**Projeto WebMeeting**

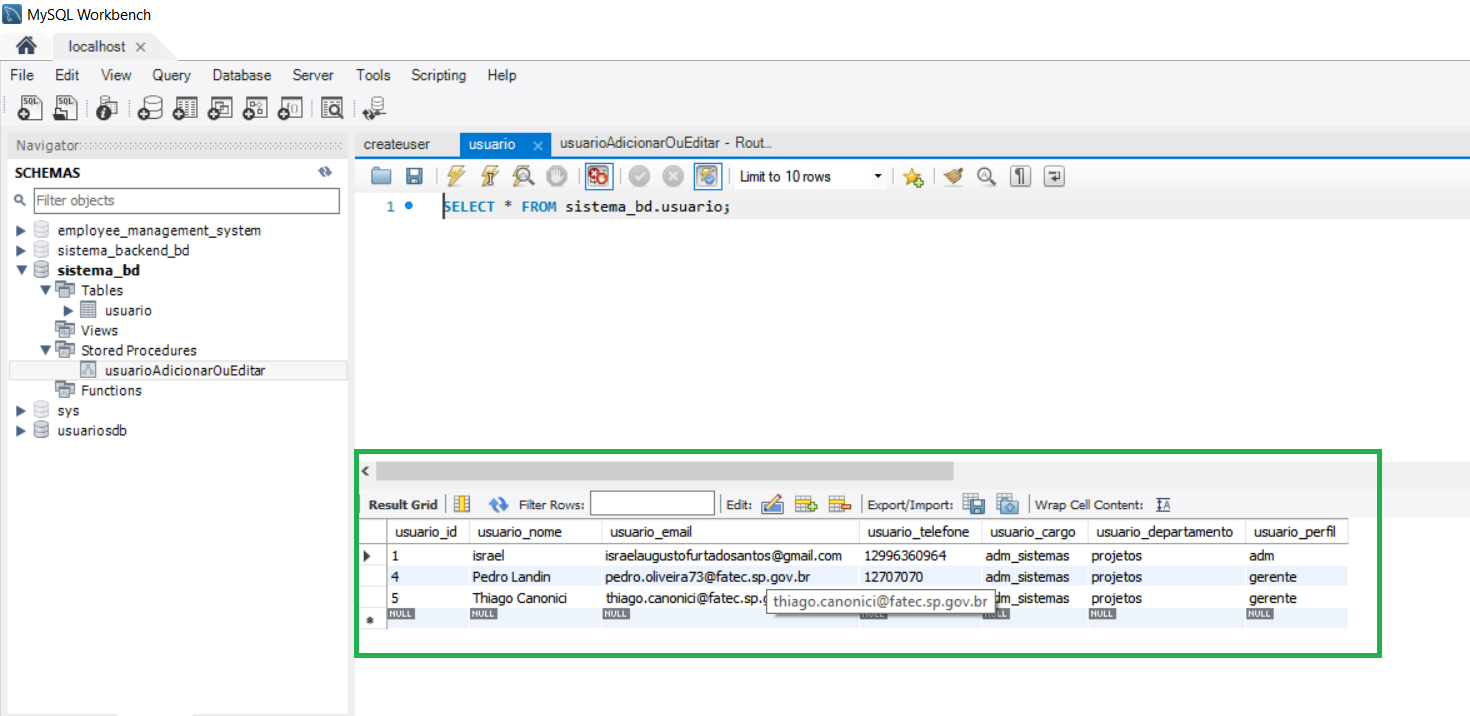
**Documentação do sistema backend para CRUD de usuário**

**Funcionalidade**

Para esta entrega foi definida a necessidade de criar um sistema para cadastro de usuários. Este sistema é dividido em ‘frontend’ e ‘backend’. Este documento explica a criação da parte de ‘backend’, que inclui o servidor responsável por receber as requisições do usuário, por meio do protocolo HTTP, e devolver uma resposta adequada. Também utiliza um banco de dados para armazenar as informações.

