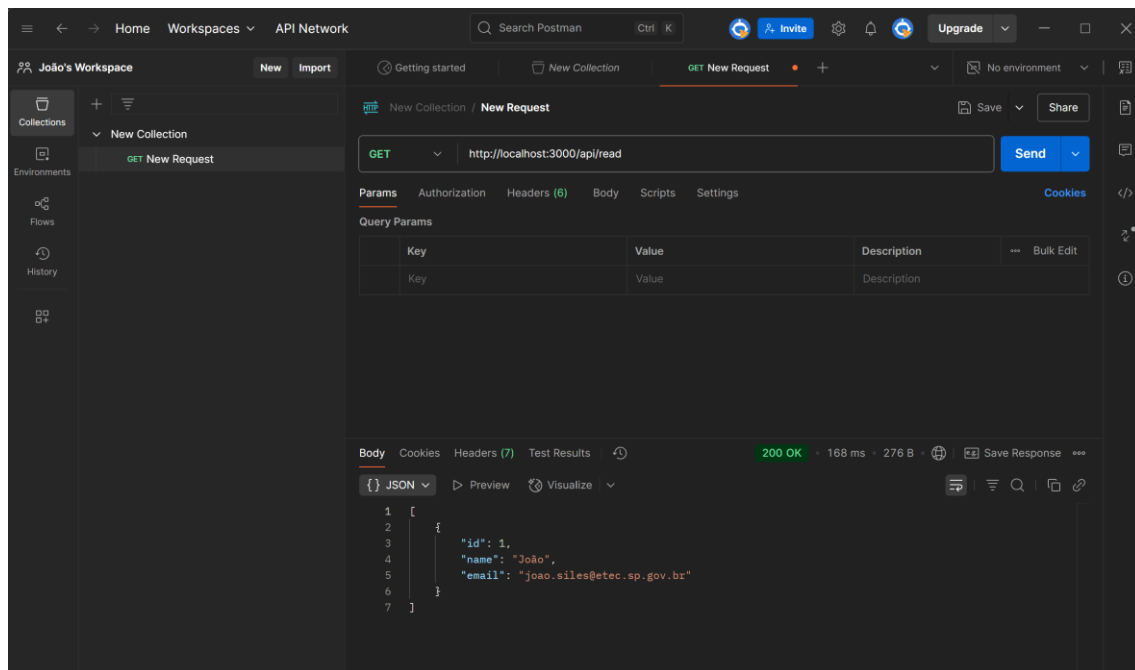
```

1 // pages/api/read.js
2 import { createConnection } from 'mysql2/promise';
3
4 // Função para conectar no MySQL
5 async function connectToDatabase() {
6   return createConnection({
7     host: 'localhost',
8     user: 'root',
9     password: '',
10    database: 'teste-api',
11  });
12 }
13
14 export default async function handler(req, res) {
15   if (req.method !== 'GET') {
16     return res.status(405).json({ error: 'Metodo não permitido' });
17   }
18
19   try {
20     // Conexão no MySQL
21     const connection = await connectToDatabase();
22
23     // Execução da query para receber dados da tabela "User"
24     const [rows] = await connection.execute('SELECT * FROM users WHERE id = 1, ');
25
26     // Verificar se o usuário existe
27     if (rows.length === 0) {
28       return res.status(404).json({ error: 'Usuário não encontrado.' });
29     }
30
31     // Fechar a conexão
32     await connection.end();
33
34     // Resposta com os dados do usuário
35     res.status(200).json(rows);
36   } catch (error) {
37     console.error('Erro de conexão com o banco:', error);
38     res.status(500).json({ error: 'Erro Interno de Servidor' });
39   }
40 }

```

Para testar o código vamos usar o software **Postman**

(<https://www.postman.com/downloads/>) e com o servidor do Next.js (**npm run dev**) iniciado e alguns dados cadastrados podemos executar nele da seguinte maneira:



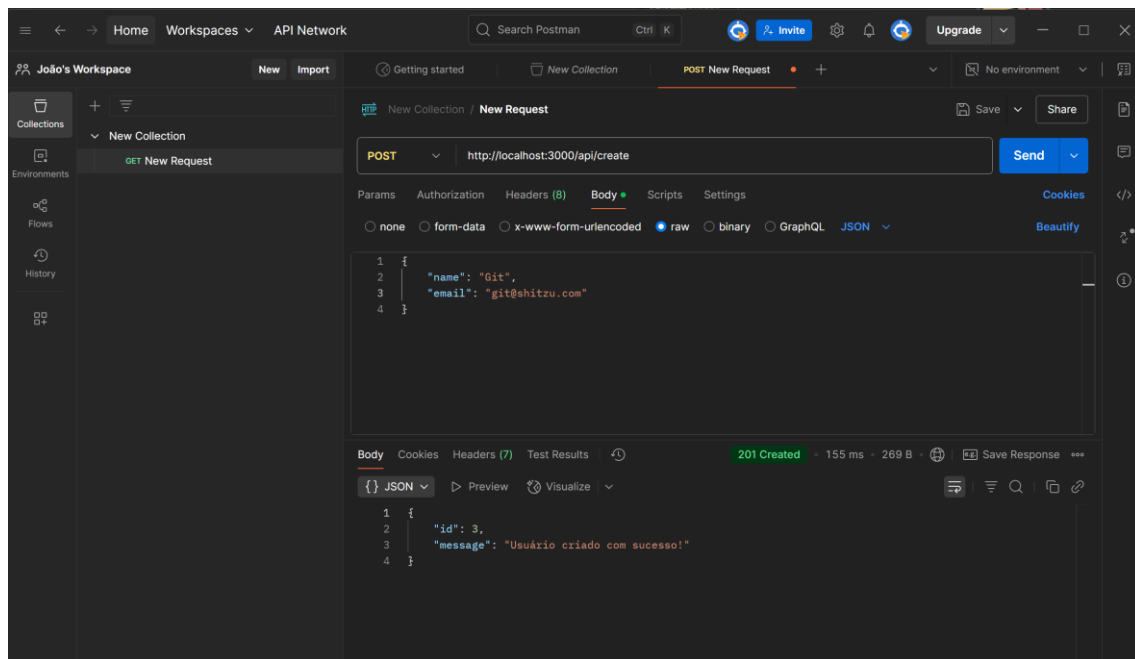
O próximo será o **create.js**:

```

1 // pages/api/create.js
2 import { createConnection } from 'mysql2/promise';
3
4 // Função para conectar no MySQL
5 async function connectToDatabase() {
6   return createConnection({
7     host: 'localhost',
8     user: 'root',
9     password: '',
10    database: 'teste-api',
11  });
12 }
13
14 export default async function handler(req, res) {
15   if (req.method !== 'POST') {
16     return res.status(405).json({ error: 'Metodo não permitido' });
17   }
18
19   const userdata = req.body;
20   console.log(userdata)
21
22   const { name, email } = userdata;
23
24   if (!name || !email) {
25     return res.status(400).json({ error: 'name e email são obrigatórios no request body.' });
26   }
27   try {
28     // Conecta no banco
29     const connection = await connectToDatabase();
30
31     // Executa a query para transacionar dados da tabela "user"
32     const [result] = await connection.execute('INSERT INTO users (name, email) VALUES (?, ?)', [
33       name,
34       email,
35     ]);
36     // Checa se o usuário existe
37
38     // Encerra conexão
39     await connection.end();
40
41     // Respond with the user data
42     res.status(201).json({ id: result.insertId, message: 'Usuário criado com sucesso!' });
43   } catch (error) {
44     console.error('Error de conexão com o banco:', error);
45     res.status(500).json({ error: 'Erro Interno de Servidor' });
46   }
47 }

```

No Postman o teste fica assim:



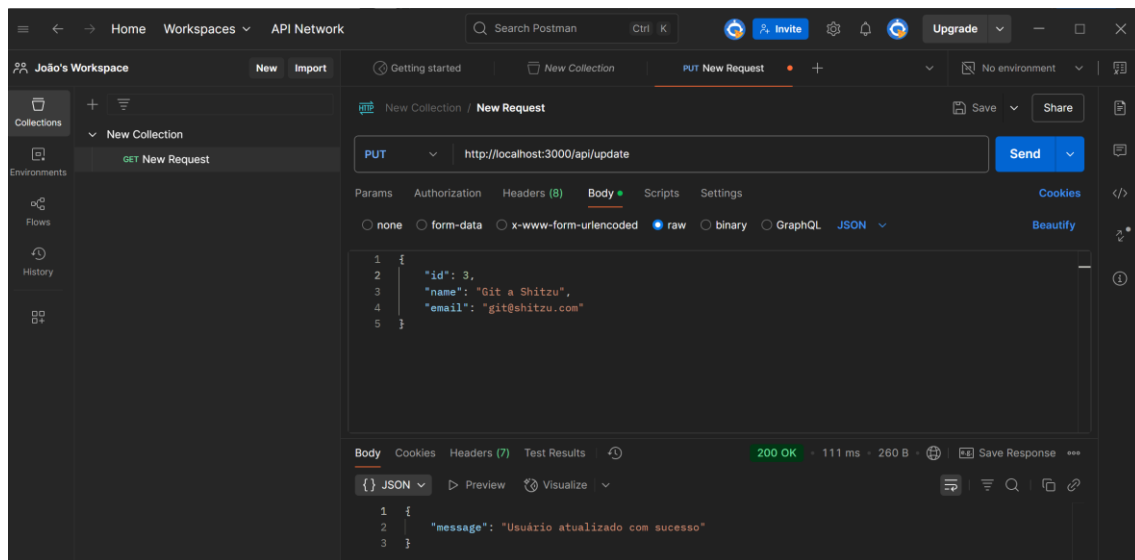
Hora do **update.js**:


```

1  import { createConnection } from 'mysql2/promise';
2
3  // Função para conectar no MySQL
4  async function connectToDatabase() {
5      return createConnection({
6          host: 'localhost',
7          user: 'root',
8          password: '',
9          database: 'teste-api',
10     });
11 }
12
13 // Rota da API Update de user
14 export default async function handler(req, res) {
15     if (req.method !== 'PUT') {
16         return res.status(405).json({ error: 'Metodo não permitido' });
17     }
18
19     const { id, name, email } = req.body;
20     console.log(req.body);
21
22     if (!id || !name || !email) {
23         return res.status(400).json({ error: 'id, name, and email são obrigatorios no request body.' });
24     }
25
26
27     try {
28         // Connect to the database
29         const connection = await connectToDatabase();
30
31         // Executa query para atualizar a tabela "users"
32         const [result] = await connection.execute(
33             'UPDATE users SET name = ?, email = ? WHERE id = ?',
34             [name, email, id]
35         );
36
37         // Fecha conexão com o banco
38         await connection.end();
39
40         // Check se o update teve sucesso
41         if (result.affectedRows === 0) {
42             return res.status(404).json({ error: 'Usuário não encontrado.' });
43         }
44
45         // Resposta do server
46         res.status(200).json({ message: 'Usuário atualizado com sucesso' });
47     } catch (error) {
48         console.error('Error de conexão com o banco:', error);
49         res.status(500).json({ error: 'Erro interno de servidor' });
50     }
51 }

