28-05-2023

Relatório do 2 trabalho de base dados 2

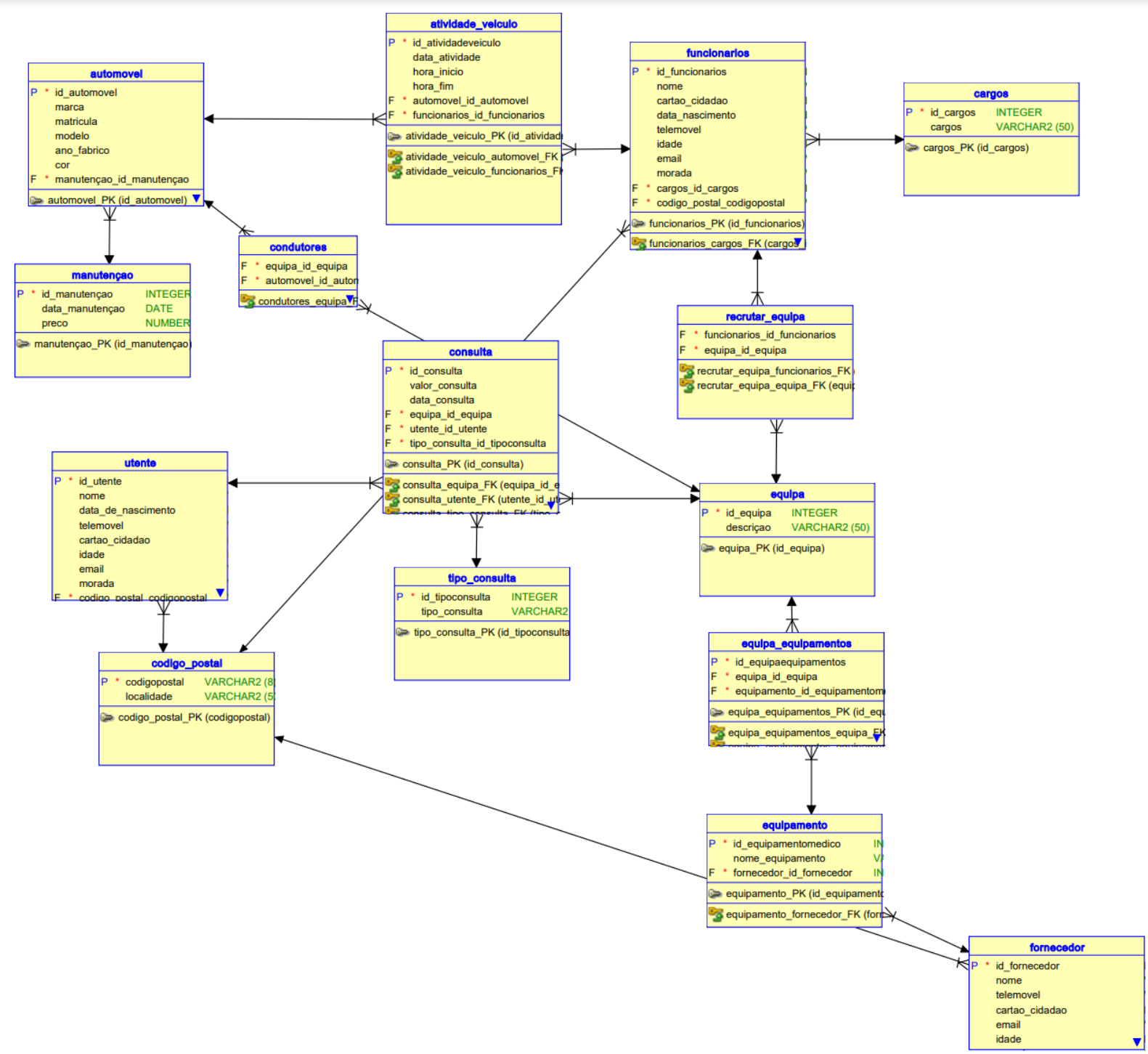
Gestão de uma unidade móvel de saúde

Grupo 8

Ivo Pimenta a48535
Ana Beatriz Alves a48520
Bernardo Nogueira a47344
Rui Aires a45296

