

FACULDADE DE INFORMÁTICA E ADMINISTRAÇÃO PAULISTA

CARLOS ALBERTO MACHARELLI JUNIOR - RM 551677

CARLOS EDUARDO MENDONÇA DA SILVA – RM 552164

EDUARDO TOSHIO ROCHA OKUBO – RM 551763

KAUÊ ALEXANDRE DE OLIVEIRA – RM 551812

VITOR MACHADO MIRANDA – RM 551451

DOCUMENTAÇÃO DO PROJETO BEAUTY TECH

SÃO PAULO

2024

CARLOS ALBERTO MACHARELLI JUNIOR - RM 551677
CARLOS EDUARDO MENDONÇA DA SILVA – RM 552164
EDUARDO TOSHIO ROCHA OKUBO – RM 551763
KAUÊ ALEXANDRE DE OLIVEIRA – RM 551812
VITOR MACHADO MIRANDA – RM 551451

DOCUMENTAÇÃO DO PROJETO BEAUTY TECH

Trabalho acadêmico apresentado à disciplina de Mastering Relational and Non-Relational Database do Curso Análise e Desenvolvimento de Sistemas da Faculdade de Informática e Administração Paulista como requisito de nota da Sprint 3 da Turma 2TDSPV. Requerido pelo prof. Renê de Ávila Mendes.

SUMÁRIO

| | |
|---|----------|
| 1 INTRODUÇÃO | 4 |
| 2 Projeto de banco de dados modelo relacional Beauty Tech..... | 5 |
| 3 Projeto de banco de dados modelo físico Beauty Tech..... | 6 |
| 4 Criamos procedures de INSERT/ UPDATE / DELETE | |
| 4.1 CRUD TABELA CATEGORIA | |
| 4.1.1 INSERT BT_CATEGORIA..... | 8 |
| 4.1.2 UPDATE BT_CATEGORIA..... | 8 |
| 4.1.3 DELETE BT_CATEGORIA..... | 8 |
| 4.2 CRUD TABELA CLIENTE | |
| 4.2.1 INSERT BT_CLIENTE..... | 9 |
| 4.2.2 UPDATE BT_CLIENTE..... | 10 |
| 4.2.3 DELETE BT_CLIENTE..... | 10 |
| 4.3 CRUD TABELA EMPRESA | |
| 4.3.1 INSERT BT_EMPRESA..... | 11 |
| 4.3.2 UPDATE BT_EMPRESA..... | 11 |
| 4.3.3 DELETE BT_EMPRESA..... | 12 |
| 4.4 CRUD TABELA ENDERECO | |
| 4.4.1 INSERT BT_ENDERECO..... | 12 |
| 4.4.2 UPDATE BT_ENDERECO..... | 13 |
| 4.4.2 DELETE BT_ENDERECO..... | 14 |
| 4.5 CRUD TABELA PRODUTO | |
| 4.5.1 INSERT BT_PRODUTO..... | 14 |
| 4.5.2 UPDATE BT_PRODUTO..... | 15 |
| 4.5.3 DELETE BT_PRODUTO | 15 |
| 4.6 CRUD BT_TELEFONE | |
| 4.6.1 INSERT BT_PRODUTO..... | 16 |
| 4.6.2 UPDATE BT_PRODUTO..... | 16 |
| 4.6.3 DELETE BT_PRODUTO..... | 17 |

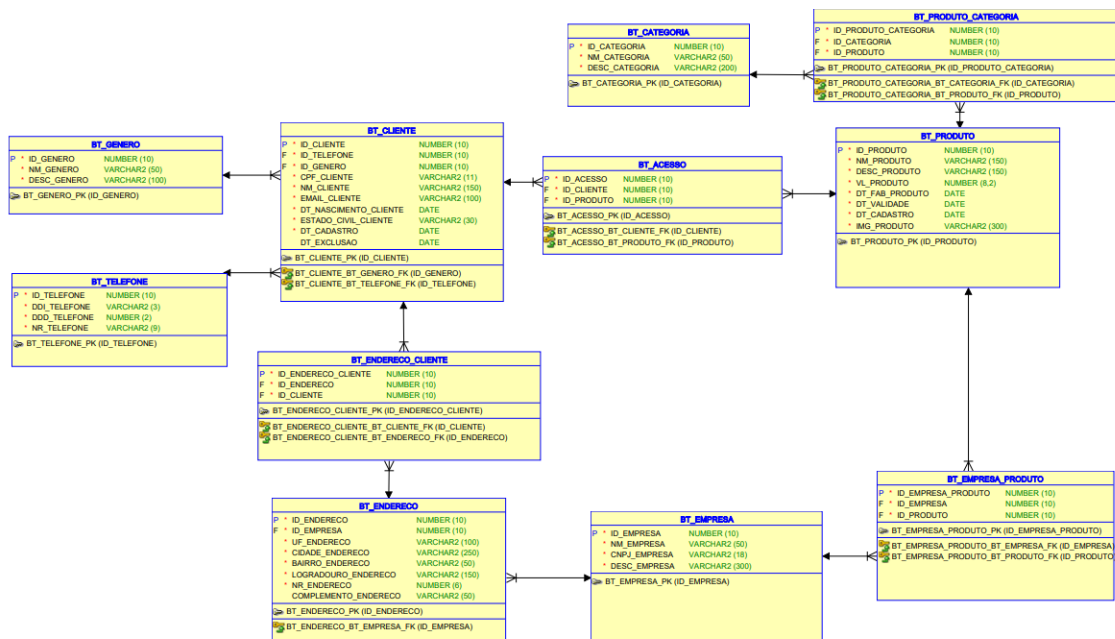
| | |
|---|----|
| 5 PROCEDURE GET_CLIENTES COM TELEFONE..... | 18 |
| 6 PROCEDURE CR_RELATORIO_ACESSOS_CLIENTE..... | 18 |
| 7 ARQUIVO PYTHON PARA CHAMADA DAS PROCEDURES..... | 19 |
| 8 FUNÇÃO VALIDAR CPF..... | 21 |
| 8.1 TESTE CORRETO FUNÇÃO VALIDAR CPF | |
| 8.2 TESTE COM ERROS FUNÇÃO VALIDAR E-MAIL | |
| 9 FUNÇÃO VALIDAR EMAIL..... | 23 |
| 9.1 TESTE CORRETO FUNÇÃO VALIDAR EMAIL | |
| 9.2 TESTE COM ERROS FUNÇÃO VALIDAR EMAIL | |
| 10 FUNÇÃO VERIFICAR MAIORIDADE..... | 24 |
| 10.1 SELECT TESTANDO FUNÇÃO VERIFICAR MAIORIDADE | |
| 11 FUNÇÃO TRANSFORMANDO EM JSON..... | 25 |
| 12 PROCEDURE GERAR JSON CLIENTE..... | 26 |
| 12.1 TESTE CORRETO PROCEDURE GERAR JSON CLIENTE | |
| 12.2 TESTE COM ERROS PROCEDURE GERAR JSON CLIENTE | |
| 13 PROCEDURE IMPRIMINDO VALORES..... | 27 |
| 13.1 TESTE PROCEDURE IMPRIMINDO VALORES | |
| 14 CRIAÇÃO TABELA AUDITORIA..... | 27 |
| 15 TRIGGER AUDITORIA BT CLIENTE..... | 28 |
| 15.1 TESTE TRIGGER COM INSERT | |
| 15.2 TESTE TRIGGER COM UPDATE | |
| 15.3 TESTE TRIGGER COM DELETE | |