```

Para testar:



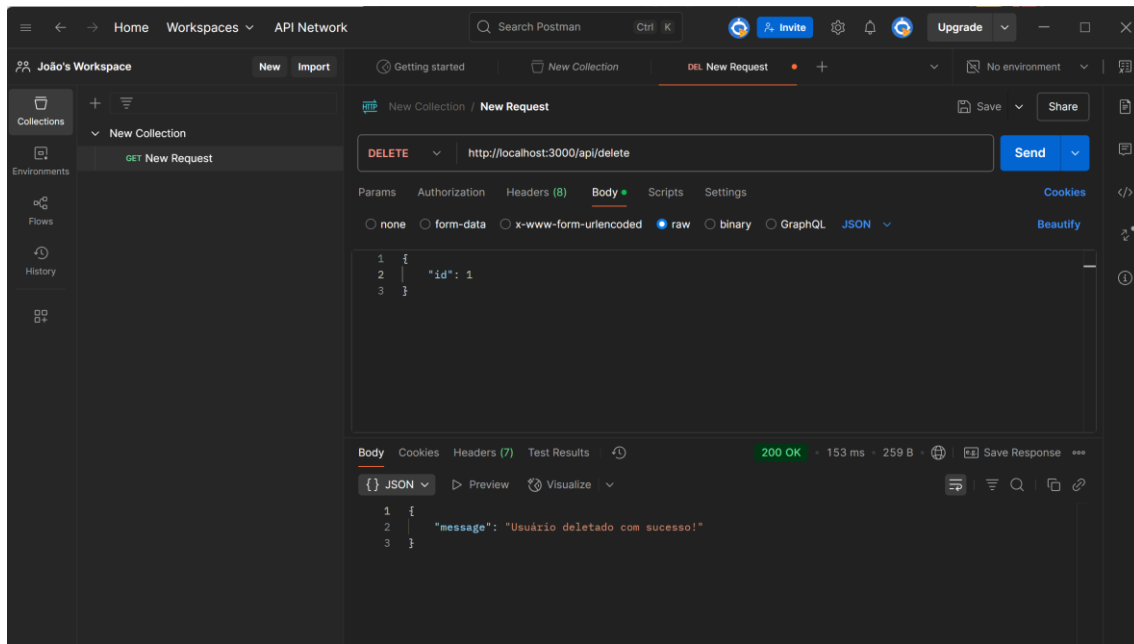
E por último o **delete.js**:

```

1 //api/delete.js
2
3 import { createConnection } from 'mysql2/promise';
4
5 // Função para conectar no MySQL
6 async function connectToDatabase() {
7   return createConnection({
8     host: 'localhost',
9     user: 'root',
10    password: '',
11    database: 'teste-api',
12  });
13 }
14
15 // Rota da API de Delete do usuário
16 export default async function handler(req, res) {
17   if (req.method !== 'DELETE') {
18     return res.status(405).json({ error: 'Método não permitido' });
19   }
20
21   const { id } = req.body;
22   console.log(req.body)
23   if (!id) {
24     return res.status(400).json({ error: 'O id é obrigatório no request body.' });
25   }
26
27   try {
28     // Conecta no banco
29     const connection = await connectToDatabase();
30
31     // Executa o delete em "users"
32     const [result] = await connection.execute('DELETE FROM users WHERE id = ?', [id]);
33
34     // Fecha conexão
35     await connection.end();
36
37     // Check se foi deletado com sucesso
38     if (result.affectedRows === 0) {
39       return res.status(404).json({ error: 'Usuário não encontrado.' });
40     }
41
42     // Respostas de sucesso
43     res.status(200).json({ message: 'Usuário deletado com sucesso!' });
44   } catch (error) {
45     console.error('Error de conexão com o banco:', error);
46     res.status(500).json({ error: 'Erro interno de servidor' });
47   }
48 }

```

No Postman:



Com isso concluímos a criação de um CRUD básico no Next.js! Espero que tenham gostado!