

PRÁTICAS AVANÇADAS EM DESENVOLVIMENTO WEB

Davi Schneid - davi.Schneid@gmail.com

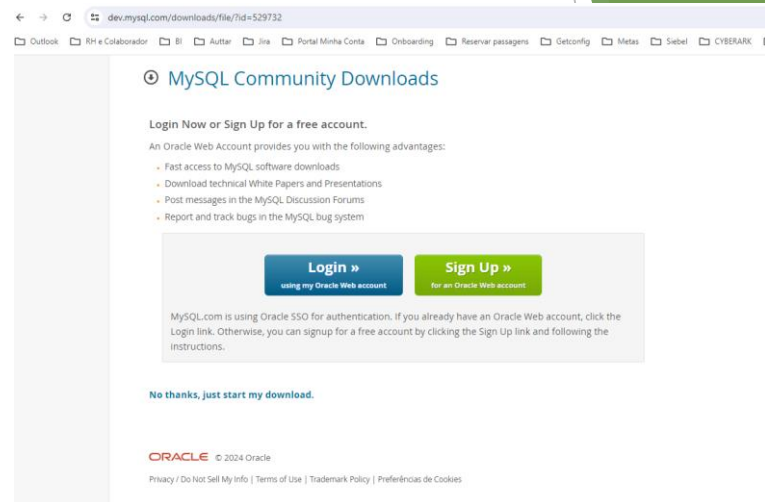
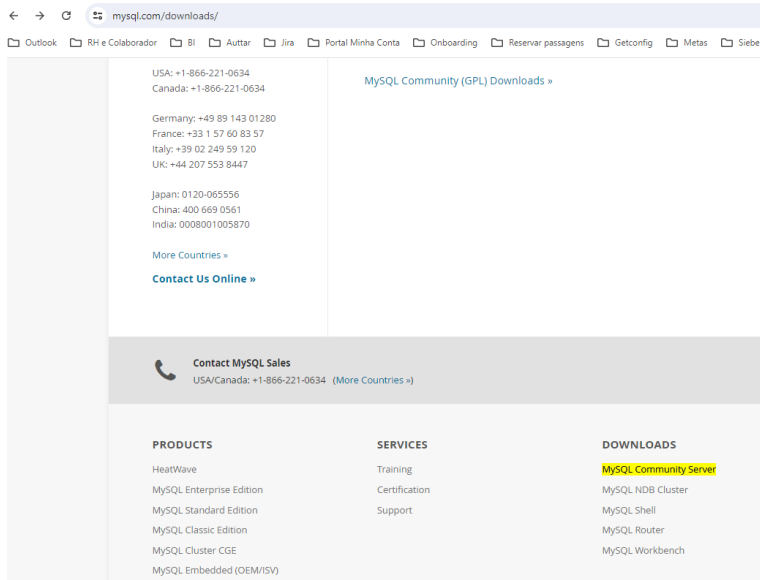
16/07/2024

Agenda

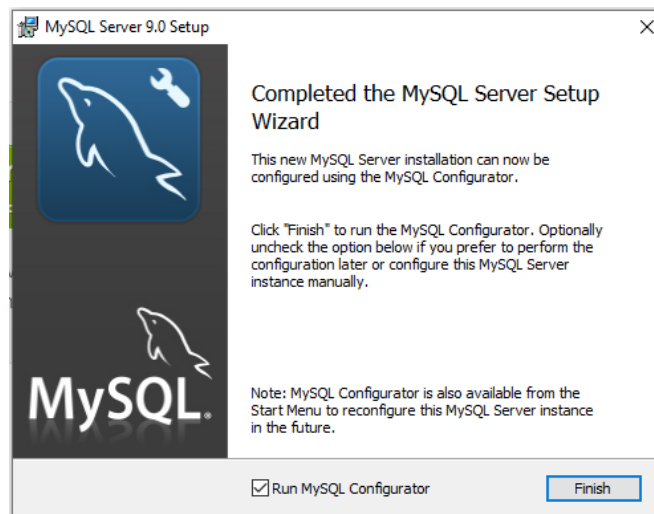
- ▶ Criar banco de dados relacional
- ▶ Criar API
- ▶ Criar funcionalidades
 - ▶ Cadastro de clientes
 - ▶ Listagem de clientes
 - ▶ Pesquisa de clientes
 - ▶ Edição de clientes
 - ▶ Exclusão de clientes
- ▶ ORM (Object-Relational Mapping)