1 INTRODUÇÃO

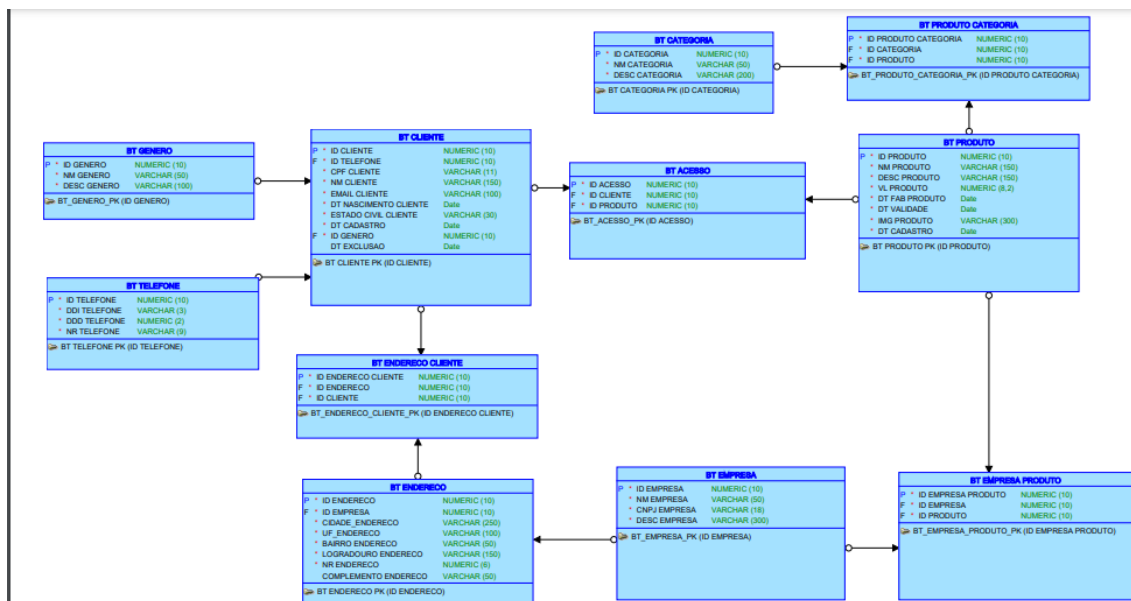
Nosso projeto, Beauty Tech, surge para abordar os desafios enfrentados no setor de cosméticos, onde a abundância de opções pode sobrecarregar os consumidores. O primeiro desafio é o desengajamento dos clientes na busca por produtos específicos devido à vasta oferta disponível. Em seguida, a gestão de estoque se torna um desafio, com excesso de produtos menos populares e escassez dos mais procurados. Por último, a intensa competição no mercado exige estratégias inovadoras para se destacar.

Nossa solução será um sistema de consulta e busca, que auxiliará o público-alvo na busca de produtos cosméticos que melhor se identifiquem com as categorias desejadas por ele (foco, cor, tipo de pele, clima, etc.), listando os produtos desejados de forma precisa e rápida. Essas buscas abrangerão clientes híbridos e digitais, já que o sistema estará disponível tanto no site quanto no aplicativo para vendas online, além de estar integrado em totens nas lojas físicas. Isso ajudará as lojas físicas a oferecerem assistência e venda dos produtos de forma mais eficiente, mostrando quantidade, especificações e detalhes, o que pode reduzir a necessidade de funcionários para assistência e aumentar a precisão das buscas para os clientes.

2 Projeto de banco de dados modelo relacional Beauty Tech:



3 Projeto de banco de dados modelo físico Beauty Tech



4 Criamos procedures de INSERT/ UPDATE / DELETE:

4.1 CRUD TABELA BT_CATEGORIA

4.1.1 INSERT BT_CATEGORIA

```
CREATE OR REPLACE PROCEDURE insert_bt_categoria (
    p_id_categoria    IN bt_categoria.id_categoria%TYPE,
    p_nm_categoria    IN bt_categoria.nm_categoria%TYPE,
    p_desc_categoria  IN bt_categoria.desc_categoria%TYPE
) IS
BEGIN
    INSERT INTO bt_categoria (id_categoria, nm_categoria, desc_categoria)
    VALUES (p_id_categoria, p_nm_categoria, p_desc_categoria);
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Não podemos ter duplicidade de ID
na tabela, faça o insert com outro ID.');
```

```
COMMIT;
END;
/
```

4.1.2 UPDATE BT_CATEGORIA

```
CREATE OR REPLACE PROCEDURE update_bt_categoria (
    p_id_categoria    IN bt_categoria.id_categoria%TYPE,
    p_nm_categoria    IN bt_categoria.nm_categoria%TYPE,
    p_desc_categoria  IN bt_categoria.desc_categoria%TYPE
) IS
BEGIN
    UPDATE bt_categoria
    SET nm_categoria = p_nm_categoria,
        desc_categoria = p_desc_categoria
    WHERE id_categoria = p_id_categoria;

    -- Verifica se alguma linha foi atualizada
    IF SQL%ROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Erro: ID da Categoria a ser atualizada
não encontrada.');
```

```
END IF;
COMMIT;
END;
/
```

4.1.3 DELETE BT_CATEGORIA

```
CREATE OR REPLACE PROCEDURE delete_bt_categoria (
    p_id_categoria    IN bt_categoria.id_categoria%TYPE
) IS
BEGIN
```



```

DELETE FROM bt_categoria
WHERE id_categoria = p_id_categoria;

IF SQL%NOTFOUND THEN
    DBMS_OUTPUT.PUT_LINE('Erro: Categoria a ser excluída não encontrada.');
```

```
END IF;
```

```
COMMIT;
```

```
END;
```

```
/
```

4.2 CRUD TABELA CLIENTE

4.2.1 INSERT BT_CLIENTE

```

CREATE OR REPLACE PROCEDURE insert_bt_cliente (
    p_id_cliente          IN bt_cliente.id_cliente%TYPE,
    p_id_telefone         IN bt_cliente.id_telefone%TYPE,
    p_id_genero           IN bt_cliente.id_genero%TYPE,
    p_cpf_cliente        IN bt_cliente.cpf_cliente%TYPE,
    p_nm_cliente         IN bt_cliente.nm_cliente%TYPE,
    p_email_cliente      IN bt_cliente.email_cliente%TYPE,
    p_dt_nascimento_cliente IN bt_cliente.dt_nascimento_cliente%TYPE,
    p_estado_civil_cliente IN bt_cliente.estado_civil_cliente%TYPE,
    p_dt_cadastro         IN bt_cliente.dt_cadastro%TYPE,
    p_dt_exclusao         IN bt_cliente.dt_exclusao%TYPE DEFAULT N
) IS
BEGIN
    INSERT INTO bt_cliente (
        id_cliente,
        id_telefone,
        id_genero,
        cpf_cliente,
        nm_cliente,
        email_cliente,
        dt_nascimento_cliente,
        estado_civil_cliente,
        dt_cadastro,
        dt_exclusao
    ) VALUES (
        p_id_cliente,
        p_id_telefone,
        p_id_genero,
        p_cpf_cliente,
        p_nm_cliente,
        p_email_cliente,
        p_dt_nascimento_cliente,
        p_estado_civil_cliente,
        p_dt_cadastro,
        p_dt_exclusao,
    )
```

```

        p_dt_exclusao
    );
    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Não podemos ter duplicidade de ID
na tabela, faça o insert com outro ID.');
```

COMMIT;

END;

/

4.2.2 UPDATE BT_CLIENTE

```

CREATE OR REPLACE PROCEDURE update_bt_cliente (
    p_id_cliente          IN bt_cliente.id_cliente%TYPE,
    p_id_telefone         IN bt_cliente.id_telefone%TYPE,
    p_id_genero           IN bt_cliente.id_genero%TYPE,
    p_cpf_cliente         IN bt_cliente.cpf_cliente%TYPE,
    p_nm_cliente          IN bt_cliente.nm_cliente%TYPE,
    p_email_cliente       IN bt_cliente.email_cliente%TYPE,
    p_dt_nascimento_cliente IN bt_cliente.dt_nascimento_cliente%TYPE,
    p_estado_civil_cliente IN bt_cliente.estado_civil_cliente%TYPE,
    p_dt_exclusao         IN bt_cliente.dt_exclusao%TYPE
) IS
BEGIN
    UPDATE bt_cliente
    SET
        id_telefone = p_id_telefone,
        id_genero = p_id_genero,
        cpf_cliente = p_cpf_cliente,
        nm_cliente = p_nm_cliente,
        email_cliente = p_email_cliente,
        dt_nascimento_cliente = p_dt_nascimento_cliente,
        estado_civil_cliente = p_estado_civil_cliente,
        dt_exclusao = p_dt_exclusao
    WHERE
        id_cliente = p_id_cliente;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro: ID do Cliente a ser atualizado não
o encontrado.');
```