**O Banco de Dados**

Foi criado, utilizando MySQL, um bando de dados de teste chamado de sistema\_bd com uma única tabela usuário, conforme mostrado abaixo. A versão final do banco de dados deverá ter varias tabelas.



Script de criação da tabela usuário:

CREATE TABLE Usuario (

usuario\_id int NOT NULL AUTO\_INCREMENT,

usuario\_nome varchar(100) NOT NULL,

usuario\_email varchar(100) NOT NULL,

usuario\_telefone varchar(15) NOT NULL,

usuario\_cargo varchar(30) NOT NULL,

usuario\_departamento varchar(30) NOT NULL,

usuario\_perfil varchar(30) NOT NULL,

CONSTRAINT Usuario\_pk PRIMARY KEY (usuario\_id)

);

Além disto, para que possamos executar as requisições de criar um novo usuário e editar um usuário pelo id foi criado duas ‘procedures’, mostradas abaixo.

// define a funcao para criar novo usuario

CREATE DEFINER=`root`@`localhost` PROCEDURE `usuarioAdicionar`(

IN \_usuario\_id INT,

IN \_usuario\_nome varchar(100),

IN \_usuario\_email varchar(100),

IN \_usuario\_telefone varchar(15),

IN \_usuario\_cargo varchar(30),

IN \_usuario\_departamento varchar(30),

IN \_usuario\_perfil varchar(30)

)

BEGIN

insert into usuario (usuario\_nome, usuario\_email, usuario\_telefone,

usuario\_cargo, usuario\_departamento, usuario\_perfil)

values (\_usuario\_nome, \_usuario\_email, \_usuario\_telefone, \_usuario\_cargo,

\_usuario\_departamento, \_usuario\_perfil);

set \_usuario\_id = last\_insert\_id();

select \_usuario\_id as 'usuario\_id';

END

// funcao para editar usuario

CREATE DEFINER=`root`@`localhost` PROCEDURE `usuarioEditar`(

IN \_usuario\_id INT,

IN \_usuario\_nome varchar(100),

IN \_usuario\_email varchar(100),

IN \_usuario\_telefone varchar(15),

IN \_usuario\_cargo varchar(30),

IN \_usuario\_departamento varchar(30),

IN \_usuario\_perfil varchar(30)

)

BEGIN

update usuario

set

usuario\_nome = \_usuario\_nome,

usuario\_email = \_usuario\_email,

usuario\_telefone = \_usuario\_telefone,

usuario\_cargo = \_usuario\_cargo,

usuario\_departamento = \_usuario\_departamento,

usuario\_perfil = \_usuario\_perfil

where usuario\_id = \_usuario\_id;

select \_usuario\_id as 'usuario\_id';

END

**O servidor**

Para a criação do servidor foram utilizados node.js e Javascript. A seguir é apresentado o código do servidor.

//importação dos modulos que serão utilizados

const mysql = require('mysql');

const express = require('express');

var app = express();

const bodyparser = require('body-parser');

app.use(bodyparser.json());

// define os parametros da conexão

var mysqlConnection = mysql.createConnection({

    host: 'localhost',

    user: 'root',

    password: 'fatec',

    database: 'sistema\_bd',

    multipleStatements: true

});

// tenta realizar a conexao

mysqlConnection.connect((err) => {

    if(!err) {

        console.log('Conectado ao banco de dados com sucesso');

    } else {

        console.log(JSON.stringify(err, undefined, 2));

    }

});

// executa o servidor na porta 3000

app.listen(3000, () => {

    console.log('express server na porta 3000');

});

//Metodo Get: lista todos os usuarios

function listaUsuarios () {

        app.get('/usuarios', (req, res) => {

        mysqlConnection.query('SELECT \* FROM usuario', (err, rows, fields) => {

            if(!err) {

                res.send(rows);

            } else {

                console.log(err);

            }

        });

    })

};

listaUsuarios();