Instalar Mysql

- Download Mysql -> <https://dev.mysql.com/downloads/mysql/>



- Instalar



MYSQL

- ▶ Criar banco de dados.
- ▶ Criar tabelas.
- ▶ Inserir registros

```
MySQL 9.0 Command Line Client
Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use API
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> CREATE TABLE IF NOT EXISTS USUARIOS(
  -> ID INT NOT NULL AUTO_INCREMENT,
  -> NOME VARCHAR(255) NOT NULL,
  -> IDADE INT NOT NULL,
  -> CIDADE VARCHAR(150) NOT NULL,
  -> PRIMARY KEY (ID)
  -> );
Query OK, 0 rows affected (0.04 sec)

mysql> show tables;
+-----+
| Tables_in_api |
+-----+
| usuarios      |
+-----+
1 row in set (0.00 sec)

mysql> INSERT INTO USUARIOS (NOME,IDADE,CIDADE) VALUES ("DAVI BORGES SCHNEID",43,"CANOAS");
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM USUARIOS;
+-----+-----+-----+-----+
| ID | NOME                | IDADE | CIDADE |
+-----+-----+-----+-----+
|  1 | DAVI BORGES SCHNEID |    43 | CANOAS |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Criar API

- ▶ `npm init -y`
- ▶ `npm install mysql2 dotenv express`
 - ▶ **mysql2**: conectar e mandar comandos SQL para o banco;
 - ▶ **dotenv**: gestão das configurações do projeto;
 - ▶ **express**: web framework para construção da infraestrutura da API;

```
Eric@DESKTOP-8CI3MGV MINGW64 /c/Users/SenacRs/PraticasAvancadasDesenvolvimentoWeb/APIMYSQL (main)
$ npm init -y
Wrote to C:\Users\SenacRs\PraticasAvancadasDesenvolvimentoWeb\APIMYSQL\package.json:

{
  "name": "apimysql",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

Eric@DESKTOP-8CI3MGV MINGW64 /c/Users/SenacRs/PraticasAvancadasDesenvolvimentoWeb/APIMYSQL (main)
$ ls -ltr
total 1
drwxr-xr-x 1 Eric 197609  0 Jul 15 15:24 Backup_BD/
-rw-r--r-- 1 Eric 197609 222 Jul 15 15:31 package.json

Eric@DESKTOP-8CI3MGV MINGW64 /c/Users/SenacRs/PraticasAvancadasDesenvolvimentoWeb/APIMYSQL (main)
$ npm install mysql2 dotenv express

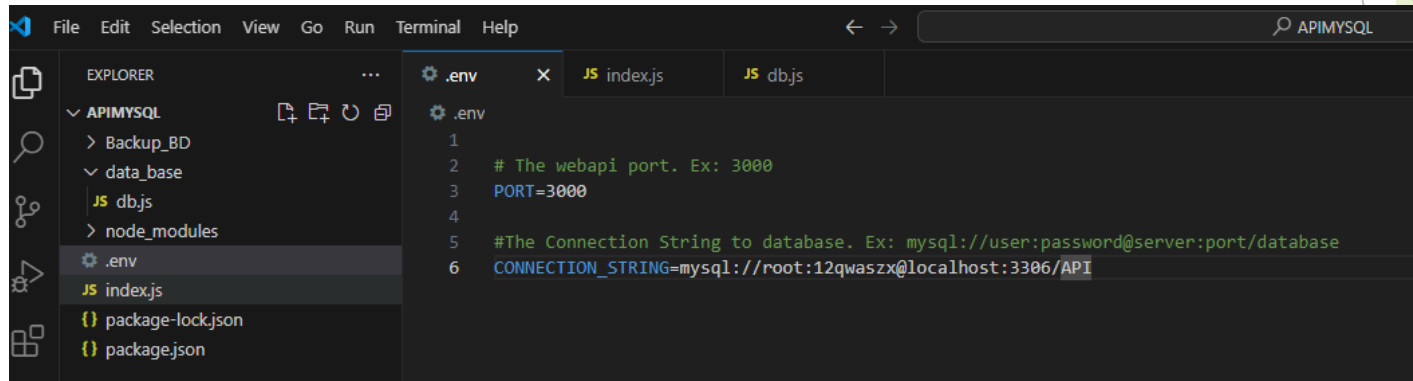
added 76 packages, and audited 77 packages in 7s