END IF;

COMMIT;

END;

/

4.2.3 DELETE BT_CLIENTE

```

CREATE OR REPLACE PROCEDURE delete_bt_cliente (
    p_id_cliente IN bt_cliente.id_cliente%TYPE
) IS
BEGIN
```

```

DELETE FROM bt_cliente
WHERE id_cliente = p_id_cliente;

IF SQL%NOTFOUND THEN
    DBMS_OUTPUT.PUT_LINE('Erro: Cliente a ser excluído não encontrado.');
```

END IF;

COMMIT;

END;

/

4.3 CRUD TABELA BT_EMPRESA

4.3.1 INSERT BT_EMPRESA

```

CREATE OR REPLACE PROCEDURE insert_bt_empresa (
    p_id_empresa    IN bt_empresa.id_empresa%TYPE,
    p_nm_empresa    IN bt_empresa.nm_empresa%TYPE,
    p_cnpj_empresa  IN bt_empresa.cnpj_empresa%TYPE,
    p_desc_empresa  IN bt_empresa.desc_empresa%TYPE
) IS
BEGIN
    INSERT INTO bt_empresa (
        id_empresa,
        nm_empresa,
        cnpj_empresa,
        desc_empresa
    ) VALUES (
        p_id_empresa,
        p_nm_empresa,
        p_cnpj_empresa,
        p_desc_empresa
    );
    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Não podemos ter duplicidade de ID na tabela, faça o insert com outro ID.');
```

COMMIT;

END;

/

4.3.2 UPDATE BT_EMPRESA

```

CREATE OR REPLACE PROCEDURE update_bt_empresa (
    p_id_empresa    IN bt_empresa.id_empresa%TYPE,
    p_nm_empresa    IN bt_empresa.nm_empresa%TYPE,
    p_cnpj_empresa  IN bt_empresa.cnpj_empresa%TYPE,
    p_desc_empresa  IN bt_empresa.desc_empresa%TYPE
) IS
BEGIN
    UPDATE bt_empresa
```

```

SET
    nm_empresa = p_nm_empresa,
    cnpj_empresa = p_cnpj_empresa,
    desc_empresa = p_desc_empresa
WHERE
    id_empresa = p_id_empresa;

IF SQL%NOTFOUND THEN
    DBMS_OUTPUT.PUT_LINE('Erro: ID da Empresa a ser atualizada não
o encontrado.');
```

o encontrado.');

```

END IF;
COMMIT;
END;
/
```

4.3.3 DELETE BT_EMPRESA

```

CREATE OR REPLACE PROCEDURE delete_bt_empresa (
    p_id_empresa IN bt_empresa.id_empresa%TYPE
) IS
BEGIN
    DELETE FROM bt_empresa
    WHERE id_empresa = p_id_empresa;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Empresa a ser excluída não encont
rada.');
```

rada.');

```

    END IF;
    COMMIT;
END;
/
```

4.4 CRUD BT_ENDERECO

4.4.1 INSERT BT_ENDERECO

```

CREATE OR REPLACE PROCEDURE insert_bt_endereco (
    p_id_endereco          IN bt_endereco.id_endereco%TYPE,
    p_id_empresa           IN bt_endereco.id_empresa%TYPE,
    p_uf_endereco          IN bt_endereco.uf_endereco%TYPE,
    p_cidade_endereco      IN bt_endereco.cidade_endereco%TYPE,
    p_bairro_endereco      IN bt_endereco.bairro_endereco%TYPE,
    p_logradouro_endereco  IN bt_endereco.logradouro_endereco%TYPE,
    p_nr_endereco          IN bt_endereco.nr_endereco%TYPE,
    p_complemento_endereco IN bt_endereco.complemento_endereco%TYPE D
EFAULT NULL
) IS
BEGIN
    INSERT INTO bt_endereco (
        id_endereco,
        id_empresa,
        uf_endereco,
        cidade_endereco,
        bairro_endereco,
```

```

        logradouro_endereco,
        nr_endereco,
        complemento_endereco
    ) VALUES (
        p_id_endereco,
        p_id_empresa,
        p_uf_endereco,
        p_cidade_endereco,
        p_bairro_endereco,
        p_logradouro_endereco,
        p_nr_endereco,
        p_complemento_endereco
    );
    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Não podemos ter duplicidade de ID
na tabela, faça o insert com outro ID.');
```

```

    COMMIT;
END;
/
```

4.4.2 ATUALIZAR BT_ENDERECO

```

CREATE OR REPLACE PROCEDURE update_bt_endereco (
    p_id_endereco          IN bt_endereco.id_endereco%TYPE,
    p_id_empresa           IN bt_endereco.id_empresa%TYPE,
    p_uf_endereco           IN bt_endereco.uf_endereco%TYPE,
    p_cidade_endereco       IN bt_endereco.cidade_endereco%TYPE,
    p_bairro_endereco       IN bt_endereco.bairro_endereco%TYPE,
    p_logradouro_endereco   IN bt_endereco.logradouro_endereco%TYPE,
    p_nr_endereco           IN bt_endereco.nr_endereco%TYPE,
    p_complemento_endereco IN bt_endereco.complemento_endereco%TYPE
) IS
BEGIN
    UPDATE bt_endereco
    SET
        id_empresa = p_id_empresa,
        uf_endereco = p_uf_endereco,
        cidade_endereco = p_cidade_endereco,
        bairro_endereco = p_bairro_endereco,
        logradouro_endereco = p_logradouro_endereco,
        nr_endereco = p_nr_endereco,
        complemento_endereco = p_complemento_endereco
    WHERE
        id_endereco = p_id_endereco;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro: ID do Endereço a ser atualizado não encontrado.');
```

```

    END IF;
    COMMIT;
END;
```

/

4.4.3 DELETE BT_ENDERECO

```

CREATE OR REPLACE PROCEDURE delete_bt_endereco (
    p_id_endereco IN bt_endereco.id_endereco%TYPE
) IS
BEGIN
    DELETE FROM bt_endereco
    WHERE id_endereco = p_id_endereco;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Endereço a ser excluído não encontrado. ');
    END IF;
COMMIT;
END;
/

```

4.5 CRUD BT_PRODUTO

4.5.1 INSERT BT_PRODUTO

```

CREATE OR REPLACE PROCEDURE insert_bt_produto (
    p_id_produto      IN bt_produto.id_produto%TYPE,
    p_nm_produto      IN bt_produto.nm_produto%TYPE,
    p_desc_produto    IN bt_produto.desc_produto%TYPE,
    p_vl_produto      IN bt_produto.vl_produto%TYPE,
    p_dt_fab_produto  IN bt_produto.dt_fab_produto%TYPE,
    p_dt_validade     IN bt_produto.dt_validade%TYPE,
    p_dt_cadastro     IN bt_produto.dt_cadastro%TYPE,
    p_img_produto     IN bt_produto.img_produto%TYPE
) IS
BEGIN
    INSERT INTO bt_produto (
        id_produto,
        nm_produto,
        desc_produto,
        vl_produto,
        dt_fab_produto,
        dt_validade,
        dt_cadastro,
        img_produto
    ) VALUES (
        p_id_produto,
        p_nm_produto,
        p_desc_produto,
        p_vl_produto,
        p_dt_fab_produto,
        p_dt_validade,
        p_dt_cadastro,
        p_img_produto
    );
EXCEPTION

```

```

        WHEN DUP_VAL_ON_INDEX THEN
            DBMS_OUTPUT.PUT_LINE('Erro: Não podemos ter duplicidade de ID
na tabela, faça o insert com outro ID.');
```

COMMIT;

END;