//Metodo Get: mostra o usuario com o id informado

function buscaUsuarioPorId () {

        app.get('/usuarios/:id', (req, res) => {

        mysqlConnection.query('SELECT \* FROM usuario WHERE usuario\_id = ?', [req.params.id], (err, rows, fields) => {

            if(!err) {

                console.log(rows);

                res.send(rows);

            } else {

                console.log(err);

            }

        });

    })

};

buscaUsuarioPorId();

//Metodo delete: exclui do banco o id informado

function deletaUsuarioPorId () {

        app.delete('/usuarios/:id', (req, res) => {

        mysqlConnection.query('DELETE FROM usuario WHERE usuario\_id = ?', [req.params.id], (err, rows, fields) => {

            if(!err) {

                console.log(rows);

                res.send('DELETED');

            } else {

                console.log(err);

            }

        });

    })

};

deletaUsuarioPorId();

//metodo Post: cria um novo usuario

function criaNovoUsuario () {

        app.post('/usuarios/newUser', (req, res) => {

        let usuario = req.body;

        // comando sql

        var sql = "SET @usuario\_id = ?; SET @usuario\_nome = ?; SET @usuario\_email = ?; \

        SET @usuario\_telefone = ?; SET @usuario\_cargo = ?; SET @usuario\_departamento = ?; \

        SET @usuario\_perfil = ?; \

        CALL usuarioAdicionar(@usuario\_id, @usuario\_nome, @usuario\_email, \

            @usuario\_telefone, @usuario\_cargo,  @usuario\_departamento, @usuario\_perfil);";

        // executa o comando sql criado anteriormente

        mysqlConnection.query(sql, [usuario.usuario\_id, usuario.usuario\_nome, usuario.usuario\_email,

            usuario.usuario\_telefone, usuario.usuario\_cargo, usuario.usuario\_departamento, usuario.usuario\_perfil],

            (err, rows, fields) => {

            if(!err) {

                res.send(rows);

            } else {

                console.log(err);

            }

        });

    })

};

criaNovoUsuario();

//Metodo Put: edita o usuario

function editaUsuario () {

        app.put('/usuarios/:id', (req, res) => {

        let usuario = req.body;

        var sql = "SET @usuario\_id = ?; SET @usuario\_nome = ?; SET @usuario\_email = ?; \

        SET @usuario\_telefone = ?; SET @usuario\_cargo = ?; SET @usuario\_departamento = ?; \

        SET @usuario\_perfil = ?; \

        CALL usuarioEditar(@usuario\_id, @usuario\_nome, @usuario\_email, \

            @usuario\_telefone, @usuario\_cargo,  @usuario\_departamento, @usuario\_perfil);";

        mysqlConnection.query(sql, [usuario.usuario\_id, usuario.usuario\_nome, usuario.usuario\_email,

            usuario.usuario\_telefone, usuario.usuario\_cargo, usuario.usuario\_departamento, usuario.usuario\_perfil],

            (err, rows, fields) => {

            if(!err) {

                res.send(rows);

            } else {

                console.log(err);

