INSTITUTO POLITÉCNICO DE BRAGANÇA
-ENGENHARIA INFORMÁTICA -

Carlos Dinis Borges Sousa - a59443
Francisco Manuel Ribeiro da Silva - a56561
Marina da Fonseca Frias de Siqueira Campos - m321987

Profº: Reis Lima Quarteu

Grupo 3

2º Trabalho Prático - Base de Dados II

Bragança- PT
2025

Relatório

a) Descrição dos requisitos

O objeto do trabalho foi desenvolver funções PL/SQL para manipulação e visualização de dados de Gestão de uma Cadeia de Hotéis. Foram utilizados conceitos de SQL e programação PL/SQL, contendo controle de fluxo, cursores, manipulação de strings e datas, e todos os demais que formaram os requisitos do trabalho como um todo. Foi implementado e testado na plataforma Oracle APEX, onde cada página corresponde a um exercício pedido.

b) Modelo de dados

O modelo utilizado foi o mesmo do Trabalho Prático 1, composto por tabelas principais como HOTEIS, CLIENTES, FUNCIONARIOS, QUARTOS, FORNECEDORES, SERVIÇOS, como também as tabelas de relacionamento como RESERVAS, PAGAMENTOS, AVALIACOES, MANUTENCOES e outras. O modelo seguiu uma estrutura normalizada e as chaves refletem a integridade entre as entidades e suas relações.

c)

1. código:

```
create or replace PROCEDURE estatisticas_avaliacoes_hotel (
   p_id_hotel IN NUMBER
) AS
   v_media   NUMBER;
   v_max     NUMBER;
   v_min     NUMBER;
   v_total   NUMBER;
   v_count   NUMBER;
BEGIN
   SELECT ROUND(AVG(nota), 2),
          MAX(nota),
          MIN(nota),
          SUM(nota),
          COUNT(*)
```

```
    INTO v_media, v_max, v_min, v_total, v_count
    FROM AVALIACOES
    WHERE ID_HOTEL = p_id_hotel
    GROUP BY ID_HOTEL
    HAVING COUNT(*) > 0;
    HTP.P('<b>Estatísticas para o Hotel ID: </b>' || p_id_hotel || '<br>');
    HTP.P('Média das Notas: ' || v_media || '<br>');
    HTP.P('Nota Máxima: ' || v_max || '<br>');
    HTP.P('Nota Mínima: ' || v_min || '<br>');
    HTP.P('Soma das Notas: ' || v_total || '<br>');
    HTP.P('Total de Avaliações: ' || v_count || '<br>');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        HTP.P('Nenhuma avaliação encontrada para o hotel informado (ID: ' || p_id_hotel || ').');
    WHEN OTHERS THEN
        HTP.P('Erro ao calcular estatísticas: ' || SQLERRM);
END;
```

Resultado:

Média das Notas:

Nota Máxima:

Nota Mínima:

Soma das Notas:

Total de Avaliações:

2.1 código:

```
create or replace PROCEDURE clientes_com_avaliacoes (
    p_nome_cliente IN VARCHAR2
) IS
    v_count NUMBER := 0;
BEGIN
    FOR rec IN (
        SELECT c.id_cliente,
               c.nome,
               c.email,
               c.telefone,
               NVL(a.nota, -1) AS nota
        FROM clientes c
        LEFT JOIN avaliacoes a ON c.id_cliente = a.id_cliente
        WHERE LOWER(c.nome) LIKE LOWER('%' || p_nome_cliente || '%')
        ORDER BY c.id_cliente
    ) LOOP
        v_count := v_count + 1;
        HTP.P('<b>Cliente ID:</b> ' || rec.id_cliente || '<br>');
        HTP.P('<b>Nome:</b> ' || rec.nome || '<br>');
        HTP.P('<b>Email:</b> ' || rec.email || '<br>');
        HTP.P('<b>Telefone:</b> ' || rec.telefone || '<br>');
```

```
      HTP.P('<b>Nota:</b> ' ||
          CASE WHEN rec.nota = -1 THEN 'sem avaliação' ELSE TO_CHAR(rec.nota) END || '<br><hr>');
   END LOOP;

   IF v_count = 0 THEN
      HTP.P('Nenhum cliente encontrado com o nome parecido com "' || p_nome_cliente || '".');
   END IF;

EXCEPTION
   WHEN OTHERS THEN
      HTP.P('Erro ao buscar clientes: ' || SQLERRM);
END;
```

Resultado:

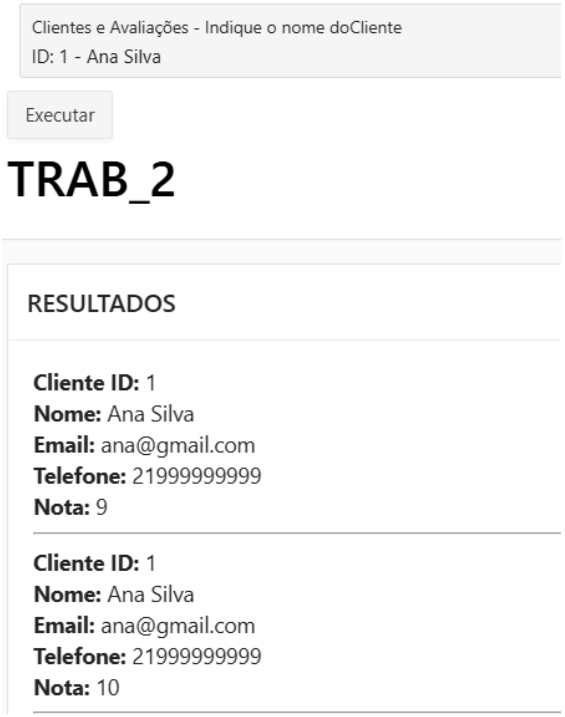
Cliente ID:

Nome:

Email:

Telefone:

Nota:



2.2 código:

```
create or replace PROCEDURE hoteis_com_avaliacoes (
   p_nome_hotel IN VARCHAR2
) IS
   v_count NUMBER := 0;
BEGIN
   FOR rec IN (
      SELECT h.id_hotel,
           h.nome,
           h.cidade,
           h.pais,
           NVL(a.nota, -1) AS nota
      FROM hoteis h
      RIGHT JOIN avaliacoes a ON h.id_hotel = a.id_hotel
      WHERE LOWER(h.nome) LIKE LOWER('%' || p_nome_hotel || '%')
      ORDER BY h.id_hotel
   ) LOOP
      v_count := v_count + 1;
      HTP.P('<b>Hotel ID:</b> ' || rec.id_hotel || '<br>');
      HTP.P('<b>Nome:</b> ' || rec.nome || '<br>');
      HTP.P('<b>Cidade:</b> ' || rec.cidade || '<br>');
      HTP.P('<b>País:</b> ' || rec.pais || '<br>');
      HTP.P('<b>Nota:</b> ' ||
          CASE WHEN rec.nota = -1 THEN 'sem avaliação' ELSE TO_CHAR(rec.nota) END || '<br><hr>');
   END LOOP;
   IF v_count = 0 THEN
      HTP.P('Nenhum hotel encontrado com o nome parecido com "' || p_nome_hotel || '".');
   END IF;
EXCEPTION
   WHEN OTHERS THEN
      HTP.P('Erro ao buscar hotéis: ' || SQLERRM);
END;
```