Índice

Modelo de Dados ER



Criação das tabelas

Tabela atividade veículo

```
CREATE TABLE atividade_veiculo (
    id_atividadeveiculo          INTEGER NOT NULL,
    data_atividade               DATE,
    hora_inicio                  DATE,
    hora_fim                     DATE,
    automovel_id_automovel       INTEGER NOT NULL,
    funcionarios_id_funcionarios INTEGER NOT NULL
);

ALTER TABLE atividade_veiculo ADD CONSTRAINT atividade_veiculo_pk PRIMARY KEY ( id_atividadeveiculo );
```

Tabela automovel

```
CREATE TABLE automovel (
    id_automovel            INTEGER NOT NULL,
    marca                   VARCHAR2(50),
    matricula               VARCHAR2(30),
    modelo                  VARCHAR2(30),
    ano_fabrico             INTEGER,
    cor                     VARCHAR2(30),
    manutençao_id_manutençao INTEGER NOT NULL
);

ALTER TABLE automovel ADD CONSTRAINT automovel_pk PRIMARY KEY ( id_automovel );
```

Tabela cargos

```
CREATE TABLE cargos (
    id_cargos INTEGER NOT NULL,
    cargos    VARCHAR2(50)
);

ALTER TABLE cargos ADD CONSTRAINT cargos_pk PRIMARY KEY ( id_cargos );
```

Tabela código postal

```
CREATE TABLE codigo_postal (
    codigopostal VARCHAR2(8) NOT NULL,
    localidade   VARCHAR2(50)
);

ALTER TABLE codigo_postal ADD CONSTRAINT codigo_postal_pk PRIMARY KEY ( codigopostal );
```

Tabela condutores

```
CREATE TABLE condutores (
    equipa_id_equipa       INTEGER NOT NULL,
    automovel_id_automovel INTEGER NOT NULL
);
```

Tabela consulta

```
CREATE TABLE consulta (
    id_consulta                INTEGER NOT NULL,
    valor_consulta               NUMBER(5, 2),
    data_consulta                DATE,
    equipa_id_equipa               INTEGER NOT NULL,
    utente_id_utente             INTEGER NOT NULL,
    tipo_consulta_id_tipoconsulta INTEGER NOT NULL
);

ALTER TABLE consulta ADD CONSTRAINT consulta_pk PRIMARY KEY ( id_consulta );
```

Tabela equipa

```
CREATE TABLE equipa (
    id_equipa INTEGER NOT NULL,
    descriçao VARCHAR2(50)
);

ALTER TABLE equipa ADD CONSTRAINT equipa_pk PRIMARY KEY ( id_equipa );
```

Tabela equipa equipamentos

```
CREATE TABLE equipa_equipamentos (
    id_equipaequipamentos           INTEGER NOT NULL,
    equipa_id_equipa                INTEGER NOT NULL,
--  ERROR: Column name length exceeds maximum allowed length(30)
    equipamento_id_equipamentomedico INTEGER NOT NULL
);

ALTER TABLE equipa_equipamentos ADD CONSTRAINT equipa_equipamentos_pk PRIMARY KEY (
id_equipaequipamentos );
```

Tabela equipamento

```
CREATE TABLE equipamento (
    id_equipamentomedico     INTEGER NOT NULL,
    nome_equipamento         VARCHAR2(50),
    fornecedor_id_fornecedor INTEGER NOT NULL
);

ALTER TABLE equipamento ADD CONSTRAINT equipamento_pk PRIMARY KEY ( id_equipamentomedico );
```