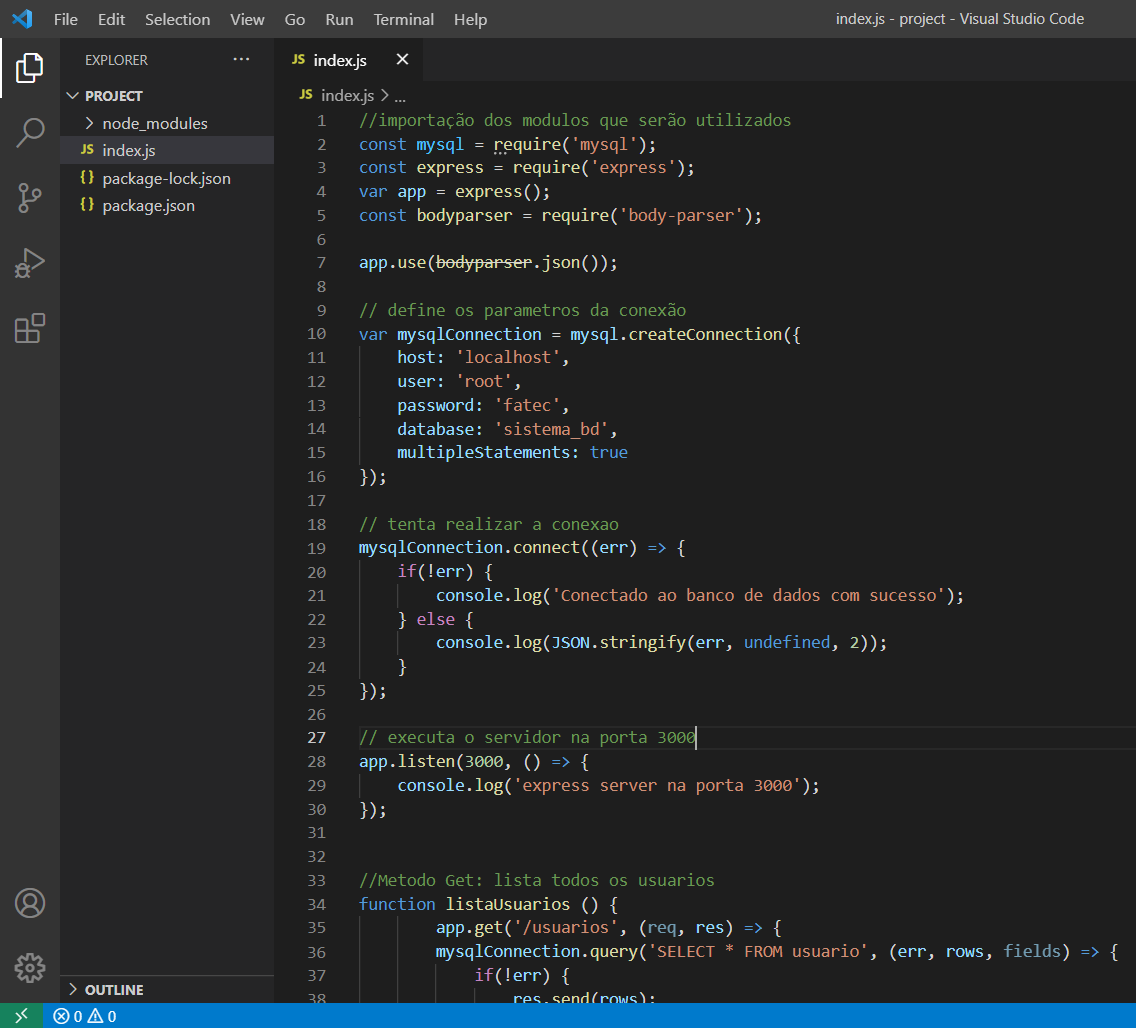
            }

        });

    })