Resultado:

Hotel ID:

Nome:

Cidade:

País:

Nota:

3.1 código:

```
create or replace PROCEDURE quarto_mais_caro_do_hotel (
    p_id_hotel IN NUMBER
) IS
    v_preco_max quartos.preco%TYPE;
    v_quarto quartos%ROWTYPE;
BEGIN
    SELECT MAX(preco)
    INTO v_preco_max
    FROM quartos
```

```
    WHERE id_hotel = p_id_hotel;
    SELECT *
    INTO v_quarto
    FROM quartos
    WHERE id_hotel = p_id_hotel
      AND preco = v_preco_max
      AND ROWNUM = 1;
    -- Exibir no APEX
    HTP.P('<b>Hotel ID:</b> ' || p_id_hotel || '<br>');
    HTP.P('<b>Quarto ID:</b> ' || v_quarto.id_quarto || '<br>');
    HTP.P('<b>Número:</b> ' || v_quarto.numero || '<br>');
    HTP.P('<b>Tipo:</b> ' || v_quarto.tipo || '<br>');
    HTP.P('<b>Preço:</b> ' || v_quarto.preco || '<br>');
    HTP.P('<b>Status:</b> ' || v_quarto.status || '<br>');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        HTP.P('Nenhum quarto encontrado para o hotel informado (ID: ' || p_id_hotel || ').');
    WHEN OTHERS THEN
        HTP.P('Erro ao buscar dados do quarto mais caro: ' || SQLERRM);
END;
```

Resultado:

Hotel ID:

Quarto ID:

Número:

Tipo:

Preço:

Status:

3.2 código:

```
create or replace PROCEDURE clientes_dos_quartos_caros (
   p_id_hotel IN NUMBER
) IS
   v_count NUMBER := 0;
BEGIN
   FOR rec IN (
      SELECT DISTINCT c.id_cliente, c.nome, c.email
      FROM clientes c
      JOIN reservas r ON c.id_cliente = r.id_cliente
      JOIN quartos q ON r.id_quarto = q.id_quarto
      WHERE q.id_hotel = p_id_hotel
        AND q.preco IN (
          SELECT preco
          FROM quartos
          WHERE id_hotel = p_id_hotel
            AND preco = (SELECT MAX(preco)
                         FROM quartos
                         WHERE id_hotel = p_id_hotel)
        )
   ) LOOP
      v_count := v_count + 1;
```

```
        HTP.P('<b>Cliente ID:</b> ' || rec.id_cliente || '<br>');
        HTP.P('<b>Nome:</b> ' || rec.nome || '<br>');
        HTP.P('<b>Email:</b> ' || rec.email || '<br><hr>');
    END LOOP;
    IF v_count = 0 THEN
        HTP.P('Nenhum cliente encontrado com reservas nos quartos mais caros do hotel ID: ' || p_id_hotel);
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        HTP.P('Erro ao buscar clientes: ' || SQLERRM);
END;
```

Resultado:

Cliente ID:

Nome:

Email:

4.1 código:

```
create or replace PROCEDURE verificar_salario_funcionario (
    p_id_funcionario IN NUMBER
) IS
    v_func funcionarios%ROWTYPE;
```

```
    v_media_salario funcionarios.salario%TYPE;
BEGIN
    SELECT *
    INTO v_func
    FROM funcionarios
    WHERE id_funcionario = p_id_funcionario;
    SELECT AVG(salario)
    INTO v_media_salario
    FROM funcionarios
    WHERE id_hotel = v_func.id_hotel;
    HTP.P('<b>Funcionário:</b> ' || v_func.nome || '<br>');
    HTP.P('<b>Salário:</b> ' || v_func.salario || '<br>');
    HTP.P('<b>Média do Hotel:</b> ' || ROUND(v_media_salario, 2) || '<br>');
    IF v_func.salario > v_media_salario THEN
        HTP.P('Este funcionário ganha <b>acima da média</b> do seu hotel.<br>');
    ELSIF v_func.salario < v_media_salario THEN
        HTP.P('Este funcionário ganha <b>abaixo da média</b> do seu hotel.<br>');
    ELSE
        HTP.P('Este funcionário ganha <b>exatamente a média</b> do seu hotel.<br>');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        HTP.P('Funcionário com ID ' || p_id_funcionario || ' não encontrado.');
    WHEN OTHERS THEN
        HTP.P('Erro ao processar dados do funcionário: ' || SQLERRM);
END;
```

Resultado:

Funcionário:

Salário:

Média do Hotel:

Este funcionário ganha exatamente a média do seu hotel.

4.2 código:

```
create or replace PROCEDURE status_quarto_mensagem (
    p_id_quarto IN NUMBER
) IS
    v_quarto quartos%ROWTYPE;
    v_hotel_nome hoteis.nome%TYPE;
    v_mensagem VARCHAR2(100);
BEGIN

    SELECT * INTO v_quarto
    FROM quartos
    WHERE id_quarto = p_id_quarto;



    SELECT nome INTO v_hotel_nome
    FROM hoteis
    WHERE id_hotel = v_quarto.id_hotel;

    v_mensagem := CASE LOWER(v_quarto.status)
                    WHEN 'disponível' THEN 'O quarto está disponível para reserva.'
                    WHEN 'ocupado' THEN 'O quarto está atualmente ocupado.'
                    WHEN 'manutenção' THEN 'O quarto está em manutenção.'
                    ELSE 'Status desconhecido.'
                  END;
```

```
    HTP.P('<b>Hotel:</b> ' || v_hotel_nome || '<br>');
    HTP.P('<b>Quarto ID:</b> ' || v_quarto.id_quarto || '<br>');
    HTP.P('<b>Número:</b> ' || v_quarto.numero || '<br>');
    HTP.P('<b>Tipo:</b> ' || v_quarto.tipo || '<br>');
    HTP.P('<b>Status:</b> ' || v_quarto.status || '<br>');
    HTP.P('<b>Mensagem:</b> ' || v_mensagem || '<br>');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        HTP.P('Quarto com ID ' || p_id_quarto || ' não encontrado.');
    WHEN OTHERS THEN
        HTP.P('Erro ao processar status do quarto: ' || SQLERRM);
END;
```

Resultado:

Reserva Nº

Cliente:

Entrada:

Saída:

Status:

5.1 código:

```
create or replace PROCEDURE listar_pagamentos_por_reserva (
   p_id_reserva IN NUMBER
) IS
BEGIN
   FOR rec IN (
      SELECT p.id_pagamento,
           p.valor_pago,
           p.metodo_pagamento,
           r.data_entrada,
           r.data_saida
      FROM pagamentos p
      JOIN reservas r ON p.id_reserva = r.id_reserva
      WHERE p.id_reserva = p_id_reserva
      ORDER BY p.id_pagamento
   ) LOOP
      HTP.P('<b>Pagamento ID:</b> ' || rec.id_pagamento || '<br>');
      HTP.P('<b>Valor Pago:</b> R$ ' || rec.valor_pago || '<br>');
      HTP.P('<b>Método:</b> ' || rec.metodo_pagamento || '<br>');
      HTP.P('<b>Período:</b> ' || TO_CHAR(rec.data_entrada, 'DD/MM/YYYY') ||
           ' a ' || TO_CHAR(rec.data_saida, 'DD/MM/YYYY') || '<br><hr>');
   END LOOP;
EXCEPTION
   WHEN OTHERS THEN
      HTP.P('Erro ao listar pagamentos: ' || SQLERRM);
END;
```

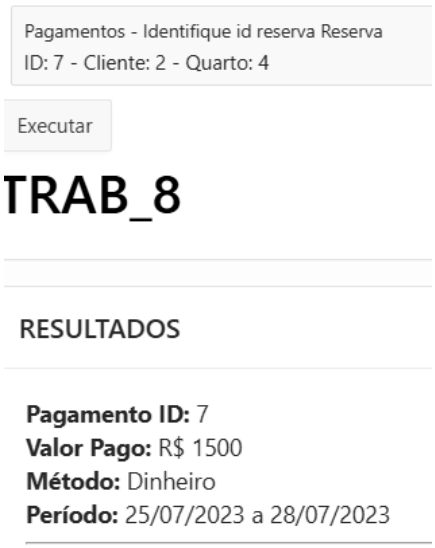
Resultado:

Pagamento ID:

Valor Pago: R$

Método:

Período:



5.2 código:

```
create or replace PROCEDURE listar_manutencoes_por_hotel (
    p_id_hotel IN NUMBER
) IS
    CURSOR c_manut IS
        SELECT m.*, q.numero AS numero_quarto
        FROM manutencoes m
        JOIN quartos q ON m.id_quarto = q.id_quarto
        WHERE q.id_hotel = p_id_hotel
        ORDER BY m.data_manutencao;
    v_manut  c_manut%ROWTYPE;
    v_index  NUMBER := 0;
BEGIN
    OPEN c_manut;
    LOOP
        FETCH c_manut INTO v_manut;
        EXIT WHEN c_manut%NOTFOUND;
        v_index := v_index + 1;
        HTP.P('<b>Manutenção #' || v_index || '</b><br>');
        HTP.P('Quarto Nº: ' || v_manut.numero_quarto || '<br>');
        HTP.P('Data: ' || TO_CHAR(v_manut.data_manutencao, 'DD/MM/YYYY') || '<br>');
        HTP.P('Descrição: ' || v_manut.descricao || '<br>');
        HTP.P('Status: ' || v_manut.status || '<br><hr>');
    END LOOP;
```

```
    CLOSE c_manut;
EXCEPTION
    WHEN OTHERS THEN
        HTP.P('Erro ao listar manutenções: ' || SQLERRM);
END;
```

Resultado:

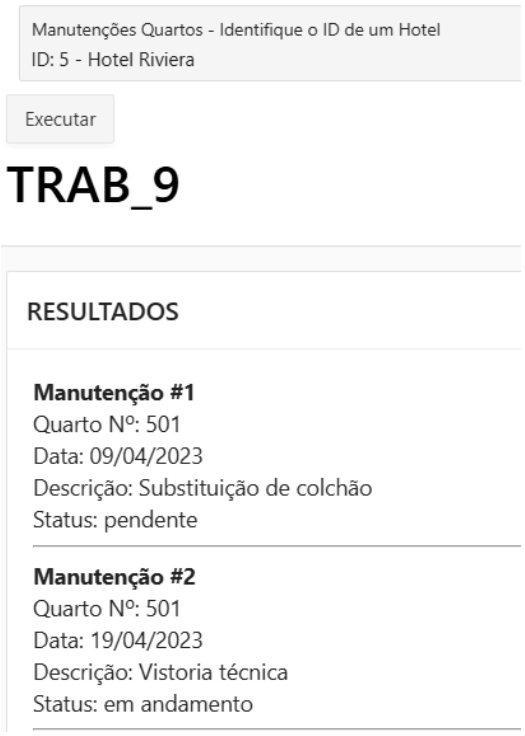
Manutenção #

Quarto Nº:

Data:

Descrição:

Status:



6.1 código:

```
create or replace PROCEDURE dados_tempo_funcionario (
    p_id_funcionario IN NUMBER
) IS
    v_data_contratacao funcionarios.data_contratacao%TYPE;
    v_nome VARCHAR2(100);
```

```
    v_dias_na_empresa NUMBER;
    v_dia_semana VARCHAR2(20);
    v_hora_contratacao VARCHAR2(10);
    v_horas_desde_contratacao NUMBER;
BEGIN
    SELECT nome, data_contratacao
    INTO v_nome, v_data_contratacao
    FROM funcionarios
    WHERE id_funcionario = p_id_funcionario;

    v_dias_na_empresa := TRUNC(SYSDATE - v_data_contratacao);
    v_dia_semana := TO_CHAR(v_data_contratacao, 'DAY', 'NLS_DATE_LANGUAGE=PORTUGUESE');
    v_hora_contratacao := TO_CHAR(v_data_contratacao, 'HH24:MI');
    v_horas_desde_contratacao := ROUND((SYSDATE - v_data_contratacao) * 24, 1);

    HTP.P('<b>Funcionário:</b> ' || v_nome || '<br>');
    HTP.P('<b>Data de Contratação:</b> ' || TO_CHAR(v_data_contratacao, 'DD/MM/YYYY') || '<br>');
    HTP.P('<b>Hora da contratação:</b> ' || v_hora_contratacao || '<br>');
    HTP.P('<b>Dia da semana:</b> ' || INITCAP(TRIM(v_dia_semana)) || '<br>');
    HTP.P('<b>Dias na empresa:</b> ' || v_dias_na_empresa || ' dias<br>');
    HTP.P('<b>Horas totais desde a contratação:</b> ' || v_horas_desde_contratacao || ' horas<br>');

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        HTP.P('Funcionário com ID ' || p_id_funcionario || ' não encontrado.');
    WHEN OTHERS THEN
        HTP.P('Erro ao obter dados temporais: ' || SQLERRM);
END;
```

Resultado:

Funcionário:

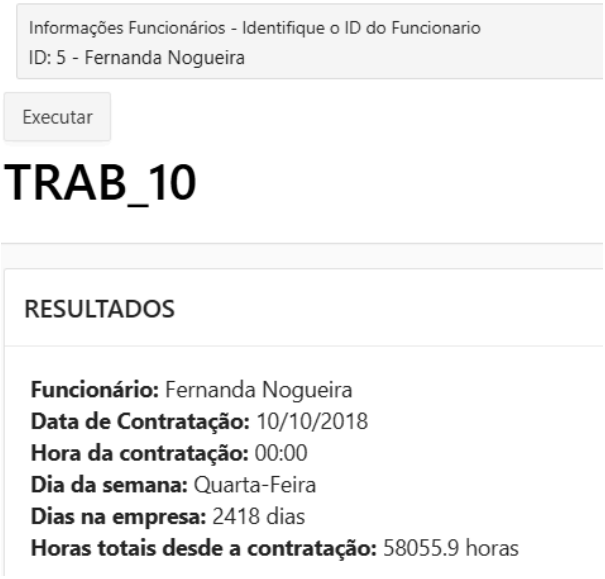
Data de Contratação:

Hora da contratação:

Dia da semana:

Dias na empresa:

Horas totais desde a contratação:



6.2 código:

```
create or replace PROCEDURE formatar_dados_cliente (
    p_id_cliente IN NUMBER
) IS
    v_nome clientes.nome%TYPE;
    v_email clientes.email%TYPE;
    v_nome_maiusculo VARCHAR2(100);
    v_nome_abreviado VARCHAR2(50);
    v_tamanho_email NUMBER;
    v_posicao_arroba NUMBER;
    v_parte_nome VARCHAR2(50);
    v_index NUMBER := 1;
BEGIN
    SELECT nome, email
    INTO v_nome, v_email
    FROM clientes
    WHERE id_cliente = p_id_cliente;

    v_nome_abreviado := '';

    LOOP
        v_parte_nome := REGEXP_SUBSTR(v_nome, '\S+', 1, v_index);
```

```
      EXIT WHEN v_parte_nome IS NULL;

      v_nome_abreviado := v_nome_abreviado || UPPER(SUBSTR(v_parte_nome, 1, 1)) || '. ';
      v_index := v_index + 1;
   END LOOP;

   v_nome_maiusculo := UPPER(v_nome);
   v_tamanho_email := LENGTH(v_email);
   v_posicao_arroba := INSTR(v_email, '@');

   HTP.P('<b>Nome original:</b> ' || v_nome || '<br>');
   HTP.P('<b>Nome em maiúsculas:</b> ' || v_nome_maiusculo || '<br>');
   HTP.P('<b>Iniciais:</b> ' || TRIM(v_nome_abreviado) || '<br>');
   HTP.P('<b>Email:</b> ' || v_email || '<br>');
   HTP.P('<b>Comprimento do e-mail:</b> ' || v_tamanho_email || ' caracteres<br>');
   HTP.P('<b>Posição do "@" no e-mail:</b> ' || v_posicao_arroba || '<br>');

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      HTP.P('Cliente com ID ' || p_id_cliente || ' não encontrado.');
   WHEN OTHERS THEN
      HTP.P('Erro ao formatar dados do cliente: ' || SQLERRM);
END;
```

Resultado:

Nome original:

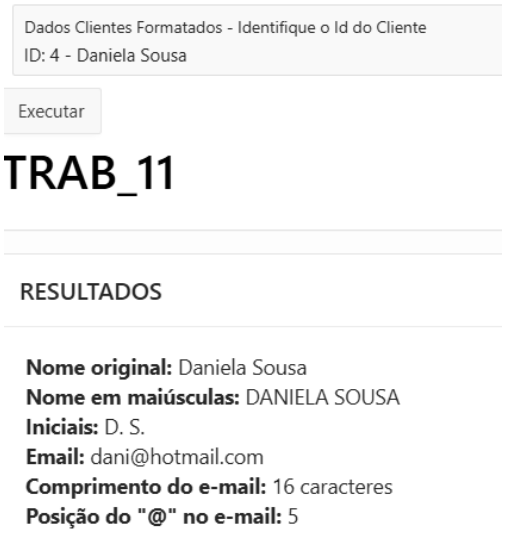
Nome em maiúsculas:

Iniciais:

Email:

Comprimento do e-mail:

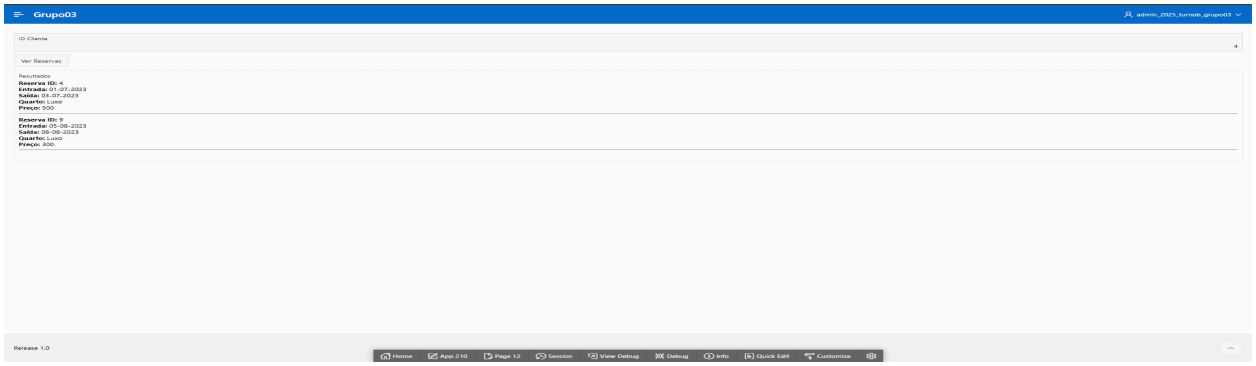
Posição do "@" no e-mail:



7.2

```
CREATE OR REPLACE PROCEDURE p_reservas_cliente (
   p_id_cliente IN RESERVAS.ID_CLIENTE%TYPE,
   p_resultado OUT CLOB
)
IS
   CURSOR c_reservas IS
      SELECT r.id_reserva, r.data_entrada, r.data_saida, q.tipo, q.preco
      FROM reservas r
      JOIN quartos q ON r.id_quarto = q.id_quarto
      WHERE r.id_cliente = p_id_cliente;

   rec_reserva c_reservas%ROWTYPE;
   v_html CLOB;
BEGIN
   -- Inicializar a variável CLOB
   v_html := EMPTY_CLOB();
```