/

4.5.2 UPDATE BT_PRODUTO

```

CREATE OR REPLACE PROCEDURE update_bt_produto (
    p_id_produto      IN bt_produto.id_produto%TYPE,
    p_nm_produto      IN bt_produto.nm_produto%TYPE,
    p_desc_produto    IN bt_produto.desc_produto%TYPE,
    p_vl_produto      IN bt_produto.vl_produto%TYPE,
    p_dt_fab_produto  IN bt_produto.dt_fab_produto%TYPE,
    p_dt_validade     IN bt_produto.dt_validade%TYPE,
    p_dt_cadastro     IN bt_produto.dt_cadastro%TYPE,
    p_img_produto     IN bt_produto.img_produto%TYPE
) IS
BEGIN
    UPDATE bt_produto
    SET
        nm_produto = p_nm_produto,
        desc_produto = p_desc_produto,
        vl_produto = p_vl_produto,
        dt_fab_produto = p_dt_fab_produto,
        dt_validade = p_dt_validade,
        dt_cadastro = p_dt_cadastro,
        img_produto = p_img_produto
    WHERE
        id_produto = p_id_produto;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro: ID do Produto a ser atualizado não
o encontrado.');
```

END IF;

COMMIT;

END;

/

4.5.3 DELETE BT_PRODUTO

```

CREATE OR REPLACE PROCEDURE delete_bt_produto (
    p_id_produto IN bt_produto.id_produto%TYPE,
    p_nm_produto OUT bt_produto.nm_produto%TYPE
) IS
BEGIN
```

```

SELECT nm_produto INTO p_nm_produto
FROM bt_produto
WHERE id_produto = p_id_produto;

DELETE FROM bt_produto
WHERE id_produto = p_id_produto;

IF SQL%NOTFOUND THEN
    DBMS_OUTPUT.PUT_LINE('Erro: Produto a ser excluído não encontrado. ');
END IF;

COMMIT;

END;
/

```

4.6 CRUD BT_TELEFONE

4.6.1 INSERT BT_TELEFONE

```

CREATE OR REPLACE PROCEDURE insert_bt_telefone (
    p_id_telefone IN bt_telefone.id_telefone%TYPE,
    p_ddi_telefone IN bt_telefone.ddi_telefone%TYPE,
    p_ddd_telefone IN bt_telefone.ddd_telefone%TYPE,
    p_nr_telefone IN bt_telefone.nr_telefone%TYPE
) IS
BEGIN
    INSERT INTO bt_telefone (
        id_telefone,
        ddi_telefone,
        ddd_telefone,
        nr_telefone
    ) VALUES (
        p_id_telefone,
        p_ddi_telefone,
        p_ddd_telefone,
        p_nr_telefone
    );
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Não podemos ter duplicidade de ID na tabela, faça o insert com outro ID. ');
COMMIT;
END;

/

```

4.6.2 UPDATE BT_TELEFONE

```

CREATE OR REPLACE PROCEDURE update_bt_telefone (
    p_id_telefone IN bt_telefone.id_telefone%TYPE,
    p_ddi_telefone IN bt_telefone.ddi_telefone%TYPE,

```

SÃO PAULO


```

        p_ddd_telefone IN bt_telefone.ddd_telefone%TYPE,
        p_nr_telefone  IN bt_telefone.nr_telefone%TYPE
    ) IS
BEGIN
    UPDATE bt_telefone
    SET
        ddi_telefone = p_ddi_telefone,
        ddd_telefone = p_ddd_telefone,
        nr_telefone = p_nr_telefone
    WHERE
        id_telefone = p_id_telefone;

    IF SQL%NOTFOUND THEN
        DBMS_OUTPUT.PUT_LINE('Erro: ID do telefone a ser atualizado n
ão encontrado.');
```

4.6.3 DELETE BT_TELEFONE

```

CREATE OR REPLACE PROCEDURE delete_bt_telefone (
    p_id_telefone IN bt_telefone.id_telefone%TYPE,
    p_nr_telefone OUT bt_telefone.nr_telefone%TYPE
) IS
BEGIN

    SELECT nr_telefone INTO p_nr_telefone
    FROM bt_telefone
    WHERE id_telefone = p_id_telefone;

    DELETE FROM bt_telefone
    WHERE id_telefone = p_id_telefone;

    IF SQL%ROWCOUNT = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Erro: Telefone a ser excluído não encon
trado.');
```

5 Criamos a procedure GET_CLIENTES_COM_TELEFONE que imprime o nome e número de telefone do cliente.

```
CREATE OR REPLACE PROCEDURE get_clientes_com_telefones IS
    CURSOR c_clientes_com_telefones IS
        SELECT
            c.id_cliente,
            c.nm_cliente,
            t.ddd_telefone,
            t.nr_telefone
        FROM
            bt_cliente c
        JOIN
            bt_telefone t ON c.id_telefone = t.id_telefone;
BEGIN
    FOR r IN c_clientes_com_telefones LOOP
        DBMS_OUTPUT.PUT_LINE('ID Cliente: ' || r.id_cliente ||
                               ', Nome: ' || r.nm_cliente ||
                               ', DDD: ' || r.ddd_telefone ||
                               ', Número: ' || r.nr_telefone);
    END LOOP;
END;
/
```

6 Criamos a procedure CR_RELATORIO_ACESSOS_CLIENTE, que nos mostra a quantidade de acessos de cada cliente, último produto acessado e seu nome para identificação.

```
CREATE OR REPLACE PROCEDURE cr_relatorio_acessos_cliente IS
BEGIN
    -- Exibindo o relatório com a quantidade de produtos acessados por cliente
    FOR r IN (
        SELECT
            c.nm_cliente AS NomeCliente,
            COUNT(a.id_acesso) AS QuantidadeAcessos,
            MAX(p.nm_produto) AS UltimoProdutoAcessado
        FROM
            bt_cliente c
            INNER JOIN bt_acesso a ON c.id_cliente = a.id_cliente
            INNER JOIN bt_produto p ON a.id_produto = p.id_produto
        GROUP BY
            c.nm_cliente
        ORDER BY
            QuantidadeAcessos DESC
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Nome do Cliente: ' || r.NomeCliente);
        DBMS_OUTPUT.PUT_LINE('Quantidade de Acessos: ' || r.QuantidadeAcessos);
        DBMS_OUTPUT.PUT_LINE('Último Produto Acessado: ' || r.UltimoProdutoAcessado);
        DBMS_OUTPUT.PUT_LINE('-----');
    END LOOP;
END;
/
```

7 ARQUIVO PYTHON PARA CHAMADA DAS PROCEDURES

```

import oracledb
from datetime import datetime

# Configurações da conexão
conexao = oracledb.connect(user="rm551451", password="fiap23", dsn="oracle.fiap.com.br/orcl")

# Função para chamar a procedure insert_bt_telefone
def inserir_bt_telefone(id_telefone, ddi_telefone, ddd_telefone, nr_telefone):
    try:
        with conexao.cursor() as cursor:
            cursor.callproc("insert_bt_telefone", [id_telefone, ddi_telefone, ddd_telefone, nr_telefone])
            print(f"Inserido: {id_telefone}, {nr_telefone}")
    except oracledb.Error as e:
        print(f"Erro ao inserir {id_telefone}: {e}")

# Função para chamar a procedure update_bt_telefone
def atualizar_bt_telefone(id_telefone, ddi_telefone, ddd_telefone, nr_telefone):
    try:
        with conexao.cursor() as cursor:
            cursor.callproc("update_bt_telefone", [id_telefone, ddi_telefone, ddd_telefone, nr_telefone])
            print(f"Atualizado: {id_telefone}, {nr_telefone}")
    except oracledb.Error as e:
        print(f"Erro ao atualizar {id_telefone}: {e}")