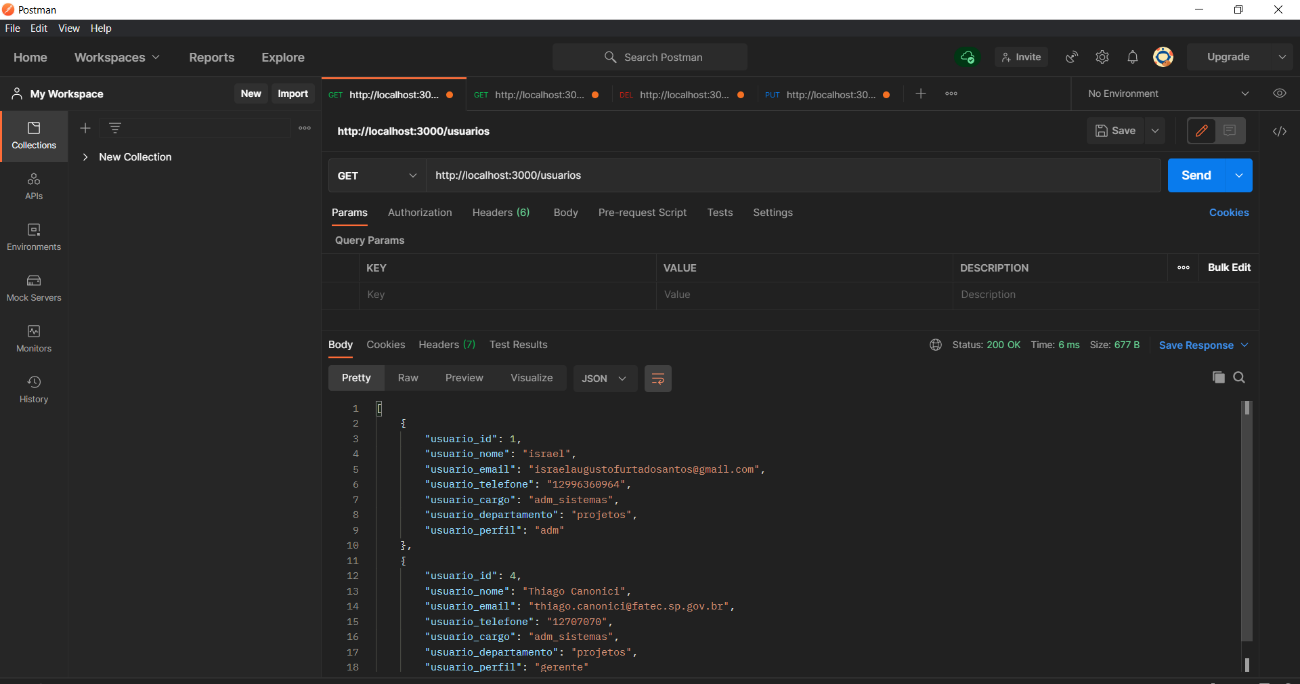
};

editaUsuario();