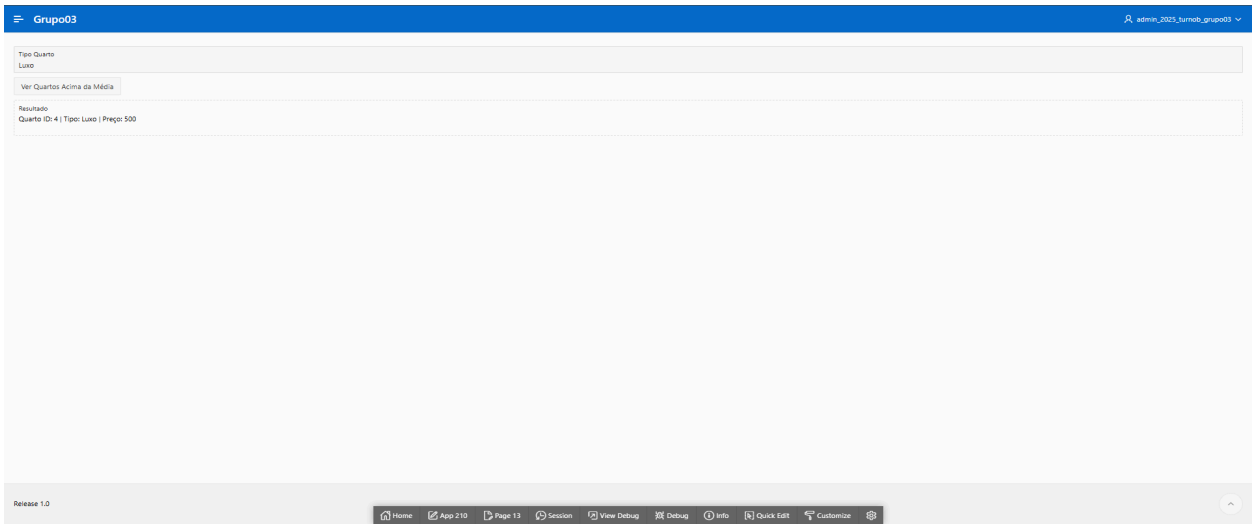
```
   OPEN c_reservas;
   LOOP
      FETCH c_reservas INTO rec_reserva;
      EXIT WHEN c_reservas%NOTFOUND;

      v_html := v_html ||
         '<b>Reserva ID:</b> ' || rec_reserva.id_reserva || '<br>' ||
         '<b>Entrada:</b> ' || TO_CHAR(rec_reserva.data_entrada, 'DD-MM-YYYY') ||
'<br>' ||
         '<b>Saída:</b> ' || TO_CHAR(rec_reserva.data_saida, 'DD-MM-YYYY') ||
'<br>' ||
         '<b>Quarto:</b> ' || rec_reserva.tipo || '<br>' ||
         '<b>Preço:</b> ' || rec_reserva.preco || '<br><hr>';
   END LOOP;
   CLOSE c_reservas;

   p_resultado := v_html;

EXCEPTION
   WHEN OTHERS THEN
      p_resultado := 'Erro: ' || SQLERRM;
END;
```



7.3

```
CREATE OR REPLACE PROCEDURE p_quartos_acima_media (
  p_tipo_quarto IN quartos.tipo%TYPE,
  p_resultado OUT CLOB
) IS
  -- Cursor com subquery: quartos acima da média do mesmo tipo
  CURSOR c_quartos IS
    SELECT q.id_quarto, q.tipo, q.preco
    FROM quartos q
    WHERE q.tipo = p_tipo_quarto
      AND q.preco > (
        SELECT AVG(preco)
        FROM quartos
        WHERE tipo = p_tipo_quarto
      );

  rec_quarto c_quartos%ROWTYPE;
  v_html CLOB := '';
  v_count NUMBER := 0;
BEGIN
  FOR rec_quarto IN c_quartos LOOP
```

```
    v_count := v_count + 1;
    v_html := v_html || 'Quarto ID: ' || rec_quarto.id_quarto ||
              ' | Tipo: ' || rec_quarto.tipo ||
              ' | Preço: ' || rec_quarto.preco || '<br>';
  END LOOP;

  IF v_count = 0 THEN
    p_resultado := 'Nenhum quarto acima da média encontrado.';
  ELSE
    p_resultado := v_html;
  END IF;

EXCEPTION
  WHEN OTHERS THEN
    p_resultado := 'Erro: ' || SQLERRM;
END;
```



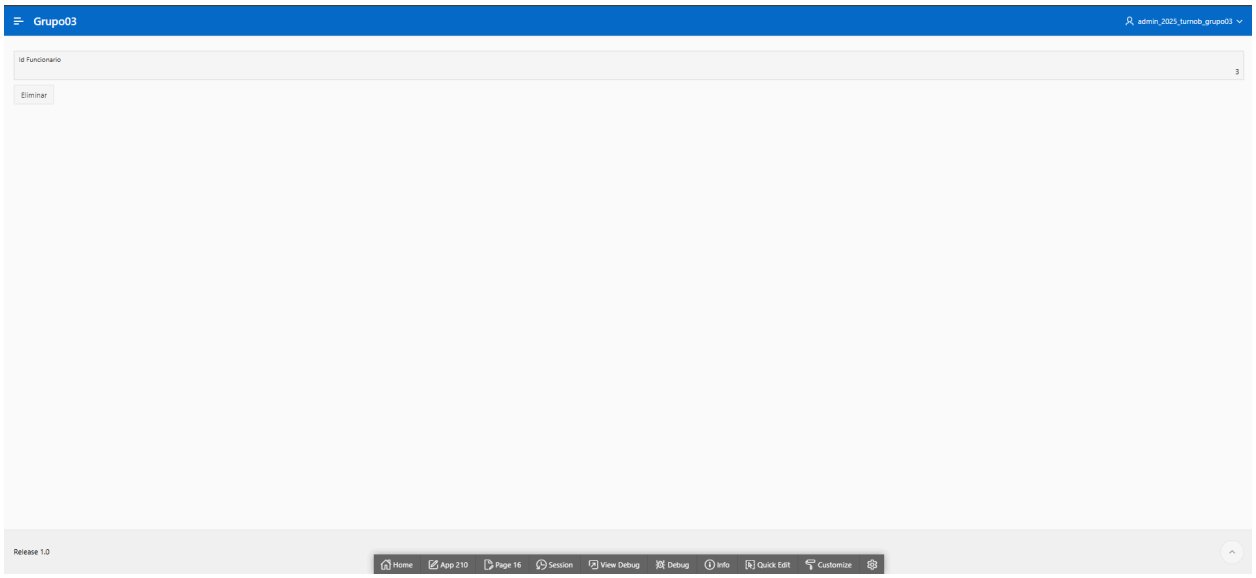
8.1

```
DECLARE
BEGIN
  INSERT INTO funcionarios (
    id_funcionario,
    nome,
    salario,
    id_hotel,
    telefone,
    email,
    cargo,
    data_contratacao
  ) VALUES (
    :P15_ID_FUNCIONARIO,
    :P15_NOME,
    :P15_SALARIO,
    :P15_ID_HOTEL,
    :P15_TELEFONE,
    :P15_EMAIL,
    :P15_CARGO,
    :P15_DATA_CONTRATACAO
  );

  HTP.P('Funcionário inserido com sucesso.');
EXCEPTION
  WHEN OTHERS THEN
    HTP.P('Erro ao inserir: ' || SQLERRM);
END;
```

8.2

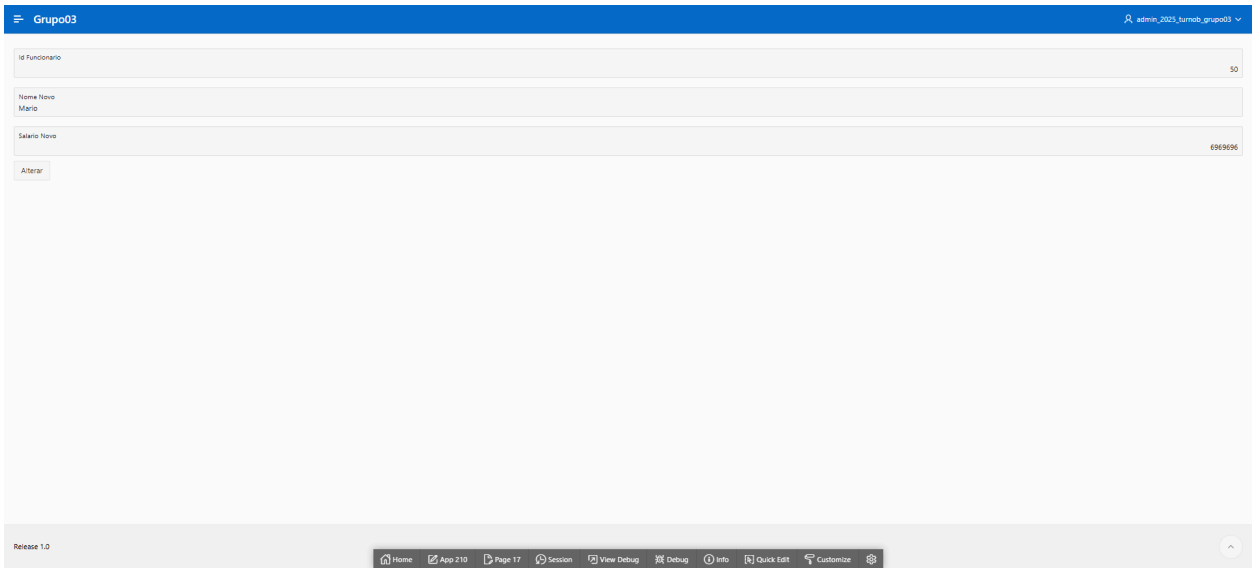
```
DECLARE
  registo_func funcionarios%ROWTYPE;
  v_id funcionarios.id_funcionario%TYPE;
BEGIN
  v_id := :P16_ID_FUNCIONARIO;

  SELECT * INTO registo_func
  FROM funcionarios
  WHERE id_funcionario = v_id;

  DELETE FROM funcionarios
  WHERE id_funcionario = v_id;

  HTP.P('Funcionário: ' || registo_func.id_funcionario || ', ' ||
      registo_func.nome || ' foi eliminado.');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
```

```
    HTP.P('Funcionário com ID ' || v_id || ' não foi encontrado.');
END;
```



8.3

```
DECLARE
BEGIN
  UPDATE funcionarios
  SET nome = :P8_NOME_NOVO,
      salario = :P8_SALARIO_NOVO
  WHERE id_funcionario = :P8_ID_FUNCIONARIO;

  HTP.P('Funcionário foi alterado com sucesso.');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    HTP.P('Funcionário não encontrado.');
END;
```

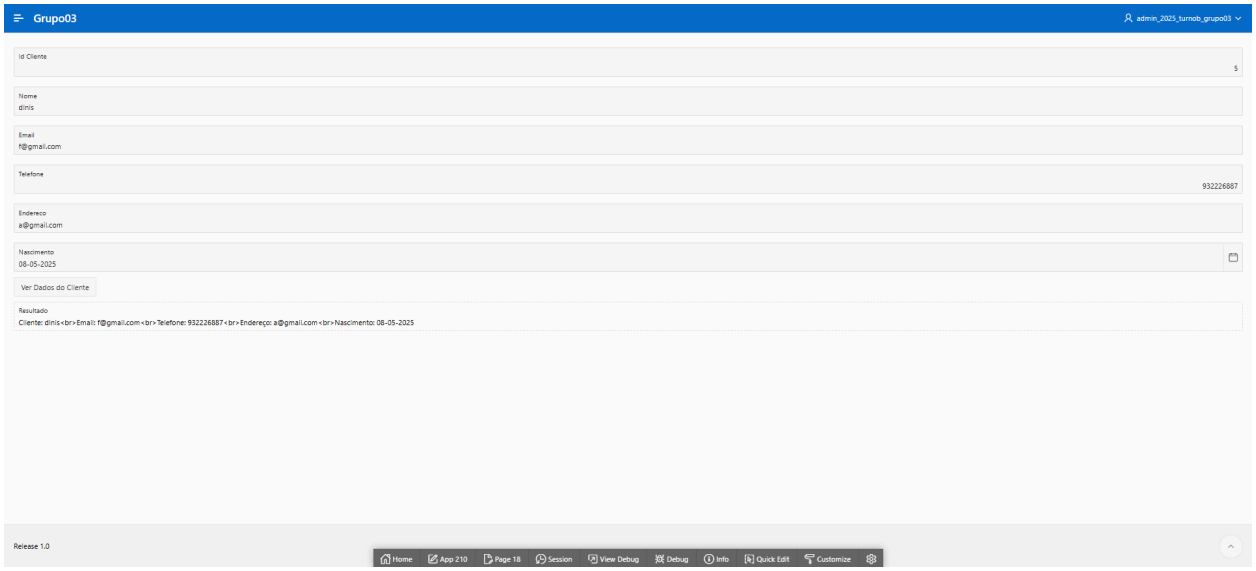


9.1

```
CREATE OR REPLACE PROCEDURE p_usar_record_cliente (
    p_id_cliente IN CLIENTES.ID_CLIENTE%TYPE,
    p_nome       IN CLIENTES.NOME%TYPE,
    p_email      IN CLIENTES.EMAIL%TYPE,
    p_telefone   IN CLIENTES.TELEFONE%TYPE,
    p_endereco   IN CLIENTES.ENDERECO%TYPE,
    p_nascimento IN CLIENTES.DATA_NASCIMENTO%TYPE,
    p_resultado  OUT CLOB
)
AS
  -- Tipo RECORD para endereço
  TYPE endereco_rec IS RECORD (
    rua CLIENTES.ENDERECO%TYPE,
    nascimento CLIENTES.DATA_NASCIMENTO%TYPE
  );
```

```
  -- Tipo RECORD para cliente (com record aninhado)
  TYPE cliente_rec IS RECORD (
    id_cliente CLIENTES.ID_CLIENTE%TYPE,
    nome       CLIENTES.NOME%TYPE,
    email      CLIENTES.EMAIL%TYPE,
    telefone   CLIENTES.TELEFONE%TYPE,
    dados_local endereco_rec
  );

  cliente cliente_rec;

BEGIN
  -- Atribuição dos dados
  cliente.id_cliente := p_id_cliente;
  cliente.nome := p_nome;
  cliente.email := p_email;
  cliente.telefone := p_telefone;
  cliente.dados_local.rua := p_endereco;
  cliente.dados_local.nascimento := p_nascimento;

  -- Exibir os dados formatados
  p_resultado := 'Cliente: ' || cliente.nome || '<br>' ||
                 'Email: ' || cliente.email || '<br>' ||
                 'Telefone: ' || cliente.telefone || '<br>' ||
                 'Endereço: ' || cliente.dados_local.rua || '<br>' ||
                 'Nascimento: ' || cliente.dados_local.nascimento;

EXCEPTION
  WHEN OTHERS THEN
    p_resultado := 'Erro: ' || SQLERRM;
END;
```

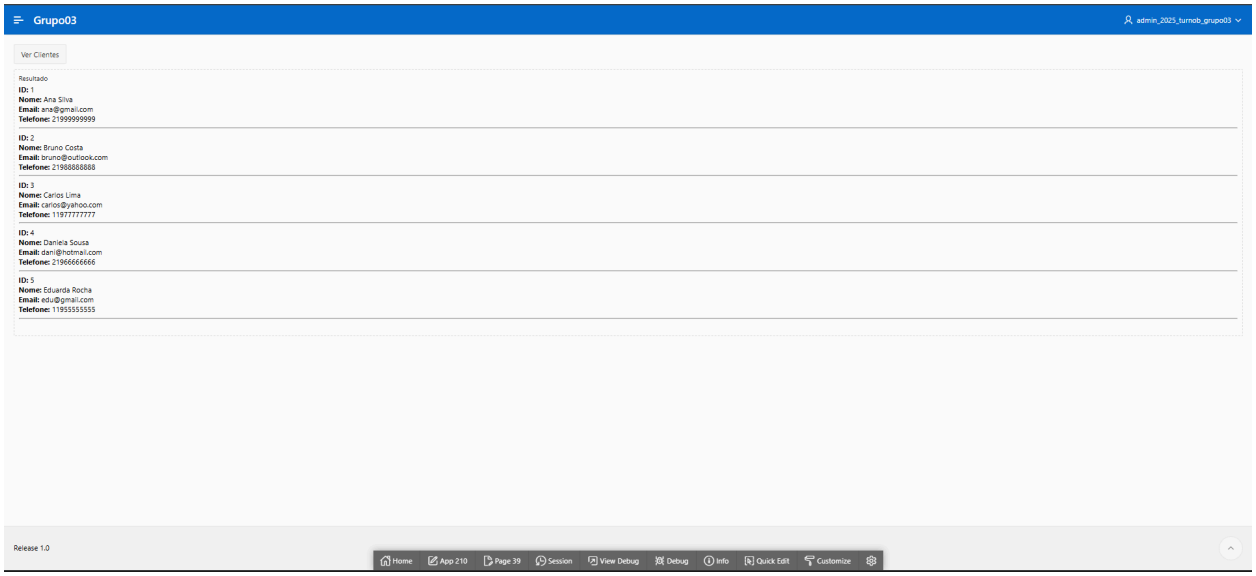
/



9.2

```
CREATE OR REPLACE PROCEDURE p_usar_rec_cursor_formatado (
   p_resultado OUT CLOB
)
IS
   TYPE cliente_rec IS RECORD (
      cli_id   CLIENTES.ID_CLIENTE%TYPE,
      cli_nome CLIENTES.NOME%TYPE,
      cli_email CLIENTES.EMAIL%TYPE,
      cli_tel   CLIENTES.TELEFONE%TYPE
   );

   pessoa cliente_rec;

   CURSOR c_clientes IS
      SELECT ID_CLIENTE, NOME, EMAIL, TELEFONE
      FROM CLIENTES;
```

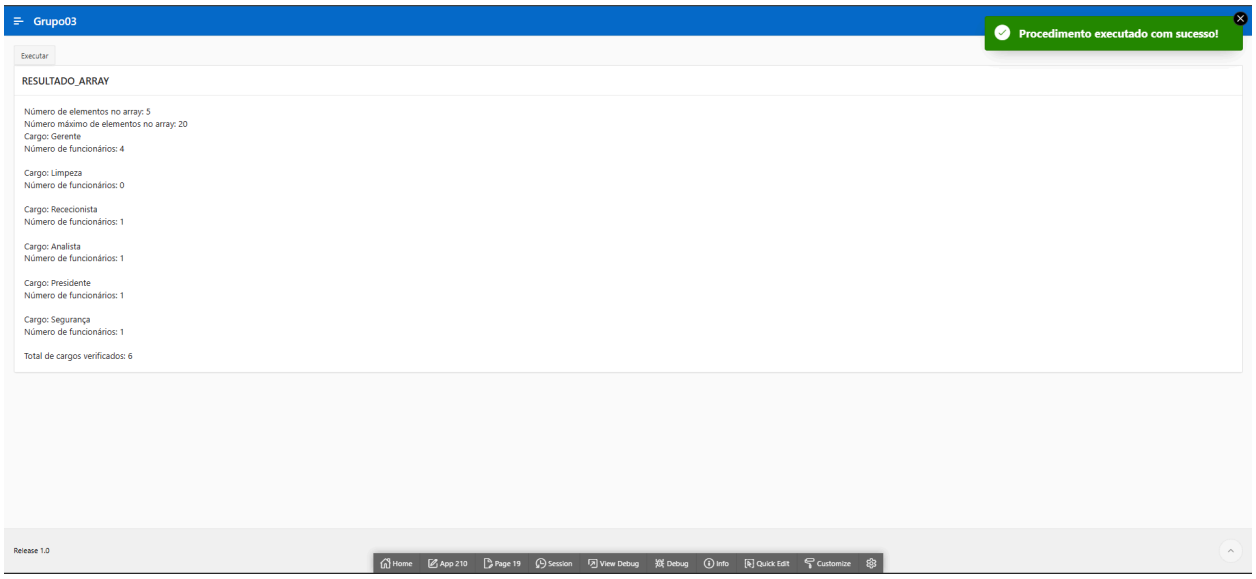
```
  v_html CLOB := '';
BEGIN
  OPEN c_clientes;
  LOOP
    FETCH c_clientes INTO pessoa;
    EXIT WHEN c_clientes%NOTFOUND;

    -- Aqui está o HTML formatado corretamente
    v_html := v_html ||
      '<div style="margin-bottom: 1em;">' ||
      '<b>ID:</b> ' || pessoa.cli_id || '<br>' ||
      '<b>Nome:</b> ' || pessoa.cli_nome || '<br>' ||
      '<b>Email:</b> ' || pessoa.cli_email || '<br>' ||
      '<b>Telefone:</b> ' || pessoa.cli_tel || '<br>' ||
      '</div>';
  END LOOP;
  CLOSE c_clientes;

  p_resultado := v_html;

EXCEPTION
  WHEN OTHERS THEN
    p_resultado := 'Erro: ' || SQLERRM;
END;
```



10.1

```
CREATE OR REPLACE PROCEDURE p_array_1 AS
  TYPE job_array IS VARRAY(20) OF VARCHAR2(10);
  prof job_array;
  howmany NUMBER;
BEGIN
  prof := job_array('clerk', 'salesman', 'manager', 'analyst', 'president');

  HTP.P('Número de elementos não matriciais: ' || prof.COUNT || '<br>');
  HTP.P('Número máximo de elementos não matriciais: ' || prof.LIMIT || '<br>');

  IF prof.LIMIT - prof.COUNT >= 1 THEN
    prof.EXTEND(1);
    prof(6) := 'clerk';
  END IF;

  FOR i IN prof.FIRST..prof.LAST LOOP
```

```
   SELECT COUNT(*) INTO howmany
   FROM funcionarios
   WHERE cargo = prof(i);

   HTP.P('Cargo: ' || prof(i) || ' - Número de funcionários: ' || TO_CHAR(howmany) ||
'<br>');
  END LOOP;
EXCEPTION
  WHEN OTHERS THEN
   HTP.P('Erro: ' || SQLERRM);
END;
```



10.2

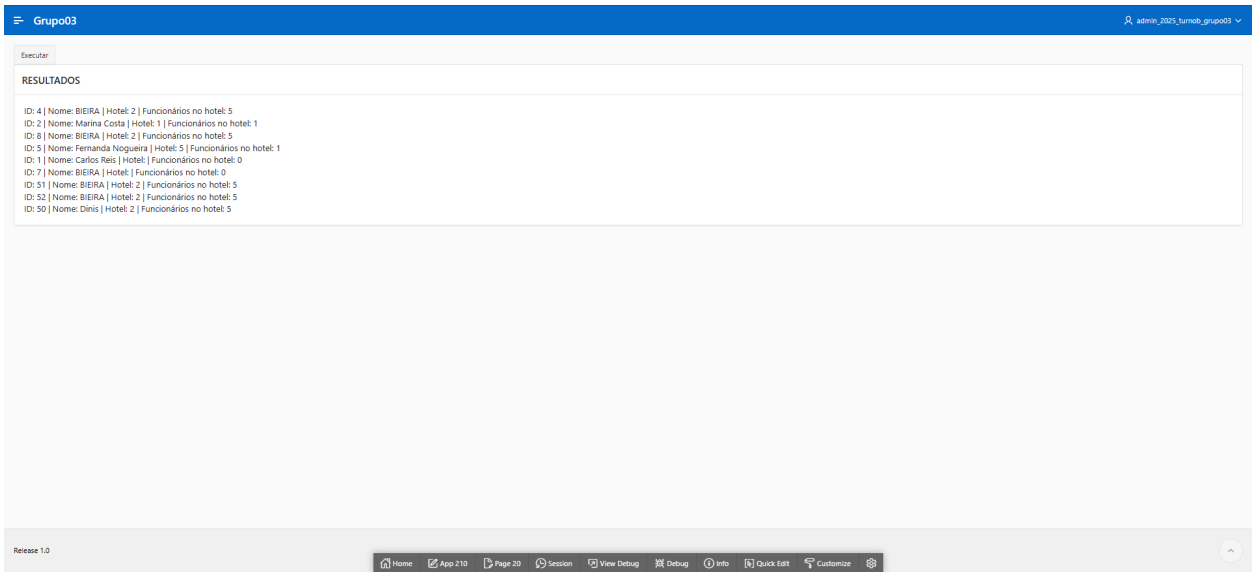
```
CREATE OR REPLACE PROCEDURE p_usar_array_cursor_func IS
  CURSOR cursor_func IS
   SELECT * FROM funcionarios;
```

```
  rec_func cursor_func%ROWTYPE;

  TYPE func_array IS VARRAY(25) OF cursor_func%ROWTYPE;
  arr_func func_array := func_array();

  howmany NUMBER := 0;
  i NUMBER := 1;
BEGIN
  OPEN cursor_func;
  LOOP
    FETCH cursor_func INTO rec_func;
    EXIT WHEN cursor_func%NOTFOUND;

    arr_func.EXTEND;
    arr_func(i) := rec_func;
    i := i + 1;
  END LOOP;
  CLOSE cursor_func;

  FOR j IN arr_func.FIRST .. arr_func.LAST LOOP
    SELECT COUNT(*) INTO howmany
    FROM funcionarios
    WHERE id_hotel = arr_func(j).id_hotel;

    HTP.P('ID: ' || arr_func(j).id_funcionario ||
        ' | Nome: ' || arr_func(j).nome ||
        ' | Hotel: ' || arr_func(j).id_hotel ||
        ' | Funcionários no hotel: ' || howmany || '<br>');
  END LOOP;
EXCEPTION
```

WHEN OTHERS THEN

HTP.P('Erro: ' || SQLERRM);

END;



11.1

11.1 código:

CREATE OR REPLACE TRIGGER verificar_idade_cliente

BEFORE INSERT OR UPDATE ON CLIENTES

FOR EACH ROW

DECLARE

v_idade NUMBER;

BEGIN v_idade := TRUNC(MONTHS_BETWEEN(SYSDATE,

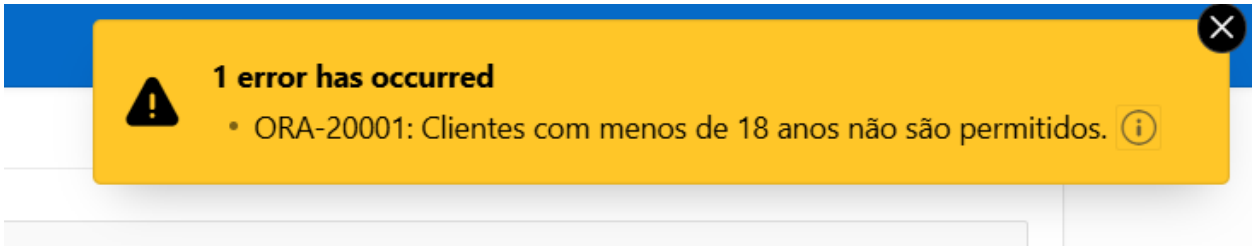
:NEW.DATA_NASCIMENTO) / 12);

IF v_idade < 18

THEN RAISE_APPLICATION_ERROR(-20001, 'Clientes com menos de 18 anos não são permitidos.');

END IF;

END;



11.2 código:

```
CREATE OR REPLACE TRIGGER trg_log_reservas
AFTER UPDATE OR DELETE ON reservas
FOR EACH ROW
BEGIN
  -- Atualização: registrar mudança de status
  IF UPDATING THEN
    INSERT INTO reservas (
      ID_RESERVA, ID_CLIENTE, ID_QUARTO,
      DATA_ENTRADA, DATA_SAIDA, STATUS, FORMA_PAGAMENTO
    )
    VALUES (
      :OLD.ID_RESERVA, :OLD.ID_CLIENTE, :OLD.ID_QUARTO,
      :OLD.DATA_ENTRADA, :OLD.DATA_SAIDA,
      'MODIFICADA: ' || :OLD.STATUS, :OLD.FORMA_PAGAMENTO
    );
  END IF;

  -- Eliminação: registrar reserva como cancelada
  IF DELETING THEN
    INSERT INTO reservas (
      ID_RESERVA, ID_CLIENTE, ID_QUARTO,
      DATA_ENTRADA, DATA_SAIDA, STATUS, FORMA_PAGAMENTO
    )
    VALUES (
```

```
    :OLD.ID_RESERVA, :OLD.ID_CLIENTE, :OLD.ID_QUARTO,
    :OLD.DATA_ENTRADA, :OLD.DATA_SAIDA,
    'CANCELADA', :OLD.FORMA_PAGAMENTO
  );
 END IF;
END;
```

d) Conclusão:

Com o trabalho foi possível aplicar de forma prática os conhecimentos aprendidos em sala dos principais recursos do PL/SQL. A utilização do Oracle APEX permitiu a realização de testes e visualizações dinâmicas e interativas de uma forma prática. As estruturas de controle, manipulação de dados e funções específicas foram utilizadas de maneira que foi possível visualizar um sistema funcional para a gestão de uma cadeia de hotéis.

Imagem do modelo relacional