Exercício 1:

Inserir uma reserva e o respetivo pagamento código python:

@app.route('/reserva/register', methods=['POST'])

def register\_reserva():

    data = request.get\_json()

    id\_utilizador = data.get("id\_utilizador")

    id\_quarto = data.get("id\_quarto")

    data\_inicio = data.get("data\_inicio")

    data\_fim = data.get("data\_fim")

    pagamento = data.get("pagamento")

*# Validação básica*

    if not all([id\_utilizador, id\_quarto, data\_inicio, data\_fim, pagamento]):

        return jsonify({"erro": "Todos os campos são obrigatórios."}), 400

    try:

        conn = get\_connection()

        cur = conn.cursor()

        cur.execute("CALL inserir\_reserva(%s, %s, %s, %s, %s)",(id\_utilizador, id\_quarto, data\_inicio, data\_fim, None))

        reserva\_id = cur.fetchone()[0]

        cur.execute("CALL inserir\_pagamento(%s, %s)", (reserva\_id, pagamento))

        conn.commit()

        cur.close()

        conn.close()

        return jsonify({"mensagem": "Reserva registada com sucesso."}), 201

    except psycopg.Error as e:

        conn.rollback()

        return jsonify({"erro": str(e)}), 400

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Os conteúdos gerados por IA poderão estar incorretos.Para visualizar o resultado utilize o seguinte link “https://bdproject-nu.vercel.app/reserva/register”:

Figura - resultado inserir reserva/pagamentos

Exercício 2:

Verificar se um quarto está disponível para reserva numa determinada data, código em python:

@app.route('/room/<room\_num>/<date>', methods=['GET'])

def get\_room\_availability(room\_num, date):

    try:

        conn = get\_connection()

        cur = conn.cursor()

        cur.execute("SELECT is\_available(%s, %s)", (room\_num, date))

        result = cur.fetchone()[0]

        conn.commit()

        cur.close()

        conn.close()

        if result is True:

            return jsonify({"disponibilidade": "Quarto disponível."}), 200

        elif result is False:

            return jsonify({"disponibilidade": "Quarto indisponível."}), 200

    except psycopg.Error as e:

        return jsonify({"erro": str(e)}), 400

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Os conteúdos gerados por IA poderão estar incorretos.Resultado para quarto indisponível com o link “bdproject-nu.vercel.app/room/123/2025-07-29”:

Figura - quarto indisponível

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Os conteúdos gerados por IA poderão estar incorretos.Resultado para quarto disponível, link “bdproject-nu.vercel.app/room/123/2025-07-31”

Figura - quarto disponivel

Devido a alguns problemas com a base de dados alocada no servidor da escola decidimos usar uma base de dados que esta alocada numa vps, hospedada na oracle cloud que roda um sitema operacional nixos que possui uma base de dados postgresql. Por esse motivo a criação de utilizadores da base de dados pode ser feita de maneira diferente.

Para criar um utilizador basta no pedaço de código da imagem adicionar “{ name = "A"; }

{ name = "B"; }” .

*# Postgres.*

    services.postgresql = {

        enable = true;

        enableTCPIP = true;

        enableJIT = true;

        ensureUsers = [

            { name = pgConf.userName; }

            { name = "A"; }

            { name = "B"; }

        ];

        ensureDatabases = [ pgConf.dbName ];

        authentication = pkgs.lib.mkOverride 10 ''

*#type  database   DBuser  auth-method*

            local  all        all     trust

            host   all        all     127.0.0.1/32   trust

            host   all        all     ::1/128        trust

            host   all        all     0.0.0.0/0      scram-sha-256

        '';

    };

Feito isto podemos verificar que os utilizadores foram criados (figura 4) mas ainda sem permissões.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Os conteúdos gerados por IA poderão estar incorretos.

Figura - confirmação da criação de utilizadores

Para adicionar as permissões e uma password, no mesmo ficheiro do código para criar adicionamos o seguinte:

*# Systemd post start script.*

systemd.services.postgresql.postStart = let

    psql = "${pkgs.postgresql}/bin/psql";

in ''

    ${psql} -U postgres -c "ALTER USER \"A\" WITH PASSWORD 'a';"

    ${psql} -U postgres -c "ALTER USER \"B\" WITH PASSWORD 'b';"

    ${psql} -U postgres -c "GRANT USAGE ON SCHEMA public TO \"A\";"

    ${psql} -U postgres -c "GRANT USAGE ON SCHEMA public TO \"B\";"

    ${psql} -U postgres -c "GRANT SELECT, UPDATE, INSERT, DELETE ON ALL TABLES IN SCHEMA public TO \"A\";"

    ${psql} -U postgres -c "GRANT SELECT, UPDATE, INSERT, DELETE ON ALL TABLES IN SCHEMA public TO \"B\";"

'';

No final fazemos o rebuild do sistema para aplicar as novas configurações.

Não foi possível implementar a ultima parte de um sistema de auditoria que registe cada vez que os endpoints (a) e (b) eram executados.