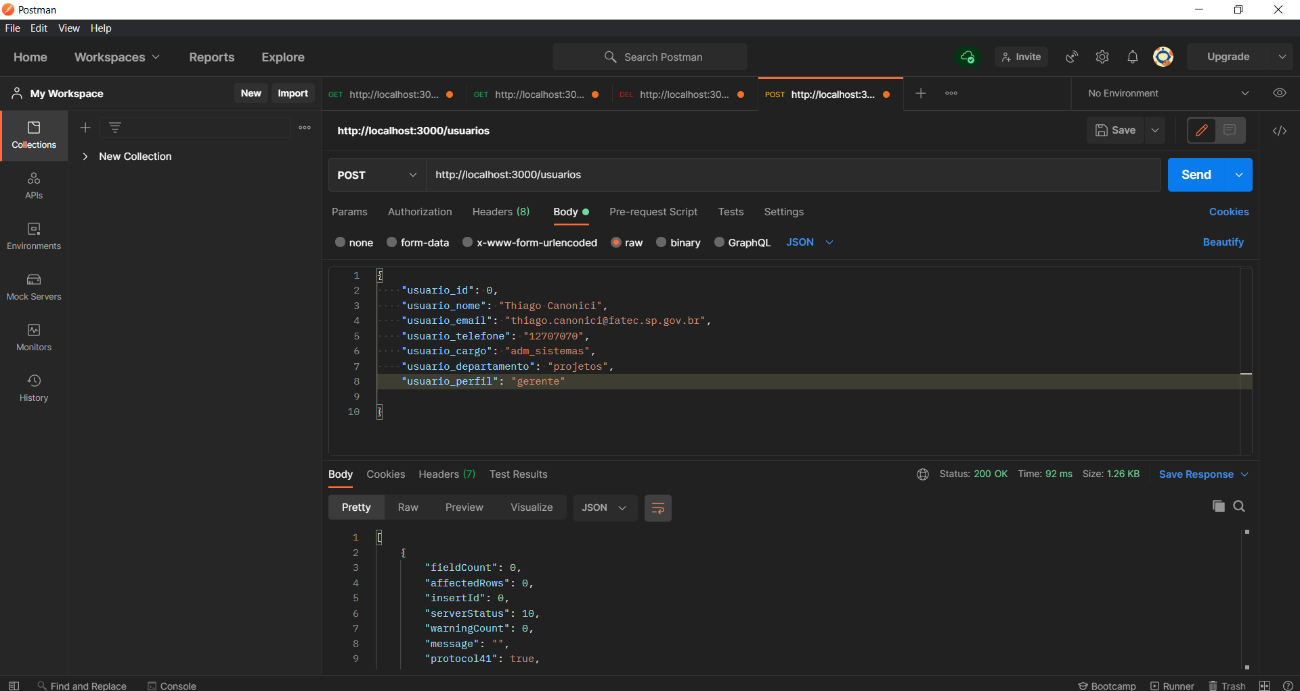


Seguem os testes feitos para confirmar o correto funcionamento do sistema.

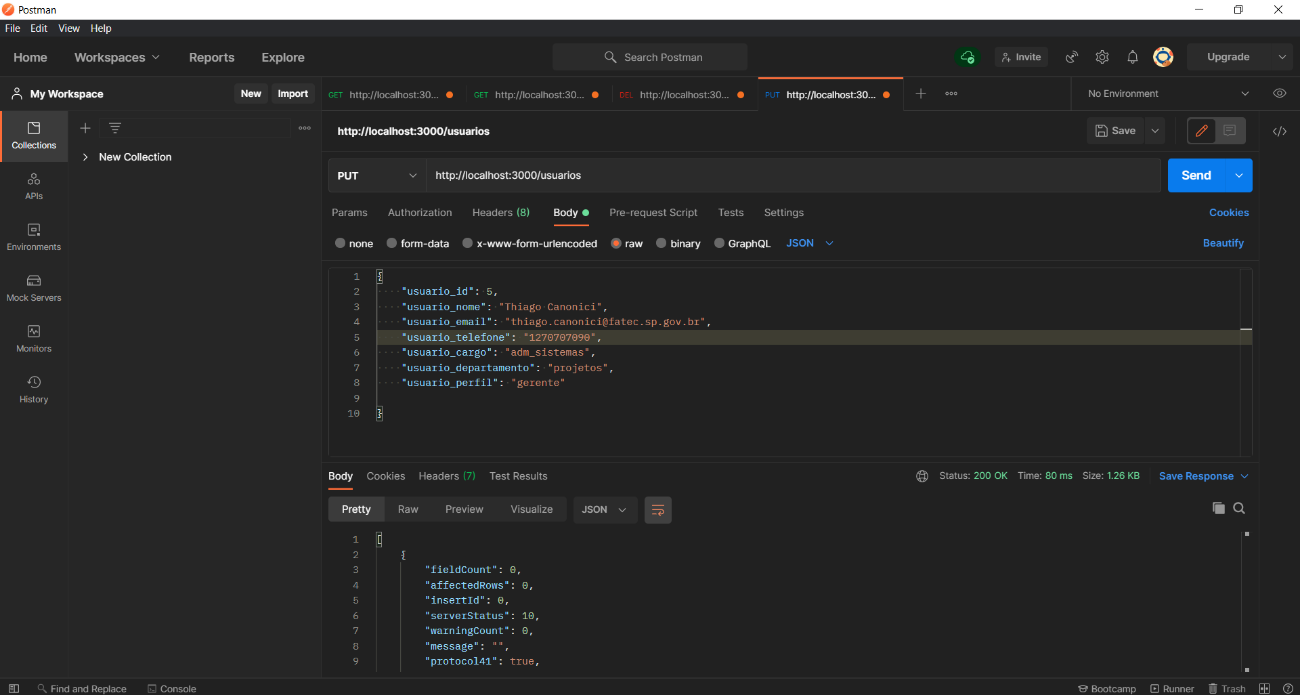
Listar os usuários:



Criar um novo usuário:



Editar um usuário pelo id:



Deletar usuário:

