Instalando dependências

Terminal vs code:

npm i axios

npm i cors

npm i express (servidor)

npm i jsonwebtoken

npm i mysql2 (bd)

npm i nodemon

npm i dotenv

No Package.json

{

  "type":"module",

  "scripts": {

*iniciar* *o* *servidor* *em* *operação*

    "start": "nodemon index.js"

  },

  "dependencies": {

    "axios": "^1.6.8",

    "cors": "^2.8.5",

    "dotenv": "^16.4.5",

    "express": "^4.19.2",

    "jsonwebtoken": "^9.0.2",

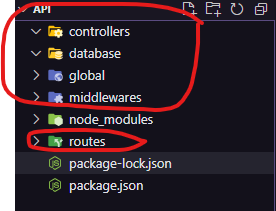
    "mysql2": "^3.9.4",

    "nodemon": "^3.1.0"

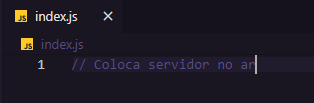
  }

}

Criar pastas no VS, raiz da API



Agora cria arquivo



No arquivo

import express from 'express';

import cors from 'cors';

import userRoutes from './routes/users.js';

const app = express();

app.use(express.json());

// Cors permite fazer requisições do frontend a meu proprio server (localhost), evitando problemas.

app.use(cors());

// Definindo rotas para o servidor, aqui se cria a rota chamada "/users"

app.use("/users", userRoutes);

// portas da url

// process.env vai ser criado, para configurar o projeto com constantes "||" expressa que

// a porta seria 8080

const port = process.env.PORT || 8080

app.listen(port, () => {

    console.log(`Servidor rodando na porta ${port}`);

})

Usa-se NPM START para rodar o servidor, no google, sem o env seria “localhost:800” o link

New file em database (db.js)

// Define a config necessaria para a comunicação com o BD

import mysql from 'mysql2';

export const db = mysql.createConnection({

    host: "localhost",

    // Porta padrão MySQL

    port: 3306,

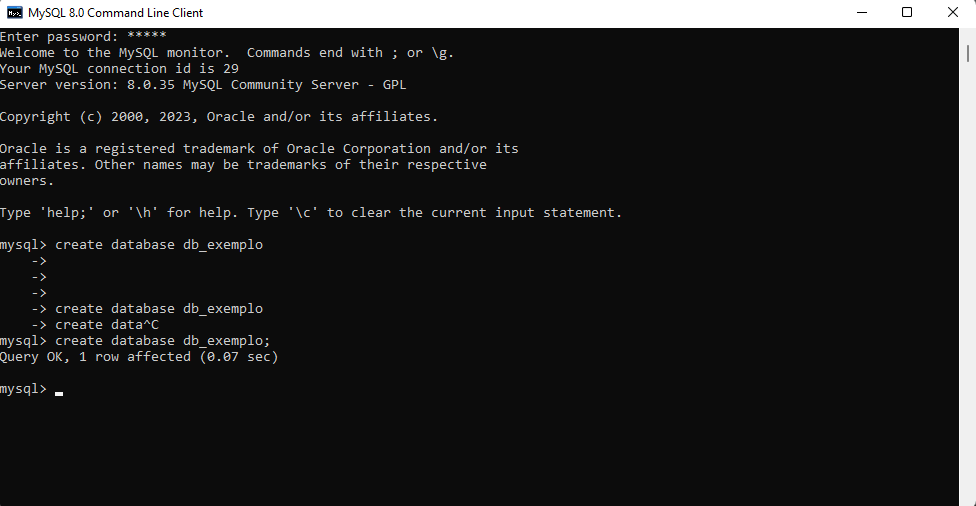
    user: "root",

    password: 'admin',

    database: "db\_exemplo"

});

(Para criar a database) após, edite-a no Dbeaver



**create** **table** usuario(

id **int** **primary** **key auto\_increment**,

nome **varchar**(45) **not** **null**,

)

De volta no VS (db.js)

// Verificação de erro

db.connect((*err*) => {

    if(err){

        console.log(`Erro de conexão ao BD: ${err}`);

        return;

    }

    console.log(`Conexão bem-sucedida`)

});

Na pasta controllers, criar UserController.js

// As chaves servem a importar a constante do arquivo

import { db } from '../database/db.js';

// O underline é o endpoint, esperando informação, sem necessidade do corpo

// aqui a requisição não precisa de processamento, por isso é omitida

export const getUsers = (*\_*, *res*) => {

    const sql = "select \* from usuario";

    db.query(sql, (*err*, *data*) => {

        if(err){

            console.log("Erro ao processar a requisição!")

            // Status code, retorna uma coisa se der erro padronizando (tipo erro 404 do google)

            return res.status(500).json(err);

        } else {

            console.log(`Dados dos usuários obtidos adequadamente!`);

            return res.status(200).json(data);

            // 500 expressa erro, já o 200, sucesso.

        }

    });

}

Na pasta routes, criar users.js

// Definir rotas

import express from 'express';

import { getUsers } from '../controllers/UserControler.js';

const router = express.Router();

// Requisição feita ao tentar acessar a rota

// router.get("/", () => {console.log("Rota de usuários acionada.")});

router.get("/", getUsers);

// Rota dentro da /users

router.get("/teste", () => {

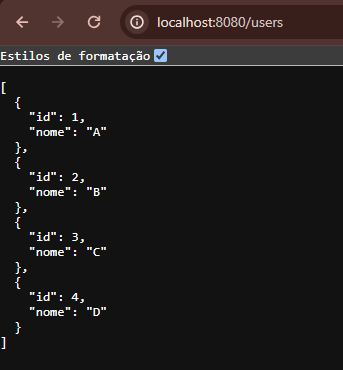
    console.log("teste adicionado")

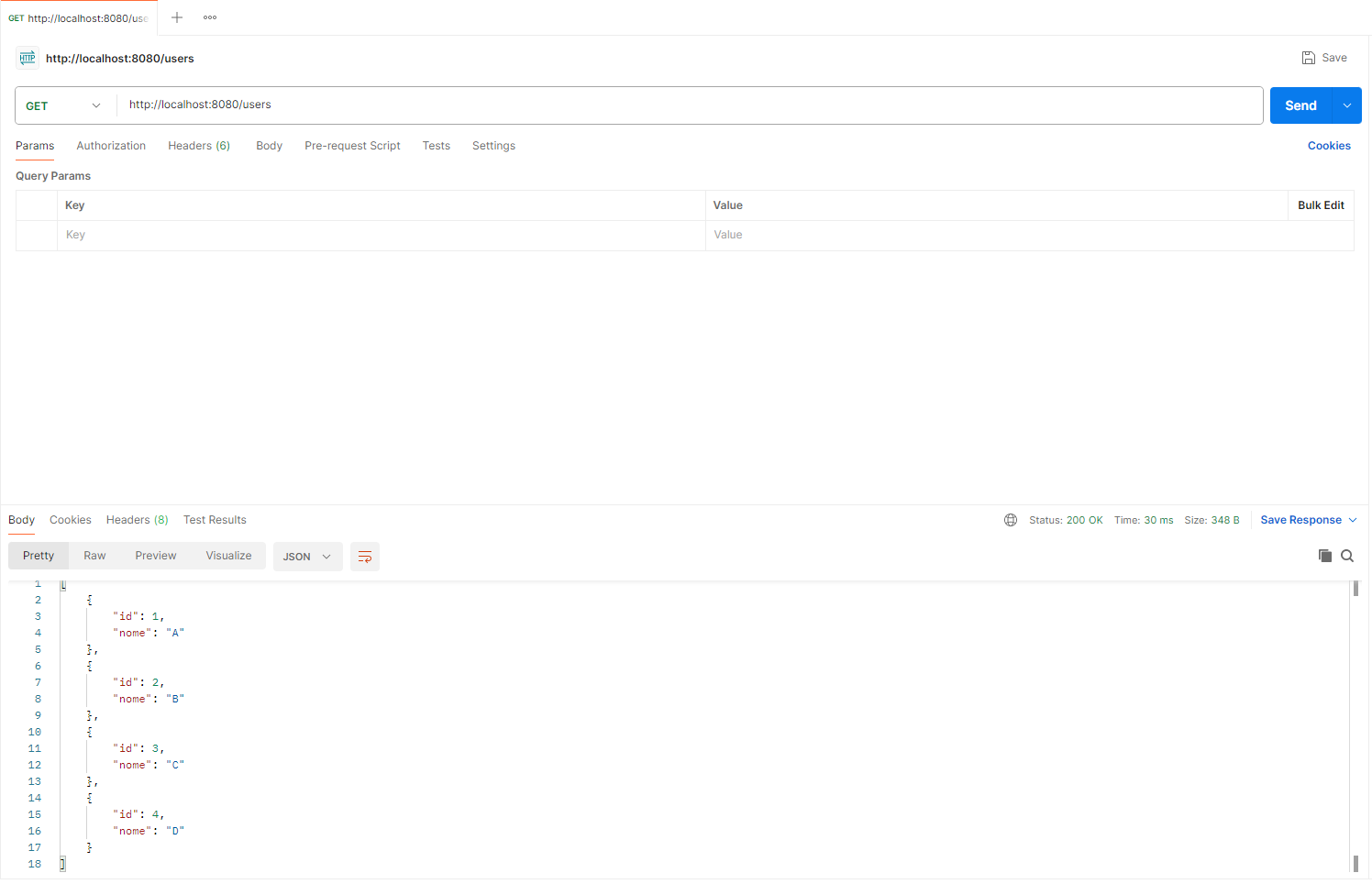
})

export default router;

// o nodemon do terminal restarta o index para cada alteração do arquivo

Ele faz rodar os usuários do BD no localhost:8080/users, até visualmente:



Testar envio de requisições com Postman

Em UserController.js (inserir)

// Recebe requisição, retorna resposta (req e res),

// na requisição, se processa o nome do usuário, por isso aqui é emitido

export const addUser = (*req*, *res*) => {

    const sql = "insert into usuario (nome) values (?)";

    const {nome} = req.body;

    // Nome a partir daqui vai ser transferido ao "?"

    db.query(sql, [nome], (*err*, *data*) => {

        if(err){

            console.log("Erro ao processar a requisição!")

            // Status code, retorna uma coisa se der erro padronizando (tipo erro 404 do google)

            return res.status(500).json(err);

        } else {

            console.log(`Usuário cadastrado adequadamente!`);

            return res.status(200).json(data);

            // 500 expressa erro, 200 sucesso.

        }

    });

}

Em users.js (inserir)

**import { getUsers, addUser } from '../controllers/UserControler.js';**

**router.post("/", addUser);**

Em UserController.js (alterar)

export const updateUser = (*req*, *res*) => {

    const sql = "update usuario set nome = ? where id = ?";

    const {id, nome} = req.body;

    // Nome a partir daqui vai ser transferido ao "?"

    db.query(sql, [nome, id], (*err*, *data*) => {

        if(err){

            console.log("Erro ao processar a requisição!")

            // Status code, retorna uma coisa se der erro padronizando (tipo erro 404 do google)

            return res.status(500).json(err);

        } else {

            console.log(`Usuário atualizado adequadamente!`);

            return res.status(200).json(data);

            // 500 expressa erro, 200 sucesso.

        }

    });

}

User.js(alterar)

import { getUsers, addUser, updateUser } from '../controllers/UserControler.js';

router.put("/", updateUser);

User.js(deletar)

import { getUsers, addUser, updateUser, deleteUser } from '../controllers/UserControler.js';

router.delete("/", deleteUser);

Em UserController.js (deletar)

export const deleteUser = (*req*, *res*) => {

    const sql = "delete from usuario where id = ?";

    const {id} = req.query;

    // Nome a partir daqui vai ser transferido ao "?"

    db.query(sql, [id], (*err*, *data*) => {

        if(err){

            console.log("Erro ao processar a requisição!")

            // Status code, retorna uma coisa se der erro padronizando (tipo erro 404 do google)

            return res.status(500).json(err);

        } else {

            console.log(`Usuário removido adequadamente!`);

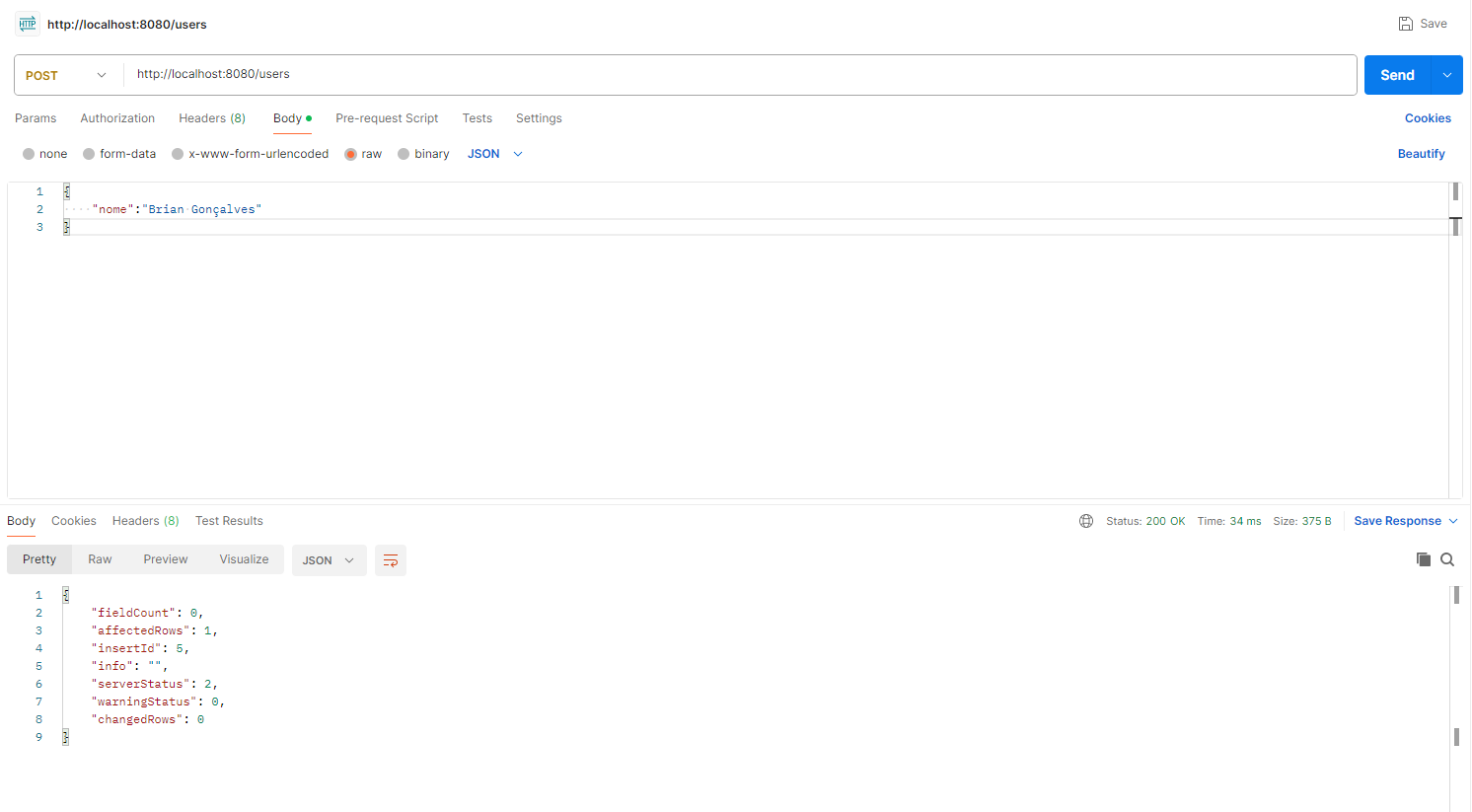
            return res.status(200).json(data);

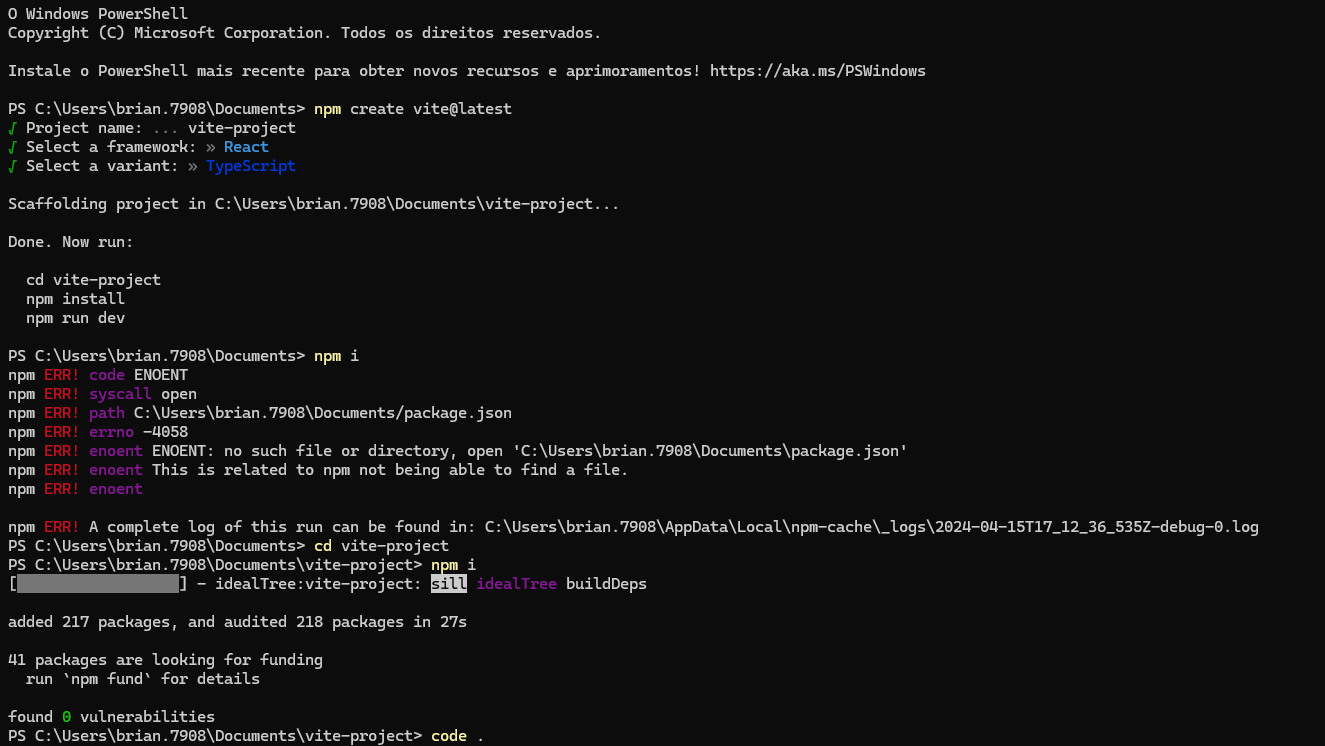
            // 500 expressa erro, 200 sucesso.

        }

    });

}

Conferindo no postman

Criando framework React (não é node)