```

```

# Função para chamar a procedure delete_bt_categoria
def deletar_bt_telefone(id_telefone):
    try:
        with conexao.cursor() as cursor:
            # Cria um array para armazenar o valor de nr_telefone
            nr_telefone = cursor.var(oracledb.STRING)
            cursor.callproc("delete_bt_telefone", [id_telefone, nr_telefone])
            print(f"Telefone deletado: {id_telefone} - {nr_telefone.getvalue()}")
    except oracledb.Error as e:
        print(f"Erro ao deletar telefone {id_telefone}: {e}")

def inserir_bt_produto(id_produto, nm_produto, desc_produto, vl_produto, dt_fab_produto, dt_validade, dt_cadastro, img_produto):
    try:
        with conexao.cursor() as cursor:
            cursor.callproc("insert_bt_produto", [
                id_produto, nm_produto, desc_produto, vl_produto,
                dt_fab_produto, dt_validade, dt_cadastro, img_produto
            ])
            print(f"Produto Inserido: {id_produto}, {nm_produto}")
    except oracledb.Error as e:
        print(f"Erro ao inserir produto {id_produto}: {e}")

```

```

+ Código + Texto
def atualizar_bt_produto(id_produto, nm_produto, desc_produto, vl_produto, dt_fab_produto, dt_validade, dt_cadastro, img_produto):
    try:
        with conexao.cursor() as cursor:
            cursor.callproc("update_bt_produto", [
                id_produto, nm_produto, desc_produto, vl_produto,
                dt_fab_produto, dt_validade, dt_cadastro, img_produto
            ])
            print(f"Produto Atualizado: {id_produto}, {nm_produto}")
        except oracledb.Error as e:
            print(f"Erro ao atualizar produto {id_produto}: {e}")

# Função para chamar a procedure delete_bt_categoria
def deletar_bt_produto(id_produto):
    try:
        with conexao.cursor() as cursor:
            # Cria um array para armazenar o valor de nm_produto
            nm_produto = cursor.var(oracledb.STRING)
            cursor.callproc("delete_bt_produto", [id_produto, nm_produto])
            print(f"Produto deletado: {id_produto} - {nm_produto.getvalue()}")
        except oracledb.Error as e:
            print(f"Erro ao deletar produto {id_produto}: {e}")
    try:

# Inserindo dados
insserir_bt_telefone(6, 55, 11, 999999999)
insserir_bt_telefone(7, 1, 14, 123456789)
insserir_bt_telefone(8, 44, 20, 987654321)

# Atualizando dados
atualizar_bt_telefone(1, 55, 11, 888888888)
atualizar_bt_telefone(2, 1, 17, 987654321)
atualizar_bt_telefone(3, 55, 19, 973661158)

deletar_bt_telefone(6)
deletar_bt_telefone(7)
deletar_bt_telefone(8)

deletar_bt_telefone(8)
# Inserindo dados
insserir_bt_produto(6, "Shampoo L'Roche Possay", "Shampoo de altíssima qualidade", 85.0, datetime(2023, 1, 18), datetime(2025, 1, 18), datetime(2024, 5, 18),
insserir_bt_produto(7, "Condicionador Amend", "Condicionador de alta qualidade ", 50.0, datetime(2023, 2, 1), datetime(2026, 8, 1), datetime(2024, 5, 18), "A
insserir_bt_produto(8, "Esmalte vermelho risque", "Esmalte na cor vermelha da marca risque", 12.0, datetime(2023, 9, 1), datetime(2024, 9, 11), datetime(2024
insserir_bt_produto(9, "Hidratante Corporal Monange ", "Hidratante Monange com qualidade duvidosa", 20.0, datetime(2023, 4, 18), datetime(2030, 4, 18), datet
insserir_bt_produto(10, "Protetor solar L'oreal Paris", "Protetor solar de altíssima qualidade", 80.0, datetime(2023, 7, 2), datetime(2025, 7, 2), datetime(202

# Atualizando dados
atualizar_bt_produto(1, "Shampoo Dove Intensivo", "Shampoo Dove para cabelos danificados", 17.0, datetime(2023, 8, 16), datetime(2025, 8, 26), datetime(2024
atualizar_bt_produto(2, "Condicionador SalonLine Intensivo", "Condicionador SalonLine para cabelos danificados", 22.0, datetime(2023, 2, 11), datetime(2025
atualizar_bt_produto(3, "Creme Neutrogena Nutritivo", "Creme nutritivo Neutrogena para a pele", 28.0, datetime(2023, 11, 8), datetime(2025, 11, 8), datetime
atualizar_bt_produto(4, "Sabonete Asepxia Esfoliante", "Sabonete Asepxia com micropartículas esfoliantes", 20.0, datetime(2024, 4, 15), datetime(2024, 10, 1
atualizar_bt_produto(5, "Desodorante Above Fresh", "Desodorante Above em spray com fragrância fresca", 15.0, datetime(2023, 7, 8), datetime(2028, 8, 7), dat

# Deletando produtos
deletar_bt_produto(6)
deletar_bt_produto(7)
deletar_bt_produto(8)
deletar_bt_produto(9)
deletar_bt_produto(10)

finally:
    # Fechando a conexão
    conexao.close()
    print("Conexão fechada.")

```

8- FUNÇÃO VALIDAR CPF

```

470 SET SERVEROUTPUT ON
471 SET VERIFY OFF
472
473 -- Verifica se o CPF tem 11 caracteres e se é composto apenas por números
474 CREATE OR REPLACE FUNCTION validar_cpf (
475     p_cpf IN VARCHAR2
476 ) RETURN BOOLEAN IS
477 BEGIN
478     IF LENGTH(p_cpf) = 11 AND REGEXP_LIKE(p_cpf, '^[0-9]{11}$') THEN
479         RETURN TRUE;
480     ELSE
481         RETURN FALSE;
482     END IF;
483 END validar_cpf;
484
485
486 -- TESTE CORRETO PARA VALIDAR FUNÇÃO CPF
487
488 DECLARE
489     v_cpf_cliente VARCHAR2(11) := '12345678901';
490 BEGIN
491     IF validar_cpf(v_cpf_cliente) THEN
492         DBMS_OUTPUT.PUT_LINE('CPF válido!');
493     ELSE
494         DBMS_OUTPUT.PUT_LINE('CPF inválido!');
495     END IF;
496 END;
497
498
499 -- TESTE COM ERRO PARA VALIDAR FUNÇÃO CPF
500
501 DECLARE
502     v_cpf_cliente VARCHAR2(11) := '1234X6Y8901';
503 BEGIN
504     IF validar_cpf(v_cpf_cliente) THEN
505         DBMS_OUTPUT.PUT_LINE('CPF válido!');
506     ELSE
507         DBMS_OUTPUT.PUT_LINE('CPF inválido!');
508     END IF;
509 END;

```

8.1 TESTE CORRETO FUNÇÃO VALIDAR CPF

```

DECLARE
    v_cpf_cliente VARCHAR2(11) := '12345678901';
BEGIN
    IF validar_cpf(v_cpf_cliente) THEN
        DBMS_OUTPUT.PUT_LINE('CPF válido!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('CPF inválido!');
    END IF;
END;

```

Saída do Script x | Tarefa concluída em 0,053 segundos

CPF válido!

Procedimento PL/SQL concluído com sucesso.

8.2 TESTE COM ERROS FUNÇÃO VALIDAR CPF

```

-- TESTE COM ERRO PARA VALIDAR FUNÇÃO CPF
DECLARE
    v_cpf_cliente VARCHAR2(11) := '1234X6Y8901';
BEGIN
    IF validar_cpf(v_cpf_cliente) THEN
        DBMS_OUTPUT.PUT_LINE('CPF válido!');
    ELSE
        DBMS_OUTPUT.PUT_LINE('CPF inválido!');
    END IF;
END;

```

Saída do Script x | Tarefa concluída em 0,046 segundos

Procedimento PL/SQL concluído com sucesso.

CPF inválido!

Procedimento PL/SQL concluído com sucesso.

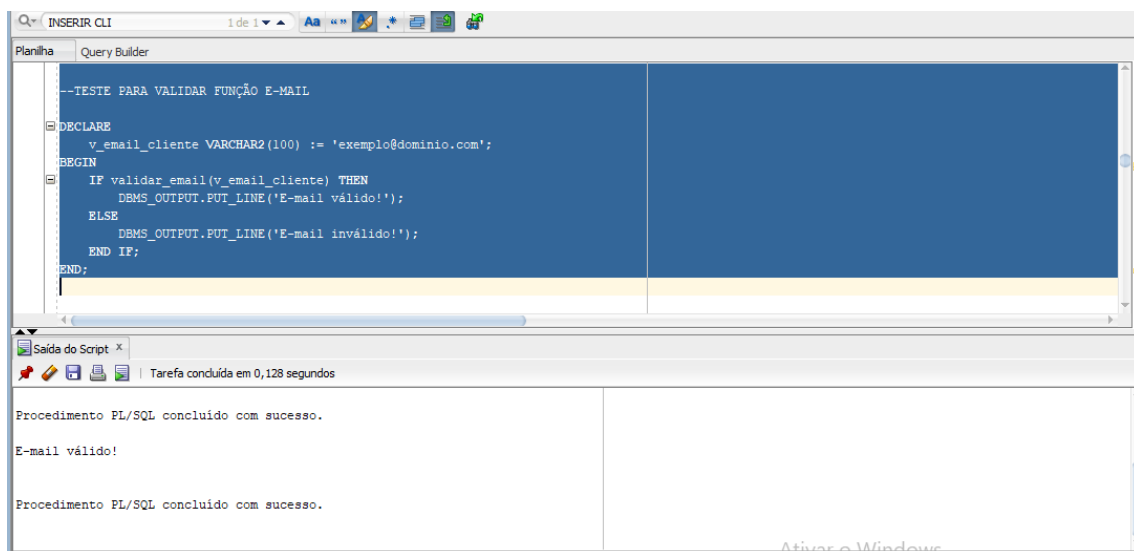
9 FUNÇÃO VALIDAR EMAIL

```

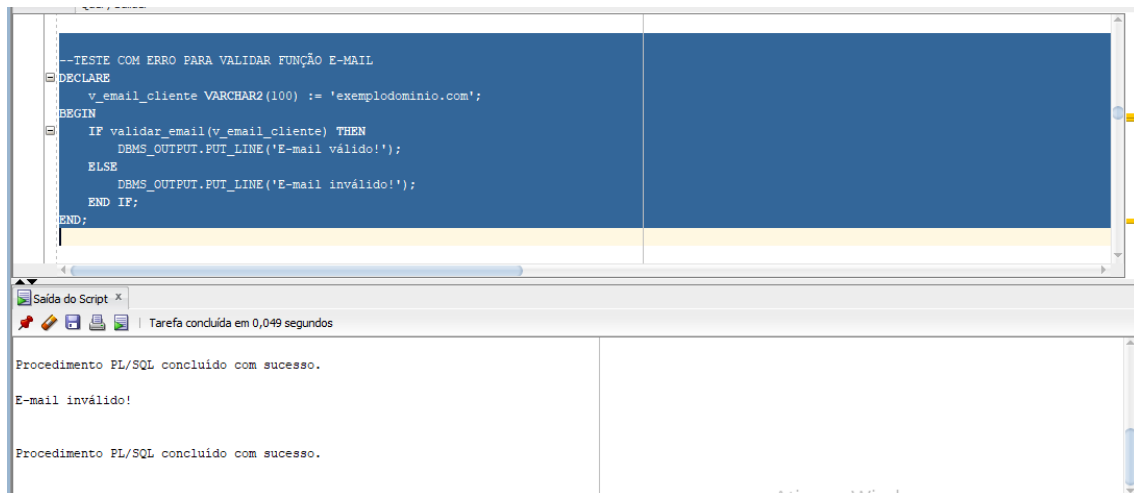
511
512
513 -- Verifica se o e-mail está todo em minúsculas, contém '@' e termina com '.com'
514 CREATE OR REPLACE FUNCTION validar_email (
515     p_email IN VARCHAR2
516 ) RETURN BOOLEAN IS
517 BEGIN
518
519     IF LOWER(p_email) = p_email
520       AND INSTR(p_email, '@') > 1
521       AND SUBSTR(p_email, -4) = '.com' THEN
522         RETURN TRUE;
523     ELSE
524         RETURN FALSE;
525     END IF;
526 END validar_email;
527
528
529 --TESTE PARA VALIDAR FUNÇÃO E-MAIL
530
531 DECLARE
532     v_email_cliente VARCHAR2(100) := 'exemplo@dominio.com';
533 BEGIN
534     IF validar_email(v_email_cliente) THEN
535         DBMS_OUTPUT.PUT_LINE('E-mail válido!');
536     ELSE
537         DBMS_OUTPUT.PUT_LINE('E-mail inválido!');
538     END IF;
539 END;
540
541
542 --TESTE COM ERRO PARA VALIDAR FUNÇÃO E-MAIL
543 DECLARE
544     v_email_cliente VARCHAR2(100) := 'exemplodominio.com';
545 BEGIN
546     IF validar_email(v_email_cliente) THEN
547         DBMS_OUTPUT.PUT_LINE('E-mail válido!');
548     ELSE
549         DBMS_OUTPUT.PUT_LINE('E-mail inválido!');
550     END IF;
551 END;
552
553

```

9.1 TESTE CORRETO FUNÇÃO VALIDAR E-MAIL



9.2 TESTE COM ERROS FUNÇÃO VALIDAR E-MAIL



```
--TESTE COM ERRO PARA VALIDAR FUNÇÃO E-MAIL
DECLARE
  v_email_cliente VARCHAR2(100) := 'exemplodominio.com';
BEGIN
  IF validar_email(v_email_cliente) THEN
    DBMS_OUTPUT.PUT_LINE('E-mail válido!');
  ELSE
    DBMS_OUTPUT.PUT_LINE('E-mail inválido!');
  END IF;
END;
```

Salida do Script x | Tarefa concluída em 0,049 segundos

Procedimento PL/SQL concluído com sucesso.

E-mail inválido!

Procedimento PL/SQL concluído com sucesso.

10 FUNÇÃO VERIFICAR MAIORIDADE

```
--CRIANDO FUNÇÃO QUE VERIFICA SE O CLIENTE É MAIOR DE IDADE

CREATE OR REPLACE FUNCTION verificar_maioridade(
  p_data_nascimento DATE
) RETURN VARCHAR2 IS
  v_idade NUMBER;
BEGIN
  -- Calcula a idade com base na data atual e na data de nascimento
  v_idade := TRUNC(MONTHS_BETWEEN(SYSDATE, p_data_nascimento) / 12);

  -- Verifica se a idade é 18 ou maior
  IF v_idade >= 18 THEN
    RETURN 'Maior de idade';
  ELSE
    RETURN 'Menor de idade';
  END IF;
END;
/

-- TESTE PARA FUNÇÃO VERIFICAR MAIORIDADE
SELECT NM_CLIENTE,
  DT_NASCIMENTO_CLIENTE,
  verificar_maioridade(TO_DATE(DT_NASCIMENTO_CLIENTE, 'DD/MM/YYYY')) as Maioridade
from BT_CLIENTE;
```


10.1 SELECT TESTANDO FUNÇÃO VERIFICAR MAIORIDADE

```
-- TESTE PARA FUNÇÃO VERIFICAR MAIORIDADE
SELECT NM_CLIENTE,
       DT_NASCIMENTO_CLIENTE,
       verificar_maioridade(TO_DATE(DT_NASCIMENTO_CLIENTE, 'DD/MM/YYYY')) as Maioridade
from BT_CLIENTE;
```

Resultado da Consulta x Resultado da Con...

Todas as Linhas Extraídas: 5 em 0,025 segundos

| NM_CLIENTE | DT_NASCIMENTO_CLIENTE | MAIORIDADE |
|-----------------------------------|-----------------------|----------------|
| 1 Vitor Machado Mirada | 12/08/80 | Maior de idade |
| 2 Adriana Rocha Machado | 18/09/76 | Maior de idade |
| 3 Beatriz Aparecida Leite e Souza | 25/02/90 | Maior de idade |
| 4 José Eduardo Machado Cardoso | 30/05/85 | Maior de idade |
| 5 Severino Sebastião de Miranda | 03/12/70 | Maior de idade |

11 FUNÇÃO TRANSFORMANDO EM JSON

```
1209
1209 -- CRIANDO FUNÇÃO QUE CONVERTE RESULTADO PRA JSON PARA USA NOS PROCEDURE A SEGUIR
1210 CREATE OR REPLACE FUNCTION transformando_em_json(p_id_cliente NUMBER)
1211 RETURN CLOB
1212 IS
1213     v_json CLOB := '{}';
1214     v_comma CHAR(1) := ',';
1215     v_data_found BOOLEAN := FALSE; -- Variável para indicar se os dados foram encontrados
1216 BEGIN
1217     -- Obter os dados do cliente
1218     FOR rec IN (
1219         SELECT id_cliente, nm_cliente, email_cliente
1220         FROM bt_cliente
1221         WHERE id_cliente = p_id_cliente
1222     )
1223     LOOP
1224         v_data_found := TRUE; -- Indica que os dados foram encontrados
1225         v_json := v_json || v_comma || '{"id_cliente": ' || rec.id_cliente || ', ' ||
1226         v_json || '"nm_cliente": ' || rec.nm_cliente || ', ' ||
1227         v_json || '"email_cliente": ' || rec.email_cliente || '}' ||
1228         v_comma := ','; -- Atualiza o separador
1229     END LOOP;
1230
1231     -- Se nenhum dado foi encontrado, levanta a exceção
1232     IF NOT v_data_found THEN
1233         RAISE NO_DATA_FOUND;
1234     END IF;
1235
1236     -- Finaliza o JSON
1237     v_json := v_json || '}';
1238
1239     RETURN v_json;
1240
1241 EXCEPTION
1242     -- Tratamento de erro quando nenhum dado é encontrado
1243     WHEN NO_DATA_FOUND THEN
1244         RAISE_APPLICATION_ERROR(-20001, 'Nenhum dado encontrado para o id_cliente: ' || p_id_cliente);
1245
1246     -- Tratamento de erro para valores inválidos
1247     WHEN VALUE_ERROR THEN
1248         RAISE_APPLICATION_ERROR(-20002, 'Erro de valor encontrado ao processar o id_cliente: ' || p_id_cliente);
1249
1250     -- Tratamento para outros erros inesperados
1251     WHEN OTHERS THEN
1252         RAISE_APPLICATION_ERROR(-20003, 'Erro inesperado ao processar o id_cliente: ' || p_id_cliente || '. Detalhes: ' || SQLERRM);
1253 END;
1254 /
```

12 PROCEDURE GERAR JSON CLIENTE

```
--CRIANDO PROCEDURE QUE GERA O RESULTADO EM JSON

CREATE OR REPLACE PROCEDURE gerar_json_cliente(p_id_cliente NUMBER)
IS
    v_json CLOB;
BEGIN
    BEGIN
        SELECT transformando_em_json(c.id_cliente)
        INTO v_json
        FROM bt_cliente c
        JOIN bt_endereco_cliente ec ON c.id_cliente = ec.id_cliente
        JOIN bt_endereco e ON ec.id_endereco = e.id_endereco
        WHERE c.id_cliente = p_id_cliente;

        DBMS_OUTPUT.PUT_LINE(v_json);
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('{"error": "Nenhum dado encontrado."}');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('{"error": "Erro inesperado."}');
    END;
END;
```

12.1 TESTE CORRETO PROCEDURE GERAR JSON CLIENTE

```
-- TESTE CORRETO PARA O PROCEDURE GERAR JSON CLIENTE
SET SERVEROUTPUT ON;

BEGIN
    mostrando_dados_clientes(1);
END;
```

Salida do Script X

Tarefa concluída em 0,299 segundos

```
{"id_cliente":1,"nm_cliente":"Vitor Machado Mirada","email_cliente":"vitor@example.com","endereco":{"id_endereco":1,"logradouro":"Av. Paulista","cidade":"São Paulo"}}
```

Procedimento PL/SQL concluído com sucesso.

12.2 TESTE COM ERROS PROCEDURE GERAR JSON CLIENTE

```
-- TESTE COM ERRO PARA A PROCEDURE GERAR JSON CLIENTE

BEGIN
    mostrando_dados_clientes(17);
END;
```

Salida do Script X

Tarefa concluída em 0,077 segundos

Procedimento PL/SQL concluído com sucesso.

```
{"error": "Nenhum dado encontrado."}
```

Procedimento PL/SQL concluído com sucesso.

13 PROCEDURE IMPRIMINDO VALORES

```
CREATE OR REPLACE PROCEDURE imprimindo_valores
IS
    v_current_value VARCHAR2(150);
    v_prev_value VARCHAR2(150);
    v_next_value VARCHAR2(150);
BEGIN
    FOR r IN (
        SELECT
            id_produto,
            nm_produto,
            LAG(nm_produto, 1, 'Vazio') OVER (ORDER BY id_produto) AS prev_value,
            LEAD(nm_produto, 1, 'Vazio') OVER (ORDER BY id_produto) AS next_value
        FROM bt_produto
    )
    LOOP
        v_current_value := r.nm_produto;
        v_prev_value := r.prev_value;
        v_next_value := r.next_value;

        DBMS_OUTPUT.PUT_LINE('Valor atual: ' || v_current_value || ', Linha anterior: ' || v_prev_value || ', Linha seguinte: ' || v_next_value);
    END LOOP;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nenhum dado encontrado.');
```

13.1 TESTE PROCEDURE IMPRIMINDO VALORES

```
-- TESTE PARA A PROCEDURE IMPRIMINDO VALORES
SET SERVEROUTPUT ON;
BEGIN
    imprimindo_valores;
END;
```

-- APAGANDO TABELA BT_AUDITORIA SOMENTE POR PRECAUÇÃO PRA NÃO TRAVAR O SCRIPT DO PROFESSOR, DESCOMENTAR SE NECESSÁRIO

```
-- DROP TABLE BT_AUDITORIA;
```

Saída do Script x

Tarefa concluída em 0,09 segundos

Valor atual: Condicionador L'Oréal, Linha anterior: Shampoo Palmolive, Linha seguinte: Creme para pentear Amênd

Valor atual: Creme para pentear Amênd, Linha anterior: Condicionador L'Oréal, Linha seguinte: Hidratante Corporal Dove

Valor atual: Hidratante Corporal Dove, Linha anterior: Creme para pentear Amênd, Linha seguinte: Protetor Solar Neutrogena

Valor atual: Protetor Solar Neutrogena, Linha anterior: Hidratante Corporal Dove, Linha seguinte: Vazio

Procedimento PL/SQL concluído com sucesso.

Compilador - Log

Ativar o Windows

Acesse Configurações para ativar o Windows.

14 CRIANDO TABELA AUDITORIA

```
-- CRIANDO UMA TABELA AUDITORIA
CREATE TABLE BT_AUDITORIA (
    id_auditoria NUMBER(10) NOT NULL,
    tabela_nome VARCHAR2(50) NOT NULL,
    operacao_tipo VARCHAR2(10) NOT NULL,
    dados_anteriores CLOB,
    dados_novos CLOB,
    usuario_nome VARCHAR2(50) NOT NULL,
    data_operacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
);

--CRIANDO SEQUENCE PARA O ID AUDITORIA
CREATE SEQUENCE BT_AUDITORIA_SEQ
START WITH 1
INCREMENT BY 1
NOCACHE
NOCYCLE;
```

15 TRIGGER AUDITORIA TABELA CLIENTE

```

CREATE OR REPLACE TRIGGER trg_auditoria_bt_cliente
AFTER INSERT OR UPDATE OR DELETE ON bt_cliente
FOR EACH ROW
DECLARE
    v_dados_anteriores CLOB;
    v_dados_novos CLOB;
BEGIN
    IF INSERTING THEN
        v_dados_novos := '{' ||
            '"id_cliente": ' || :NEW.id_cliente || ', ' ||
            '"id_telefone": ' || :NEW.id_telefone || ', ' ||
            '"id_genero": ' || :NEW.id_genero || ', ' ||
            '"cpf_cliente": ' || :NEW.cpf_cliente || ', ' ||
            '"nm_cliente": ' || :NEW.nm_cliente || ', ' ||
            '"email_cliente": ' || :NEW.email_cliente || ', ' ||
            '"dt_nascimento_cliente": ' || :NEW.dt_nascimento_cliente || ', ' ||
            '"estado_civil_cliente": ' || :NEW.estado_civil_cliente || ', ' ||
            '"dt_cadastro": ' || :NEW.dt_cadastro || ', ' ||
            '"dt_exclusao": ' || :NEW.dt_exclusao || '}' ;
        v_dados_anteriores := NULL;

        INSERT INTO BT_AUDITORIA (
            id_auditoria, tabela_nome, operacao_tipo, dados_anteriores, dados_novos,
            usuario_nome, data_operacao
        ) VALUES (
            BT_AUDITORIA_SEQ.NEXTVAL, 'bt_cliente', 'INSERT', v_dados_anteriores,
            v_dados_novos, SYS_CONTEXT('USERENV', 'SESSION_USER'), SYSDATE
        );
    ELSIF UPDATING THEN

```

```

v_dados_anteriores := '{' ||
    '"id_cliente": ' || :OLD.id_cliente || ', ' ||
    '"id_telefone": ' || :OLD.id_telefone || ', ' ||
    '"id_genero": ' || :OLD.id_genero || ', ' ||
    '"cpf_cliente": ' || :OLD.cpf_cliente || ', ' ||
    '"nm_cliente": ' || :OLD.nm_cliente || ', ' ||
    '"email_cliente": ' || :OLD.email_cliente || ', ' ||
    '"dt_nascimento_cliente": ' || :OLD.dt_nascimento_cliente || ', ' ||
    '"estado_civil_cliente": ' || :OLD.estado_civil_cliente || ', ' ||
    '"dt_cadastro": ' || :OLD.dt_cadastro || ', ' ||
    '"dt_exclusao": ' || :OLD.dt_exclusao || '}' ;

```

```

v_dados_novos := '{' ||
    '"id_cliente": ' || :NEW.id_cliente || ', ' ||
    '"id_telefone": ' || :NEW.id_telefone || ', ' ||
    '"id_genero": ' || :NEW.id_genero || ', ' ||
    '"cpf_cliente": ' || :NEW.cpf_cliente || ', ' ||
    '"nm_cliente": ' || :NEW.nm_cliente || ', ' ||
    '"email_cliente": ' || :NEW.email_cliente || ', ' ||
    '"dt_nascimento_cliente": ' || :NEW.dt_nascimento_cliente || ', ' ||
    '"estado_civil_cliente": ' || :NEW.estado_civil_cliente || ', ' ||
    '"dt_cadastro": ' || :NEW.dt_cadastro || ', ' ||
    '"dt_exclusao": ' || :NEW.dt_exclusao || '}' ;

```

```

INSERT INTO BT_AUDITORIA (
    id_auditoria, tabela_nome, operacao_tipo, dados_anteriores, dados_novos,
    usuario_nome, data_operacao
) VALUES (
    BT_AUDITORIA_SEQ.NEXTVAL, 'bt_cliente', 'UPDATE', v_dados_anteriores,
    v_dados_novos, SYS_CONTEXT('USERENV', 'SESSION_USER'), SYSDATE
);

```

ELSIF DELETING THEN

```

v_dados_antiores := '{' ||
    '"id_cliente": ' || :OLD.id_cliente || ', ' ||
    '"id_telefone": ' || :OLD.id_telefone || ', ' ||
    '"id_genero": ' || :OLD.id_genero || ', ' ||
    '"cpf_cliente": ' || :OLD.cpf_cliente || ', ' ||
    '"nm_cliente": ' || :OLD.nm_cliente || ', ' ||
    '"email_cliente": ' || :OLD.email_cliente || ', ' ||
    '"dt_nascimento_cliente": ' || :OLD.dt_nascimento_cliente || ', ' ||
    '"estado_civil_cliente": ' || :OLD.estado_civil_cliente || ', ' ||
    '"dt_cadastro": ' || :OLD.dt_cadastro || ', ' ||
    '"dt_exclusao": ' || :OLD.dt_exclusao || '}' ;

v_dados_novos := NULL;

```

INSERT INTO BT_AUDITORIA (

id_auditoria, tabela_nome, operacao_tipo, dados_antiores, dados_novos,
usuario_nome, data_operacao

) VALUES (

BT_AUDITORIA_SEQ.NEXTVAL, 'bt_cliente', 'DELETE', v_dados_antiores,
v_dados_novos, SYS_CONTEXT('USERENV', 'SESSION_USER'), SYSDATE

);

END IF;

END;

/

15.1 TESTE TRIGGER COM INSERT

```
-- TESTE DE INSERT CORRETO TRIGGER AUDITORIA BT CLIENTE

INSERT INTO bt_cliente (id_cliente, id_telefone, id_genero, cpf_cliente, nm_cliente, email_cliente, dt_nascimento_cliente, estado_civil_cliente,
VALUES (
12,
1,
1,
'15926935748',
'Ronaldo Souza Miranda',
'ronaldo.souza@example.com',
TO_DATE('2028-12-08', 'YYYY-MM-DD'),
'Solteiro',
SYSDATE);

SELECT * FROM BT_AUDITORIA WHERE operacao_tipo = 'INSERT';
```

SELECT * FROM BT_AUDITORIA WHERE operacao_tipo = 'INSERT';

Todas as Linhas Extraídas: 1 em 0,03 segundos

| ID_AUDITORIA | TABELA_NOME | OPERACAO_TIPO | DADOS_ANTERIORES | DADOS_NOVOS | USUARIO_NOME | DATA_OPERACAO |
|--------------|-------------|---------------|------------------|---|--------------|-----------------------------|
| 1 | bt_cliente | INSERT | (null) | ("id_cliente": "12", "id_telefone": "1", "id_genero": "1", "cpf_cliente"... | RM551451 | 15/09/24 16:33:02,000000000 |

15.2 TESTE TRIGGER COM UPDATE

```
-- TESTE DE UPDATE CORRETO TRIGGER AUDITORIA BT CLIENTE

UPDATE BT_CLIENTE
SET email_cliente = 'vitormiranda4321@outlook.com'
where id_cliente = 1;

SELECT * FROM BT_AUDITORIA WHERE operacao_tipo = 'UPDATE';
```

Todas as Linhas Extraídas: 1 em 0,021 segundos

| ID_AUDITORIA | TABELA_NOME | OPERACAO_TIPO | DADOS_ANTERIORES | DADOS_NOVOS |
|--------------|-------------|---------------|---|--|
| 1 | bt_cliente | UPDATE | ("id_cliente": "1", "id_telefone": "1", "id_genero": "1", "cpf_cliente": "... | ("id_cliente": "1", "id_telefone": "1", "id_genero": "1", "cpf_cli |

15.3 TESTE TRIGGER COM DELETE

```
-- TESTE DE DELETE CORRETO TRIGGER AUDITORIA BT CLIENTE

DELETE FROM BT_CLIENTE WHERE ID_CLIENTE = 12;

SELECT * FROM BT_AUDITORIA WHERE operacao_tipo = 'DELETE';
```

Todas as Linhas Extraídas: 1 em 0,048 segundos

| ID_AUDITORIA | TABELA_NOME | OPERACAO_TIPO | DADOS_ANTERIORES | DADOS_NOVOS | USUARIO_NOME | DATA_OPERACAO |
|--------------|-------------|---------------|--|-------------|--------------|-----------------------------|
| 1 | bt_cliente | DELETE | ("id_cliente": "12", "id_telefone": "1", "id_genero": "1", "cpf_cliente": "... | (null) | RM551451 | 15/09/24 16:34:45,000000000 |