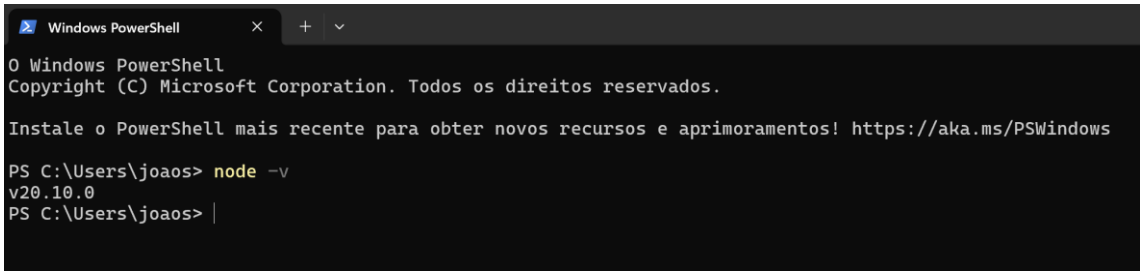
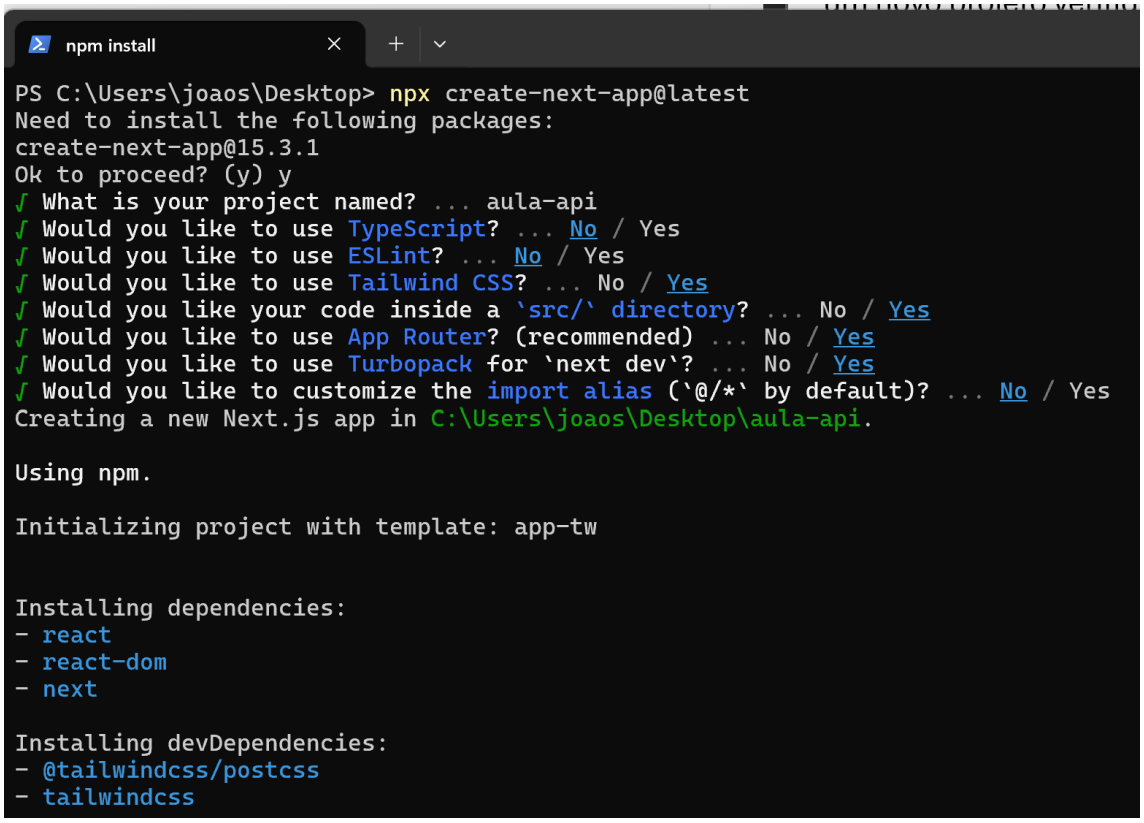
# CRUD API no Next.Js

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:




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:



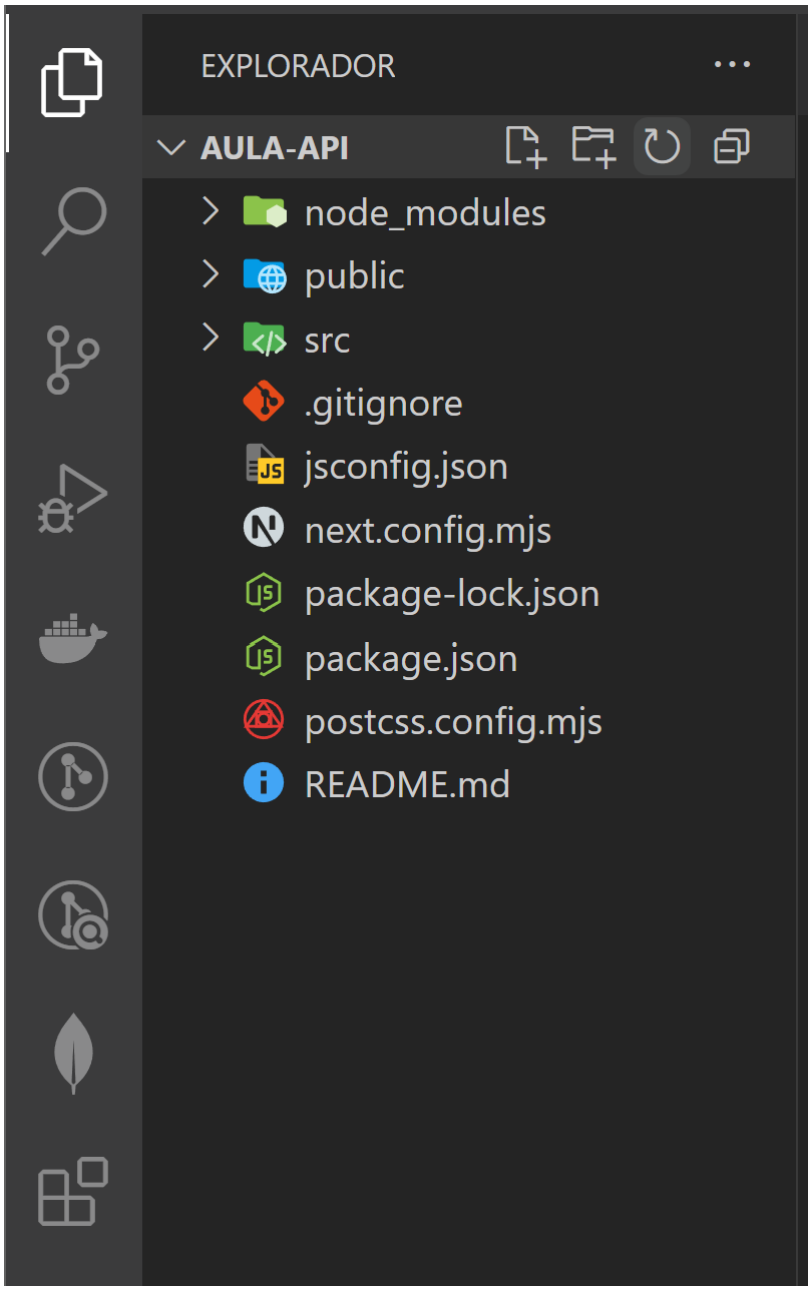

Após criar abra com o VS Code usando os seguintes comandos:

```
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
```

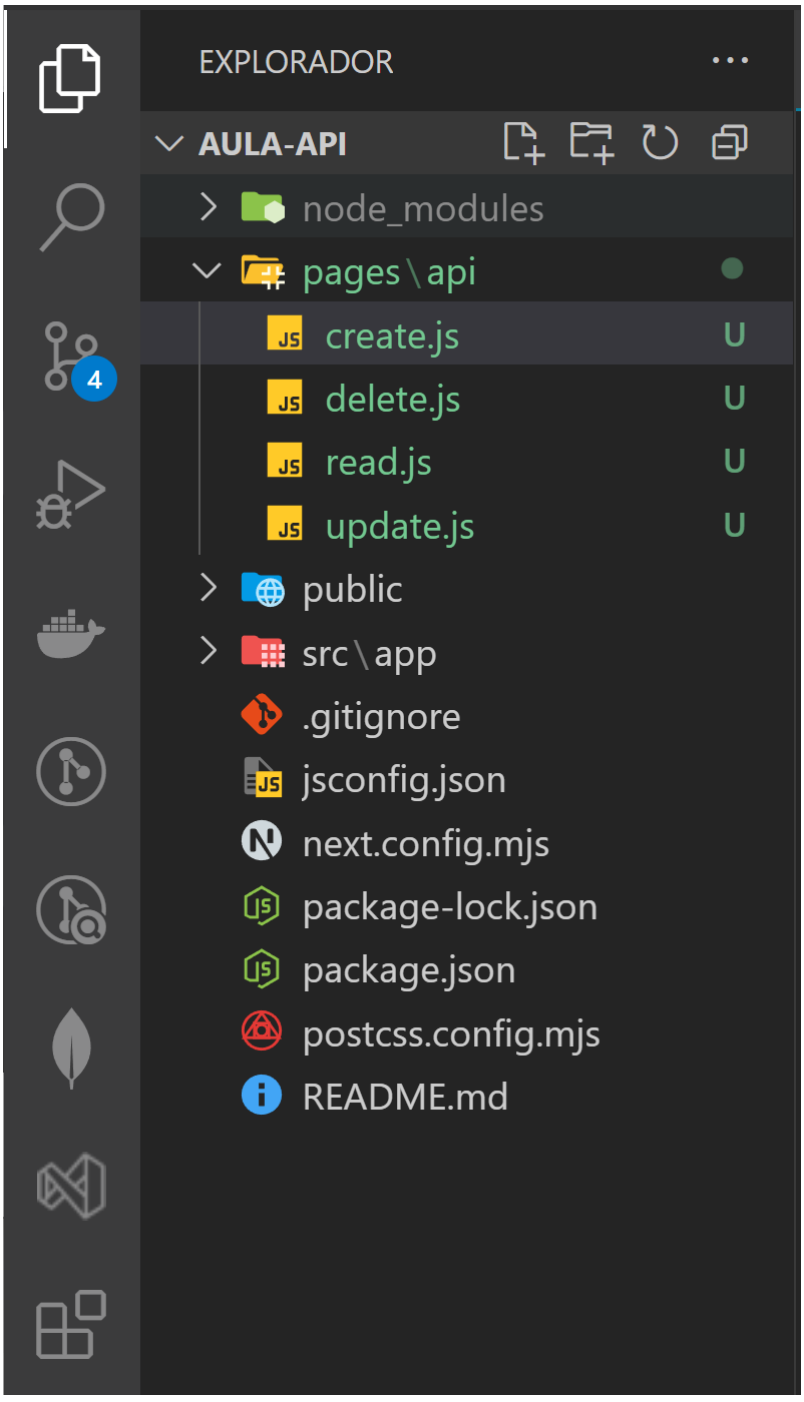
Com isso você verá a seguinte estrutura de pastas:




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**.

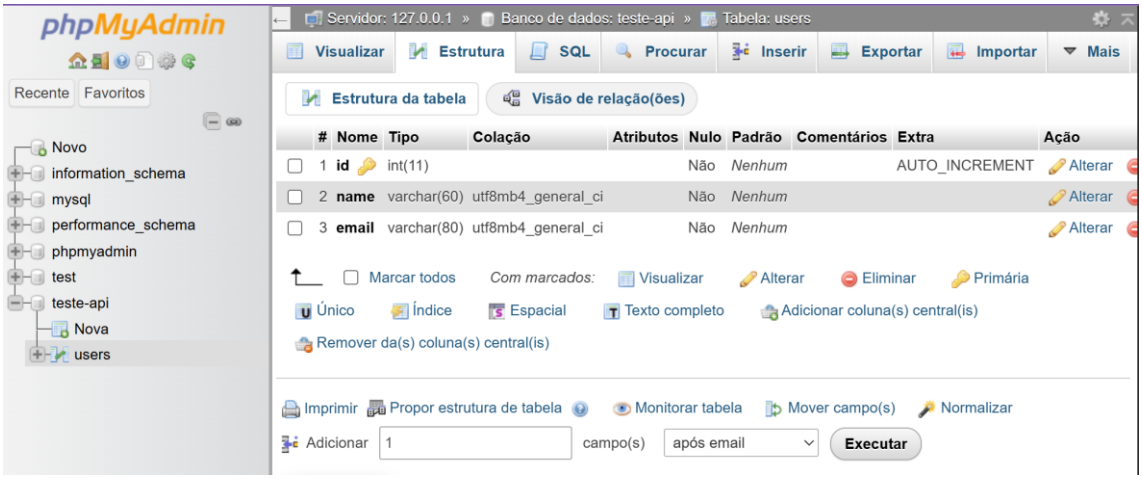
Dentro do diretório api, crie quatro arquivos:

- 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**.

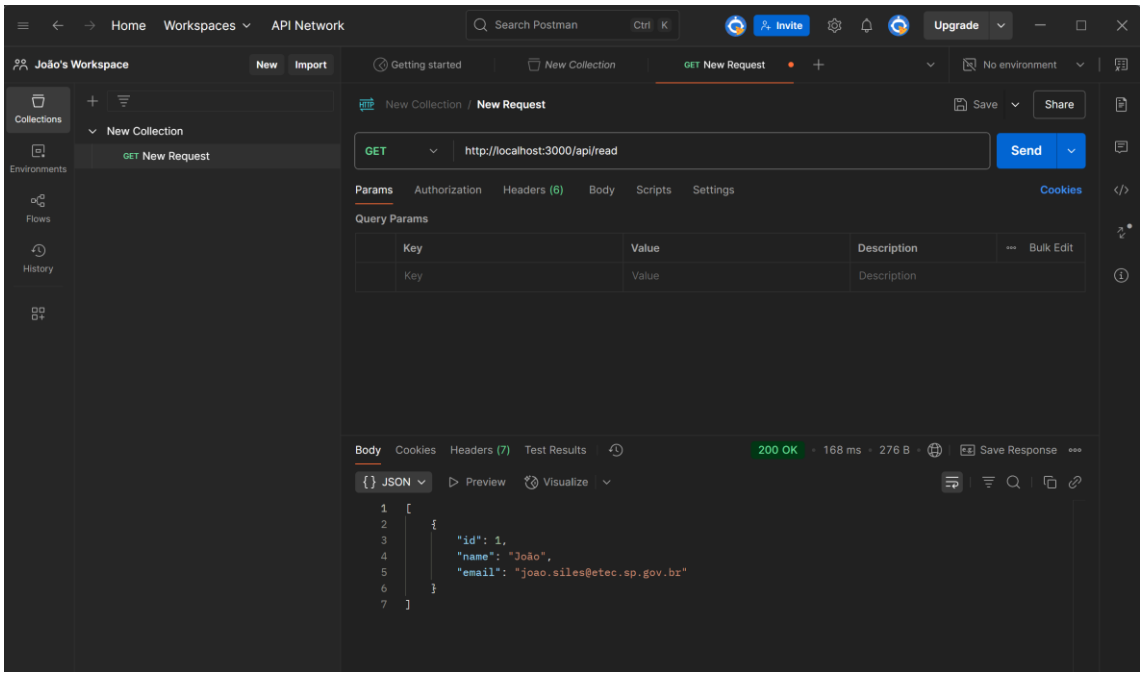

Vamos criar a seguinte tabela:




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

```
// pages/api/read.js
import { createConnection } from 'mysql2/promise';

// Função para conectar no MySQL
async function connectToDatabase() {
  return createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'teste-api',
  });
}

export default async function handler(req, res) {
  if (req.method !== 'GET') {
    return res.status(405).json({ error: 'Metodo não permitido' });
  }

  try {
    // Conexão no MySQL
    const connection = await connectToDatabase();

    // Execução da query para receber dados da tabela "User"
    const [rows] = await connection.execute('SELECT * FROM users WHERE id = 1', );

    // Verificar se o usuário existe
    if (rows.length === 0) {
      return res.status(404).json({ error: 'Usuário não encontrado.' });
    }

    // Fechar a conexão
    await connection.end();

    // Resposta com os dados do usuário
    res.status(200).json(rows);
  } catch (error) {
    console.error('Erro de conexão com o banco:', error);
    res.status(500).json({ error: 'Erro Interno de Servidor' });
  }
}
```

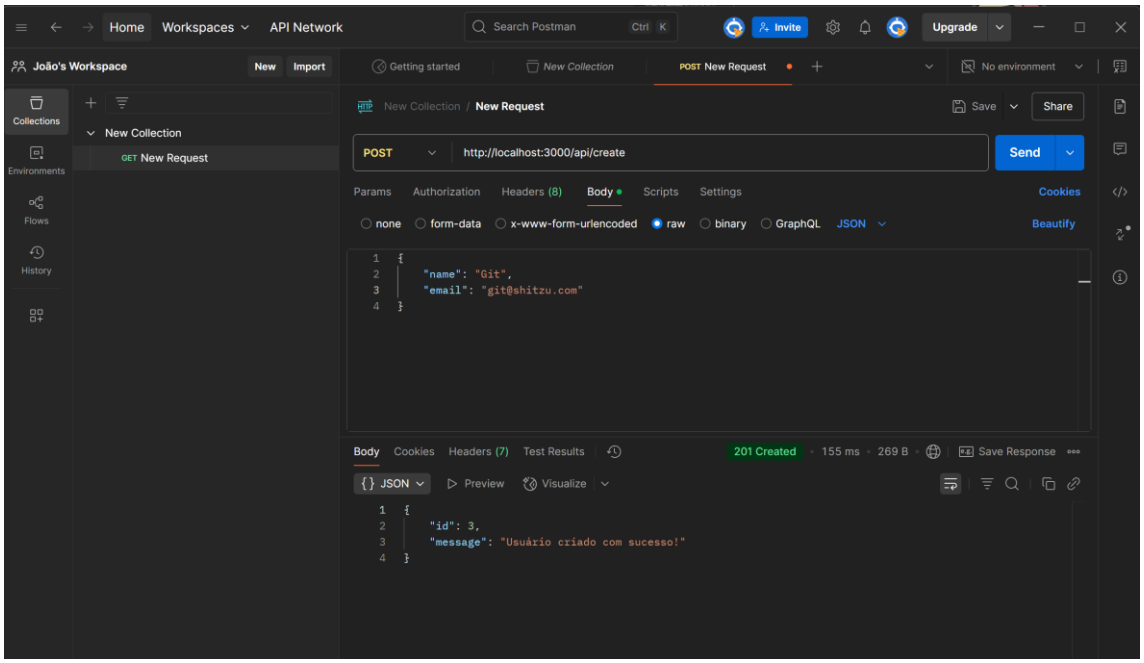
Para testar o código vamos usar o software **Postman** (**https://www.postman.com/downloads/**) e com o servidor do Next.js (**npm run dev**) iniciado e alguns dados cadastrados podemos executar nele da seguinte maneira:




O próximo será o **create.js**:

```
// pages/api/create.js
import { createConnection } from 'mysql2/promise';

// Função para conectar no MySQL
async function connectToDatabase() {
  return createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'teste-api',
  });
}

export default async function handler(req, res) {
  if (req.method !== 'POST') {
    return res.status(405).json({ error: 'Metodo não permitido' });
  }

  const userdata = req.body;
  console.log(userdata)

  const { name, email} = userdata;

  if (!name || !email ) {
    return res.status(400).json({ error: 'name e email são obrigatórios no request body.' });
  }
  try {
    // Conecta no banco
    const connection = await connectToDatabase();

    // Executa a query para transacionar dados da tabela "user"
    const [result] = await connection.execute('INSERT INTO users (name, email) VALUES (?, ?)', [
      name,
      email,
    ]);
    // Checa se o usuário existe

    // Encerra conexão
    await connection.end();

    // Respond with the user data
    res.status(201).json({ id: result.insertId, message: 'Usuário criado com sucesso!' });
  } catch (error) {
    console.error('Error de conexão com o banco:', error);
    res.status(500).json({ error: 'Erro Interno de Servidor' });
  }
}
```

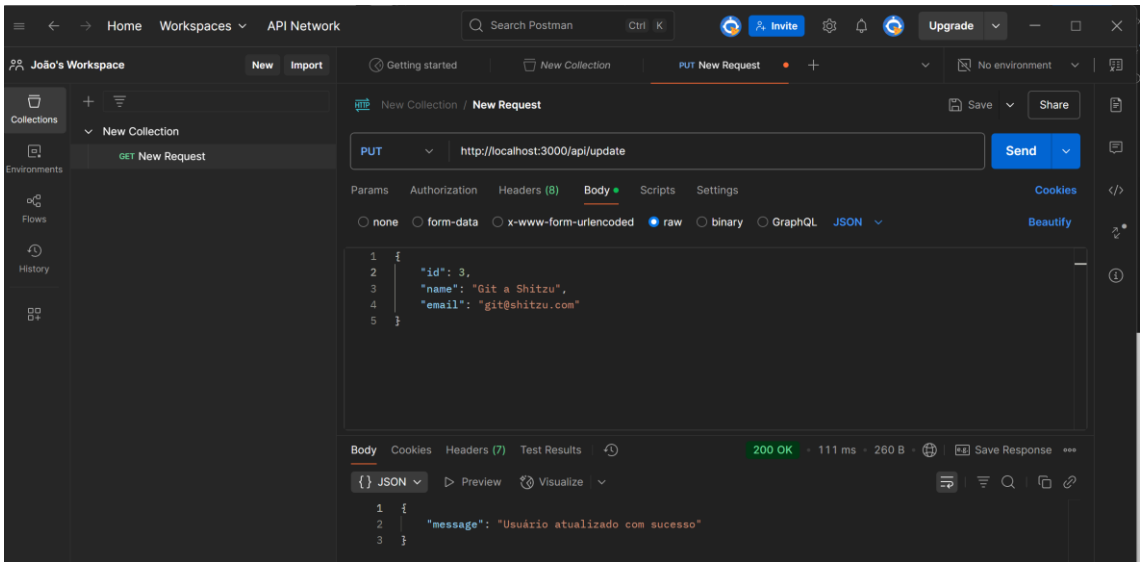
No Postman o teste fica assim:




Hora do **update.js**:

```js
import { createConnection } from 'mysql2/promise';

// Função para conectar no MySQL
async function connectToDatabase() {
  return createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'teste-api',
  });
}

// Rota da API Update de user
export default async function handler(req, res) {
  if (req.method !== 'PUT') {
    return res.status(405).json({ error: 'Metodo não permitido' });
  }

  const { id, name, email } = req.body;
  console.log(req.body);

  if (!id || !name || !email) {
    return res.status(400).json({ error: 'id, name, and email são obrigatorios no request body.' });
  }


  try {
    // Connect to the database
    const connection = await connectToDatabase();

    // Executa query para atualizar a tabela "users"
    const [result] = await connection.execute(
      'UPDATE users SET name = ?, email = ? WHERE id = ?',
      [name, email, id]
    );

    // Fecha conexão com o banco
    await connection.end();

    // Check se o update teve sucesso
    if (result.affectedRows === 0) {
      return res.status(404).json({ error: 'Usuário não encontrado.' });
    }

    // Resposta do server
    res.status(200).json({ message: 'Usuário atualizado com sucesso' });
  } catch (error) {
    console.error('Error de conexão com o banco:', error);
    res.status(500).json({ error: 'Erro interno de servidor' });
  }
}
```

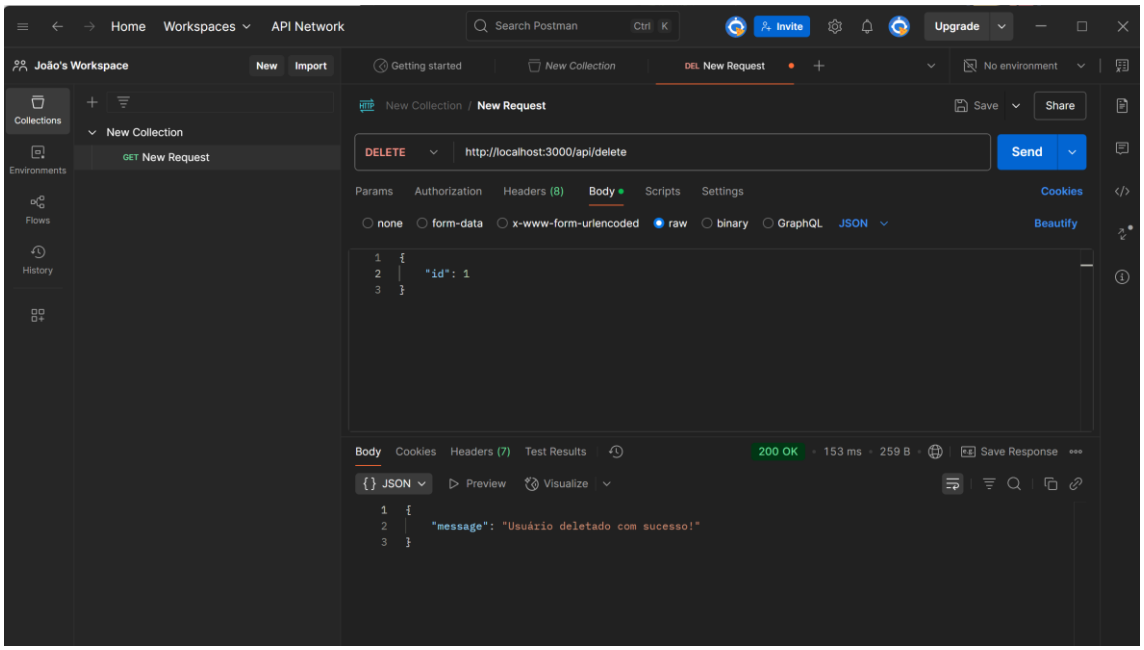
Para testar:




E por último o **delete.js:**

```
//api/delete.js

import { createConnection } from 'mysql2/promise';

// Função para conectar no MySQL
async function connectToDatabase() {
  return createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'teste-api',
  });
}

// Rota da API de Delete do usuário
export default async function handler(req, res) {
  if (req.method !== 'DELETE') {
    return res.status(405).json({ error: 'Método não permitido' });
  }

  const { id } = req.body;
console.log(req.body)
  if (!id) {
    return res.status(400).json({ error: 'O id é obrigatório no request body.' });
  }

  try {
    // Conecta no banco
    const connection = await connectToDatabase();

    // Executa o delete em "users"
    const [result] = await connection.execute('DELETE FROM users WHERE id = ?', [id]);

    // Fecha conexão
    await connection.end();

    // Check se foi deletado com sucesso
    if (result.affectedRows === 0) {
      return res.status(404).json({ error: 'Usuário não encontrado.' });
    }

    // Respostas de sucesso
    res.status(200).json({ message: 'Usuário deletado com sucesso!' });
  } catch (error) {
    console.error('Error de conexão com o banco:', error);
    res.status(500).json({ error: 'Erro interno de servidor' });
  }
}
```

No Postman:




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