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

found 0 vulnerabilities
```

Arquivo configuração

- ▶ Criar arquivo sem nome .env na raiz do projeto
- ▶ Adicionar os parâmetros nele

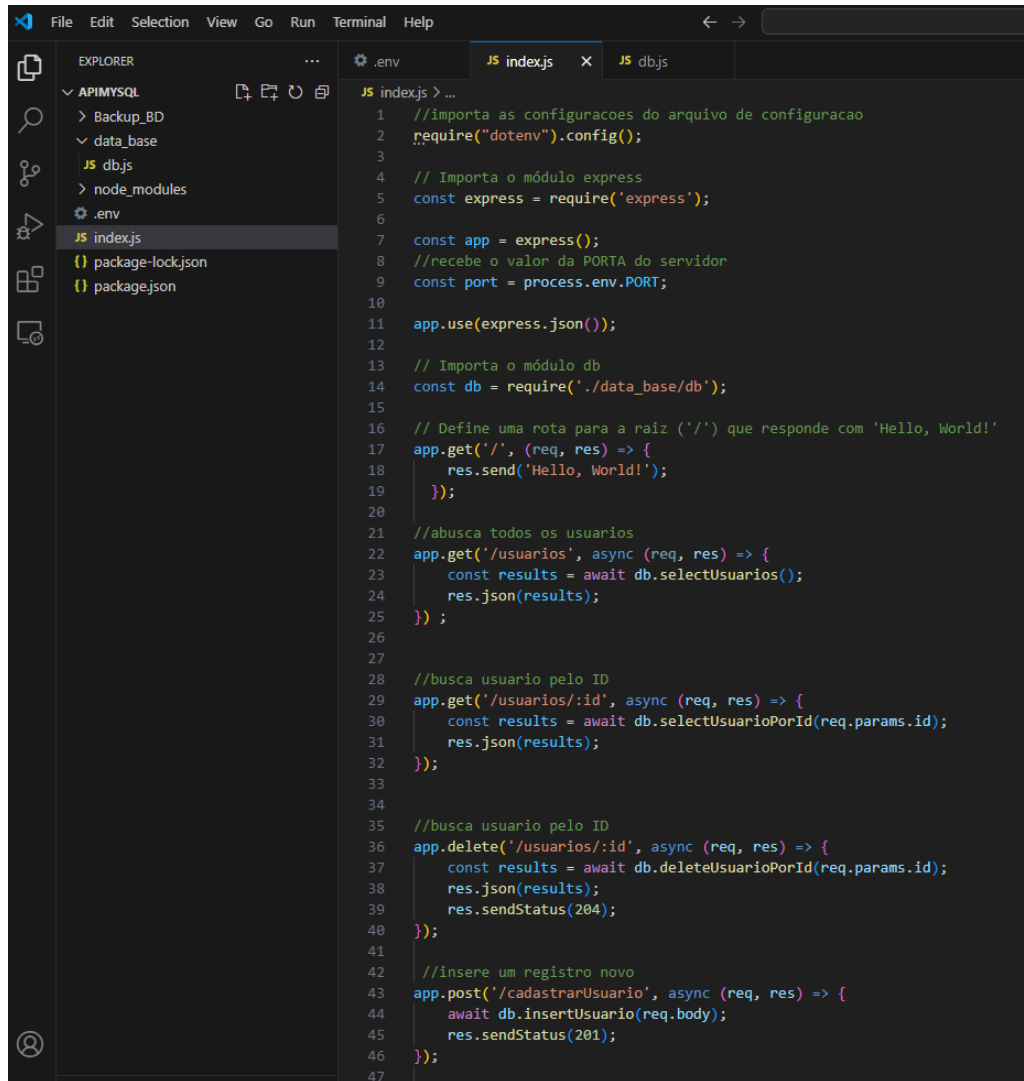


The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the project structure for 'APIMYSQL'. The files listed are Backup_BD, data_base, db.js, node_modules, .env, index.js, package-lock.json, and package.json. The main editor area shows the content of the .env file, which includes comments and two environment variables: PORT and CONNECTION_STRING.

```
.env
1
2 # The webapi port. Ex: 3000
3 PORT=3000
4
5 #The Connection String to database. Ex: mysql://user:password@server:port/database
6 CONNECTION_STRING=mysql://root:12qwaszx@localhost:3306/API
```

Criar index.js

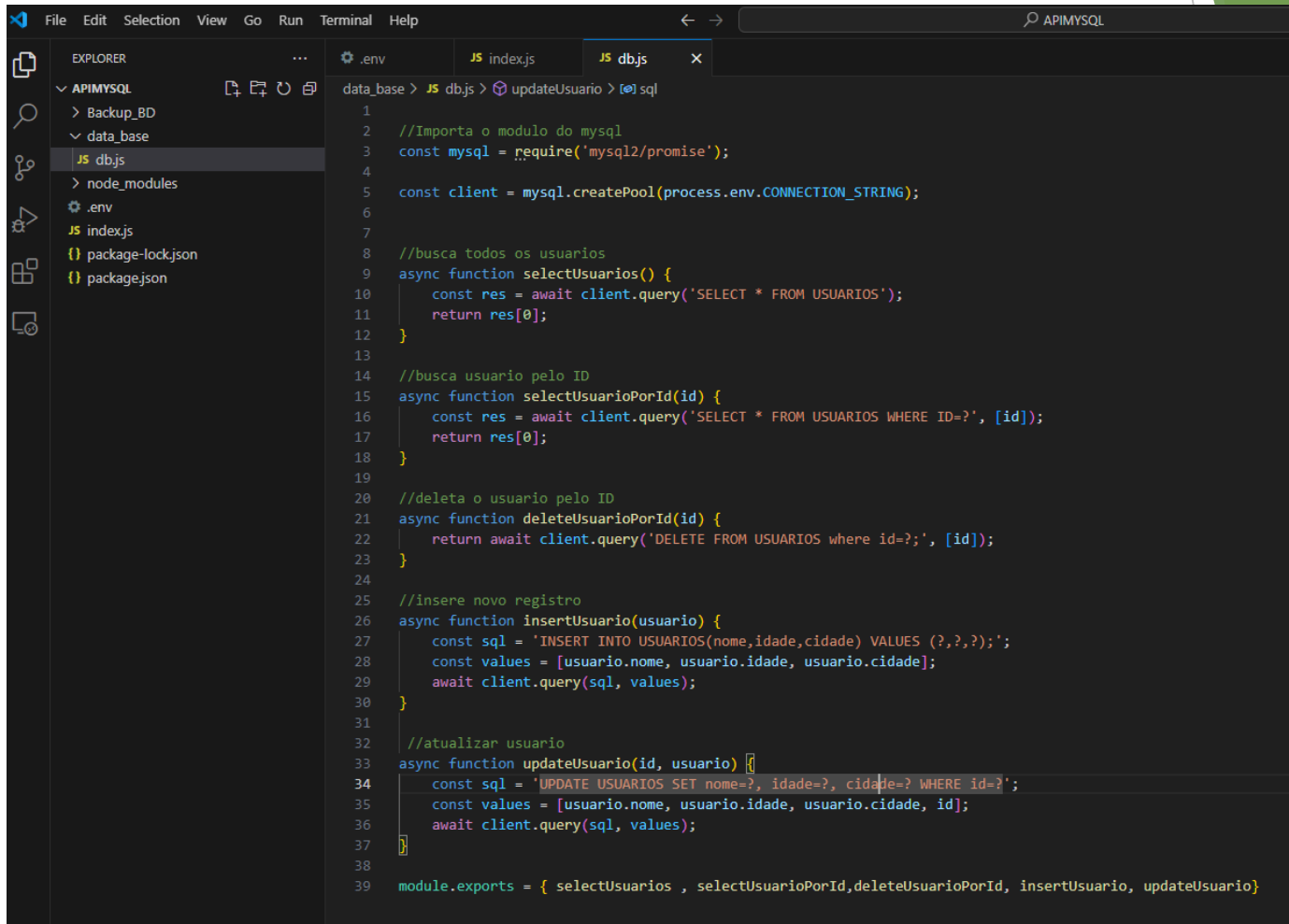
► Criar arquivo index.js

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with folders 'Backup_BD' and 'data_base', and files 'db.js', 'index.js' (selected), 'package-lock.json', and 'package.json'. The main editor area displays the content of 'index.js'. The code is written in JavaScript and uses ES6 features like 'const', 'async', and 'await'. It sets up an Express.js application, connects to a database using 'require' and 'db', and defines several REST API endpoints: a root route returning 'Hello, World!', a GET route for listing users, a GET route for retrieving a user by ID, a DELETE route for removing a user by ID, and a POST route for creating a new user. The code is numbered from 1 to 47.

```
1 //importa as configuracoes do arquivo de configuracao
2 require("dotenv").config();
3
4 // Importa o módulo express
5 const express = require('express');
6
7 const app = express();
8 //recebe o valor da PORTA do servidor
9 const port = process.env.PORT;
10
11 app.use(express.json());
12
13 // Importa o módulo db
14 const db = require('./data_base/db');
15
16 // Define uma rota para a raiz ('/') que responde com 'Hello, World!'
17 app.get('/', (req, res) => {
18   res.send('Hello, World!');
19 });
20
21 //abusa todos os usuarios
22 app.get('/usuarios', async (req, res) => {
23   const results = await db.selectUsuarios();
24   res.json(results);
25 });
26
27
28 //busca usuario pelo ID
29 app.get('/usuarios/:id', async (req, res) => {
30   const results = await db.selectUsuarioPorId(req.params.id);
31   res.json(results);
32 });
33
34
35 //busca usuario pelo ID
36 app.delete('/usuarios/:id', async (req, res) => {
37   const results = await db.deleteUsuarioPorId(req.params.id);
38   res.json(results);
39   res.sendStatus(204);
40 });
41
42 //insere um registro novo
43 app.post('/cadastrarUsuario', async (req, res) => {
44   await db.insertUsuario(req.body);
45   res.sendStatus(201);
46 });
47
```

Criar db.js

- Criar arquivo db.js

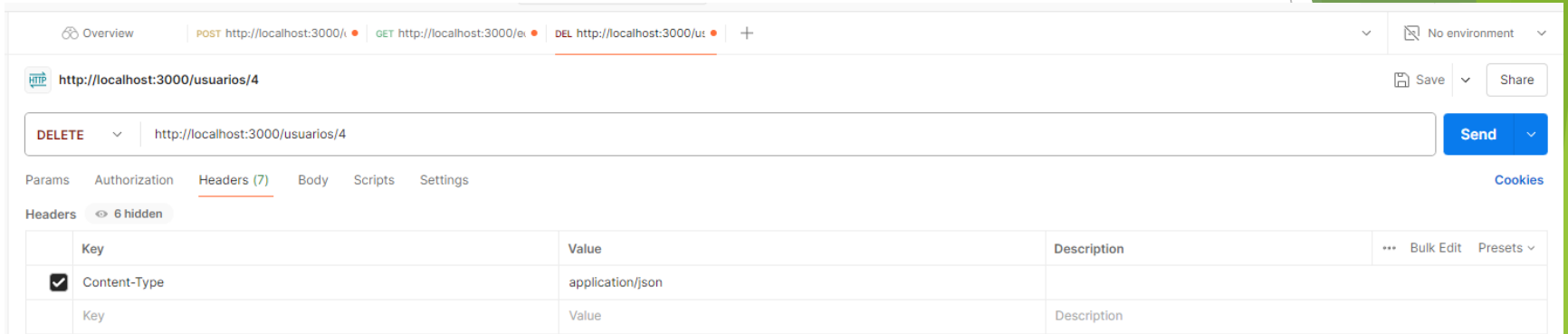


The screenshot shows a code editor with a dark theme. On the left, the Explorer panel shows a project structure with folders 'Backup_BD' and 'data_base', and files 'db.js', 'node_modules', '.env', 'index.js', 'package-lock.json', and 'package.json'. The 'db.js' file is selected. The main editor area shows the content of 'db.js', which is a JavaScript file for connecting to a MySQL database using the 'mysql2/promise' module. The code includes functions for selecting all users, selecting a user by ID, deleting a user by ID, inserting a new user, and updating a user. The file is named 'db.js' and is located in the 'data_base' directory. The code is as follows:

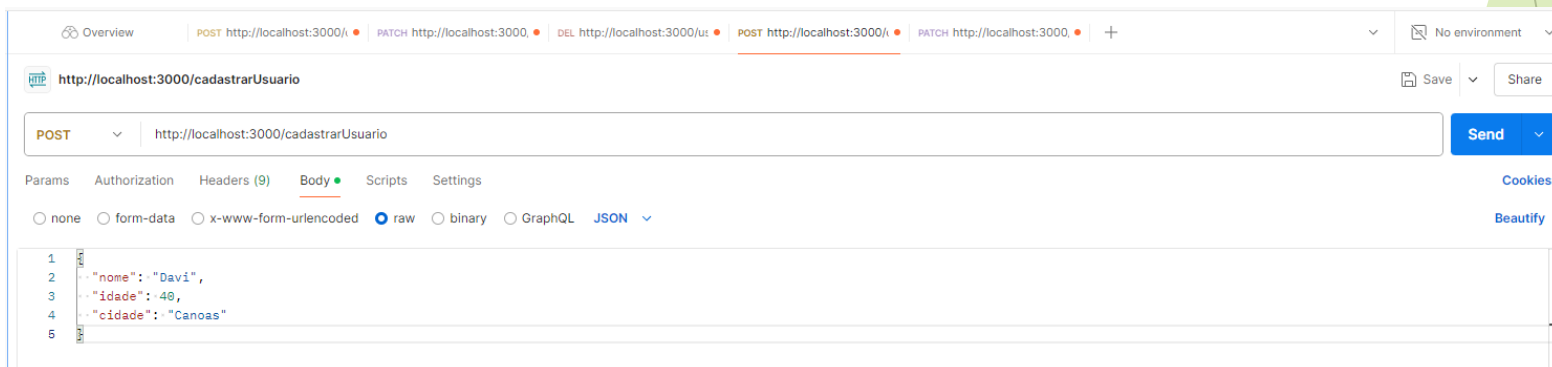
```
1
2 //Importa o modulo do mysql
3 const mysql = require('mysql2/promise');
4
5 const client = mysql.createPool(process.env.CONNECTION_STRING);
6
7
8 //busca todos os usuarios
9 async function selectUsuarios() {
10   const res = await client.query('SELECT * FROM USUARIOS');
11   return res[0];
12 }
13
14 //busca usuario pelo ID
15 async function selectUsuarioPorId(id) {
16   const res = await client.query('SELECT * FROM USUARIOS WHERE ID=?', [id]);
17   return res[0];
18 }
19
20 //deleta o usuario pelo ID
21 async function deleteUsuarioPorId(id) {
22   return await client.query('DELETE FROM USUARIOS where id=?', [id]);
23 }
24
25 //insere novo registro
26 async function insertUsuario(usuario) {
27   const sql = 'INSERT INTO USUARIOS(nome,idade,cidade) VALUES (?,?,:)';
28   const values = [usuario.nome, usuario.idade, usuario.cidade];
29   await client.query(sql, values);
30 }
31
32 //atualizar usuario
33 async function updateUsuario(id, usuario) {
34   const sql = 'UPDATE USUARIOS SET nome=?, idade=?, cidade=? WHERE id=?';
35   const values = [usuario.nome, usuario.idade, usuario.cidade, id];
36   await client.query(sql, values);
37 }
38
39 module.exports = { selectUsuarios , selectUsuarioPorId,deleteUsuarioPorId, insertUsuario, updateUsuario}
```


Postman

► Executa a rota delete

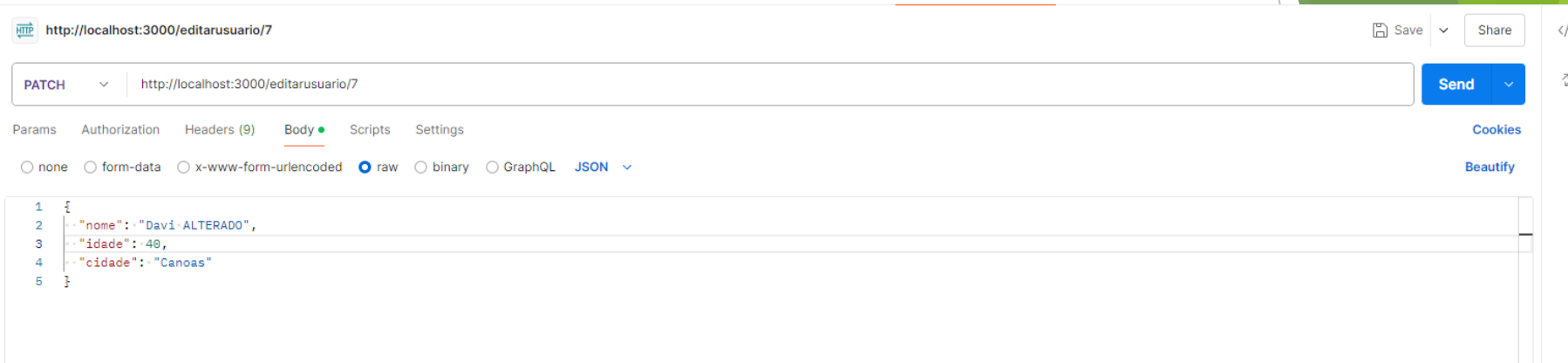


► Executa a rota insert

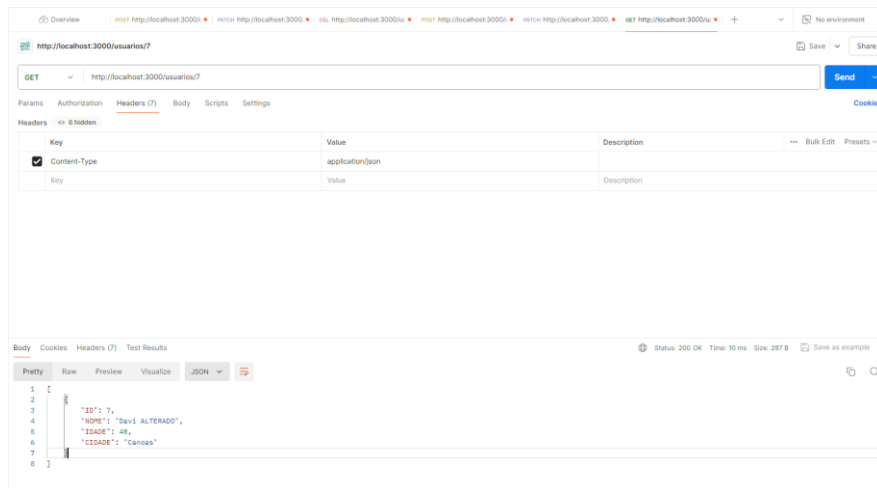


Postman

► Executa a rota update

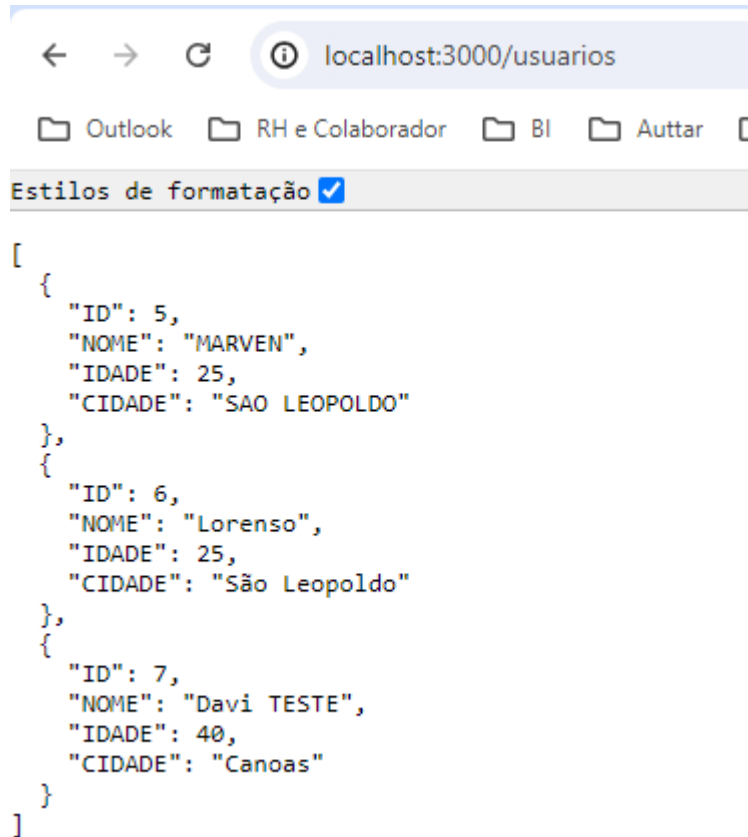


► Executa a rota buscar por ID



Postman

► Executa buscar usuarios



```
[
  {
    "ID": 5,
    "NOME": "MARVEN",
    "IDADE": 25,
    "CIDADE": "SAO LEOPOLDO"
  },
  {
    "ID": 6,
    "NOME": "Lorenzo",
    "IDADE": 25,
    "CIDADE": "São Leopoldo"
  },
  {
    "ID": 7,
    "NOME": "Davi TESTE",
    "IDADE": 40,
    "CIDADE": "Canoas"
  }
]
```