Tabela fornecedor

```
CREATE TABLE fornecedor (
    id_fornecedor            INTEGER NOT NULL,
    nome                     VARCHAR2(50),
    telemovel                VARCHAR2(9),
    cartao_cidadao           INTEGER,
    email                    VARCHAR2(50),
    idade                    INTEGER,
    morada                   VARCHAR2(50),
    data_nascimento          DATE,
    codigo_postal_codigopostal VARCHAR2(8) NOT NULL
);

ALTER TABLE fornecedor ADD CONSTRAINT fornecedor_pk PRIMARY KEY ( id_fornecedor );
```

Tabela funcionarios

```
CREATE TABLE funcionarios (
    id_funcionarios           INTEGER NOT NULL,
    nome                      VARCHAR2(50),
    cartao_cidadao            INTEGER,
    data_nascimento           DATE,
    telemovel                 VARCHAR2(9),
    idade                     INTEGER,
    email                     VARCHAR2(30),
    morada                    VARCHAR2(50),
    cargos_id_cargos          INTEGER NOT NULL,
    codigo_postal_codigopostal VARCHAR2(8) NOT NULL
);

ALTER TABLE funcionarios ADD CONSTRAINT funcionarios_pk PRIMARY KEY ( id_funcionarios );
```

Tabela manutenção

```
CREATE TABLE manutençao (
    id_manutençao   INTEGER NOT NULL,
    data_manutençao DATE,
    preco           NUMBER(5, 2)
);

ALTER TABLE manutençao ADD CONSTRAINT manutençao_pk PRIMARY KEY ( id_manutençao );
```

Tabela recrutar equipa

```
CREATE TABLE recrutar_equipa (
    funcionarios_id_funcionarios INTEGER NOT NULL,
    equipa_id_equipa             INTEGER NOT NULL
);
```

Tabela tipo consulta

```
CREATE TABLE tipo_consulta (
    id_tipoconsulta INTEGER NOT NULL,
    tipo_consulta   VARCHAR2(30)
);

ALTER TABLE tipo_consulta ADD CONSTRAINT tipo_consulta_pk PRIMARY KEY ( id_tipoconsulta );
```

Tabela utente

```
CREATE TABLE utente (
    id_utente               INTEGER NOT NULL,
    nome                    VARCHAR2(50),
    data_de_nascimento      DATE,
    telemovel               VARCHAR2(9),
    cartao_cidadao          INTEGER,
    idade                   INTEGER,
    email                   VARCHAR2(50),
    morada                  VARCHAR2(50),
    codigo_postal_codigopostal VARCHAR2(8) NOT NULL
);

ALTER TABLE utente ADD CONSTRAINT utente_pk PRIMARY KEY ( id_utente );
```

Foreign Keys

```
ALTER TABLE atividade_veiculo
    ADD CONSTRAINT atividade_veiculo_automovel_fk FOREIGN KEY ( automovel_id_automovel )
        REFERENCES automovel ( id_automovel );

--  ERROR: FK name length exceeds maximum allowed length(30)
ALTER TABLE atividade_veiculo
    ADD CONSTRAINT atividade_veiculo_funcionarios_fk FOREIGN KEY ( funcionarios_id_funcionarios )
        REFERENCES funcionarios ( id_funcionarios );

ALTER TABLE automovel
    ADD CONSTRAINT automovel_manutençao_fk FOREIGN KEY ( manutençao_id_manutençao )
        REFERENCES manutençao ( id_manutençao );

ALTER TABLE condutores
    ADD CONSTRAINT condutores_automovel_fk FOREIGN KEY ( automovel_id_automovel )
        REFERENCES automovel ( id_automovel );

ALTER TABLE condutores
    ADD CONSTRAINT condutores_equipa_fk FOREIGN KEY ( equipa_id_equipa )
        REFERENCES equipa ( id_equipa );

ALTER TABLE consulta
    ADD CONSTRAINT consulta_equipa_fk FOREIGN KEY ( equipa_id_equipa )
        REFERENCES equipa ( id_equipa );

ALTER TABLE consulta
    ADD CONSTRAINT consulta_tipo_consulta_fk FOREIGN KEY ( tipo_consulta_id_tipoconsulta )
        REFERENCES tipo_consulta ( id_tipoconsulta );

ALTER TABLE consulta
    ADD CONSTRAINT consulta_utente_fk FOREIGN KEY ( utente_id_utente )
        REFERENCES utente ( id_utente );

ALTER TABLE equipa_equipamentos
    ADD CONSTRAINT equipa_equipamentos_equipa_fk FOREIGN KEY ( equipa_id_equipa )
```

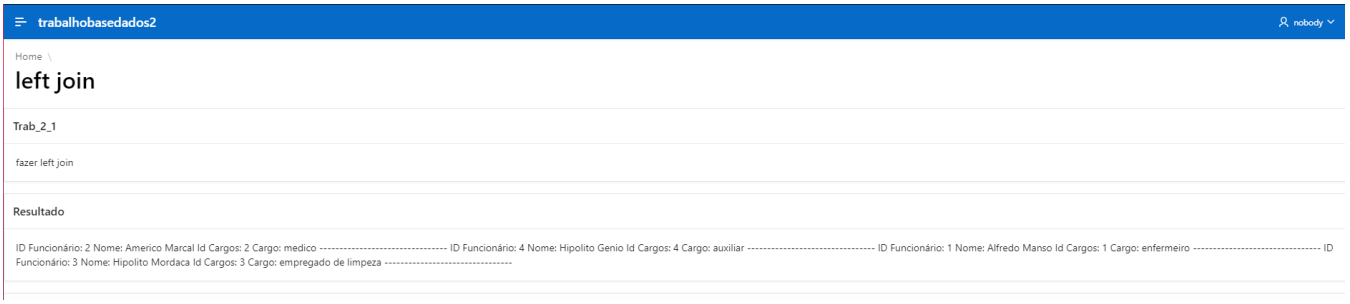
```
      REFERENCES equipa ( id_equipa );

--  ERROR: FK name length exceeds maximum allowed length(30)
ALTER TABLE equipa_equipamentos
   ADD CONSTRAINT equipa_equipamentos_equipamento_fk FOREIGN KEY (
equipamento_id_equipamentomedico )
      REFERENCES equipamento ( id_equipamentomedico );

ALTER TABLE equipamento
   ADD CONSTRAINT equipamento_fornecedor_fk FOREIGN KEY ( fornecedor_id_fornecedor )
      REFERENCES fornecedor ( id_fornecedor );

ALTER TABLE fornecedor
   ADD CONSTRAINT fornecedor_codigo_postal_fk FOREIGN KEY ( codigo_postal_codigopostal )
      REFERENCES codigo_postal ( codigopostal );

ALTER TABLE funcionarios
   ADD CONSTRAINT funcionarios_cargos_fk FOREIGN KEY ( cargos_id_cargos )
      REFERENCES cargos ( id_cargos );

ALTER TABLE funcionarios
   ADD CONSTRAINT funcionarios_codigo_postal_fk FOREIGN KEY ( codigo_postal_codigopostal )
      REFERENCES codigo_postal ( codigopostal );

ALTER TABLE recrutar_equipa
   ADD CONSTRAINT recrutar_equipa_equipa_fk FOREIGN KEY ( equipa_id_equipa )
      REFERENCES equipa ( id_equipa );

--  ERROR: FK name length exceeds maximum allowed length(30)
ALTER TABLE recrutar_equipa
   ADD CONSTRAINT recrutar_equipa_funcionarios_fk FOREIGN KEY ( funcionarios_id_funcionarios )
      REFERENCES funcionarios ( id_funcionarios );

ALTER TABLE utente
   ADD CONSTRAINT utente_codigo_postal_fk FOREIGN KEY ( codigo_postal_codigopostal )
      REFERENCES codigo_postal ( codigopostal );
```

Utilizar as funções AVG, MIN, MAX, COUNT e SUM

```
create or replace PROCEDURE TRAB_1 AS
avg_valor_consulta NUMBER;
min_valor_consulta NUMBER;
max_valor_consulta NUMBER;
count_valor_consulta NUMBER;
sum_valor_consulta NUMBER;
BEGIN
SELECT AVG(valor_consulta), MIN(valor_consulta), MAX(valor_consulta), COUNT(valor_consulta),
SUM(valor_consulta) INTO avg_valor_consulta, min_valor_consulta, max_valor_consulta, count_valor_consulta,
sum_valor_consulta
FROM consulta;
htp.p ('Média do valor das consultas é: ' || avg_valor_consulta ||'<br>');
htp.p (' O valor minimo das consultas é: ' || min_valor_consulta|| '<br>');
htp.p ('O valor maximo das consultas  é: ' || max_valor_consulta || '<br>');
htp.p ('Em ' || count_valor_consulta ||' consultas<br>');
htp.p ('Somatório do valor das consultas é: ' || sum_valor_consulta);
END TRAB_1;
```

Criar um procedimento que utilize LEFT JOIN

```
CREATE OR REPLACE PROCEDURE Trab_2_1 AS
 CURSOR cur IS
  SELECT id_funcionarios, nome, id_cargos, papel
  FROM cargos
  LEFT JOIN funcionarios ON funcionarios.id_funcionarios = cargos.id_cargos;
 TYPE reg IS RECORD(
  id_funcionarios funcionarios.id_funcionarios%TYPE,
  nome funcionarios.nome%TYPE,
   id_cargos cargos.id_cargos%TYPE,
  papel cargos.papel%TYPE
 );
 registo reg;
BEGIN
 OPEN cur;
 LOOP
  FETCH cur INTO registo.id_funcionarios, registo.nome, registo.id_cargos, registo.papel;
  EXIT WHEN cur%NOTFOUND;
  htp.p('ID Funcionário: ' || registo.id_funcionarios);
  htp.p('Nome: ' || registo.nome);
  htp.p('Id Cargos: ' || registo.id_cargos);
  htp.p('Cargo: ' || registo.papel);
  htp.p('--------------------------------');
 END LOOP;
 CLOSE cur;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
  htp.p('Informação não encontrada');
 WHEN OTHERS THEN
  htp.p('Erro 404');
END;
```



Criar um procedimento que utilize RIGHT JOIN

```
create or replace procedure Trab_2_2 as
Cursor cur is
select id_consulta,data_consulta, id_tipoconsulta, tipoconsulta
from consulta
right join tipo_consulta
on consulta.id_consulta=tipo_consulta.id_tipoconsulta;

type reg is Record(
 id_consulta consulta.id_consulta%type,
 data_consulta consulta.data_consulta%type,
 id_tipoconsulta tipo_consulta.id_tipoconsulta%type,
 tipoconsulta tipo_consulta.tipoconsulta%type
);
registo reg;
begin
 open cur;
 loop
 fetch cur into registo.id_consulta, registo.data_consulta, registo.id_tipoconsulta, registo.tipoconsulta;
 exit when cur%notfound;
 htp.p(' Data da Consulta: ' || registo.data_consulta || '( Id Consulta: ' || registo.id_consulta || ' ) : <br>
');
 htp.p(' Tipo de Consulta: ' || registo.tipoconsulta || ' ( ID Tipo de Consulta: ' || registo.id_tipoconsulta
|| ') : <br> ');
 end loop;
 close cur;
```

```
EXCEPTION
When no_data_found THEN
htp.p('Informação não encontrada');
When others then
htp.p('Erro 404');
end;
```

Criar um procedimento que utilize Single-Row Subqueries (pelo menos 1 parâmetro de entrada)

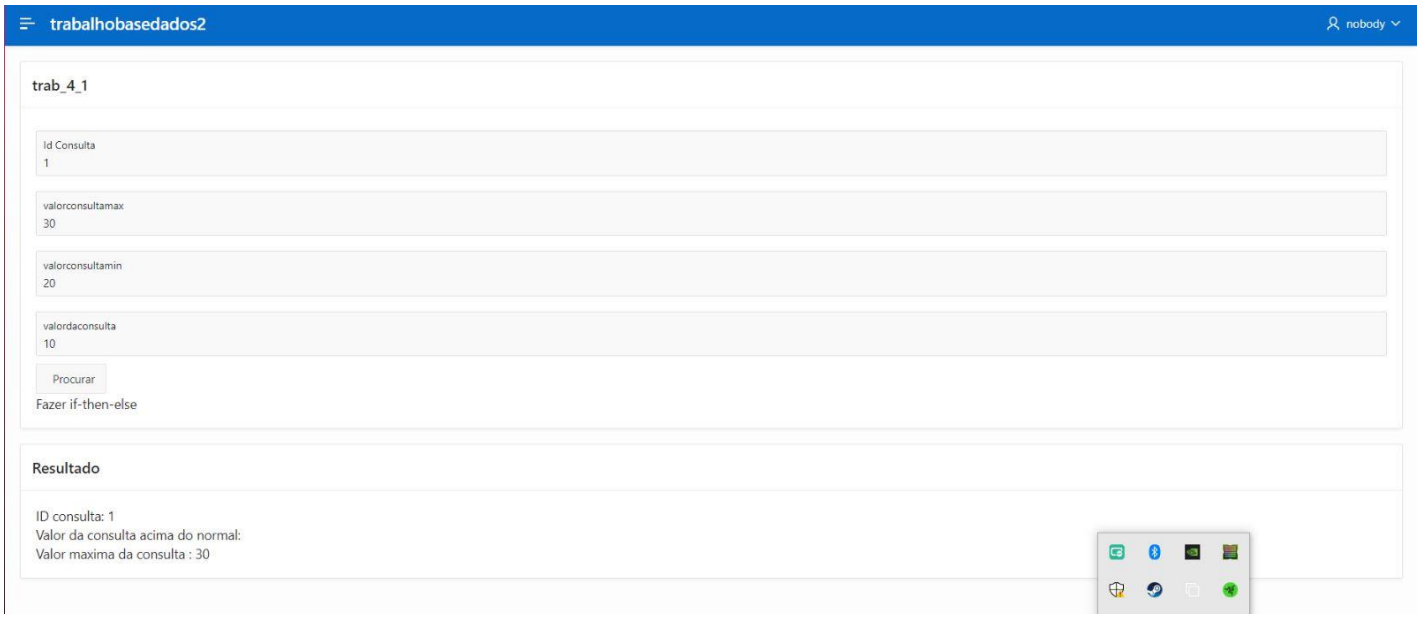
```
create or replace PROCEDURE TRAB_3_1(id_consul IN consulta.id_consulta%type) is
val_consul consulta.valor_consulta%type;
data_consul consulta.data_consulta%type;
BEGIN
SELECT valor_consulta, data_consulta
 INTO val_consul, data_consul
 FROM consulta
 WHERE id_consulta = id_consul;
HTP.P('Data da consulta : ' || data_consul ||', Valor da consulta: ' || val_consul || '<br>');
END;
```

Criar um procedimento que utilize Multiple-Row Subqueries (pelo menos 1 parâmetro de entrada)

```
Create or replace PROCEDURE TRAB_3_2 AS
 nome utente.nome%TYPE;
 localidade codigo_postal.localidade%TYPE;

 CURSOR cursor_ute IS
 SELECT nome, localidade
 FROM utente, codigo_postal
 WHERE utente.id_utente = codigo_postal.codigopostal;

BEGIN
 OPEN cursor_ute;
 LOOP
 FETCH cursor_ute INTO nome, localidade;

 EXIT WHEN cursor_ute%NOTFOUND;
 HTP.P('Nome do Utente: '||nome||', Localidade do Utente: ' || localidade || '<br>');
 END LOOP;
 HTP.P('Numero de Utentes: '|| TO_CHAR(cursor_ute%ROWCOUNT || '<br>'));
 CLOSE cursor_ute;
END TRAB_3_2;
```

Criar um procedimento que utilize a estrutura IF-THEN-ELSE

```
create or replace procedure TRAB_4_1(id_con in consulta.id_consulta%TYPE,
Valor_consultaMAX in NUMBER,
Valor_consultaMIN in NUMBER, valornormal in NUMBER) as
 valor_consulta_registo consulta%ROWTYPE;
 valor_cons consulta.valor_consulta%TYPE;
 Begin
 SELECT *
 INTO valor_consulta_registo
 FROM consulta
 Where id_consulta=id_con;
 valor_cons:=valor_consulta_registo.id_consulta;
 IF valornormal > Valor_consultaMAX THEN
 valor_cons:= valornormal-Valor_consultaMAX;
 ELSIF valor_cons <= Valor_consultaMIN AND valornormal <= Valor_consultaMAX
 Then valor_cons := valornormal;
 ELSE
 valor_cons := Valor_consultaMIN - Valor_consultaMAX ;
 END IF;
 HTP.P('ID consulta: ' || valor_consulta_registo.id_consulta);
HTP.P('<BR>Valor da consulta acima do normal: ' || TO_CHAR(valornormal -
Valor_consultaMAX) || ' ');

HTP.P('<BR>Valor maxima da consulta : ' || TO_CHAR(Valor_consultaMAX) || ' ');
EXCEPTION
WHEN OTHERS THEN
HTP.P('ID consulta ' || TO_CHAR(id_con) || 'não existe');
END TRAB_4_1;
```



Criar um procedimento que utilize a estrutura CASE (pelo menos 1 parâmetro de entrada)

```
create or replace procedure trab_4_2(codigo in utente.codigo_postal_codigopostal%type) as
registo utente%rowtype;
mora utente.morada%type;
begin
 select *
 into registo
 from utente
 where codigo_postal_codigopostal = codigo;
 mora:= registo.morada;
 case (registo.codigo_postal_codigopostal)
 when '343-332' then
 mora:= ' rua dos campeoes ';
 when '890-463' then
 mora:= ' rua dos mafiosos ';
 when '085-896' then
 mora:= ' rua dos ladroes ';
 end case;
 HTP.P(' Nome do utente: ' || registo.nome);
 HTP.P(' Codigo postal: ' || registo.codigo_postal_codigopostal);
```

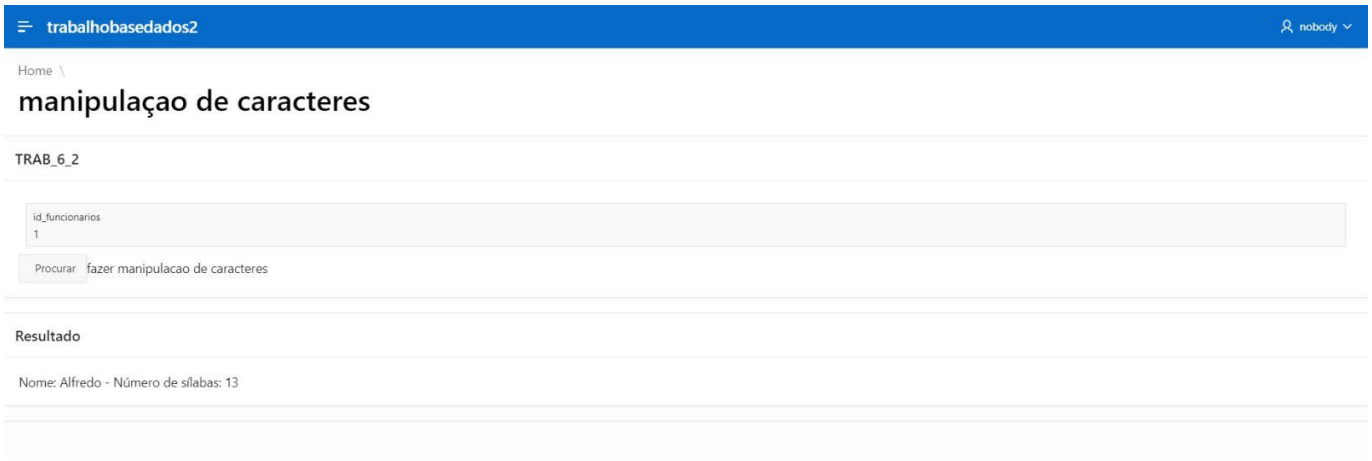
```
HTP.P(' telemovel: ' || registo.telemovel);
HTP.P(' Morada: ' || registo.morada);
end trab_4_2;
```

Utilizar as estruturas LOOP, FOR e WHILE

```
create or replace PROCEDURE TRAB_5 AS
nome fornecedor.nome%TYPE;
mo fornecedor.morada%TYPE;
CURSOR fornecedores_f IS
SELECT nome, morada
FROM fornecedor;
BEGIN
OPEN fornecedores_f;
LOOP
FETCH fornecedores_f INTO nome, mo;
EXIT WHEN fornecedores_f%NOTFOUND;
HTP.P('O nome dos fornecedor é '||nome||' e tem a morada '||mo||'.<br>');
END LOOP;
CLOSE fornecedores_f;
END TRAB_5;
```

Criar um procedimento (ou funçao) que utilize pelo menos 2 funçoes de manipulaçao de caracteres

```
create or replace PROCEDURE TRAB_6_2(id_fun in funcionarios.id_funcionarios%type) AS
 funcionarios_rec funcionarios%ROWTYPE;
BEGIN
 select*
 into funcionarios_rec
 from funcionarios
 where id_funcionarios = id_fun;
 HTP.P('Nome: ' || RPAD(funcionarios_rec.nome, 8) || '- Número de sílabas: ' ||
length(funcionarios_rec.nome));
END TRAB_6_2;
```



Criar um procedimento que utilize 1 cursor e dados de diferentes tabelas

```
create or replace procedure trab_7_1 (id_ute in utente.id_utente%type) as
nome utente.nome%type;
mora utente.morada%type;
cod_posta codigo_postal.codigopostal%type;
cursor ute is
select nome, morada, codigopostal
from utente A, codigo_postal C
where A.codigo_postal_codigopostal = C.codigopostal
 and id_utente= id_ute;
 begin
 open ute;
 fetch ute into nome, mora, cod_posta;
 if(ute % notfound) then
 htp.p('O utente nº '|| id_ute || ' nao existe');
else
htp.p('Nome do utente: ' || nome || '<br>');
htp.p('Morada: ' || mora || '<br>');
htp.p('Codigo-postal: ' || cod_posta || '<br>');
end if;
close ute;
end trab_7_1;
```

Criar um procedimento que utilize 2 cursor e dados de diferentes tabelas

```
create or replace procedure trab_7_2(id_ute in
utente.id_utente%type, id_for in
fornecedor.id_fornecedor%type) as
 nome utente.nome%type;
 mora utente.morada%type;
 cod_posta codigo_postal.codigopostal%type;
 nome_for fornecedor.nome%type;
 mora_for fornecedor.morada%type;
cursor ute is
 select nome,morada,codigo_postal_codigopostal
 from utente A, codigo_postal C
 where A.codigo_postal_codigopostal = C.codigopostal
 and id_utente = id_ute;
 cursor forn is
 select nome, morada, codigo_postal_codigopostal
 from fornecedor FO, codigo_postal CP
 where FO.codigo_postal_codigopostal = CP.codigopostal
 and id_fornecedor = id_for;
begin
 htp.p('----------cursor1---------- <br>');
 open ute;
 loop
 fetch ute into nome,mora,cod_posta;
 exit when ute%notfound;
 htp.p('O nome do utente é: '||nome||' <br> A
morada é: '||mora|| ' <br> O codigo-postal é: '||cod_posta);
 end loop;
 htp.p('<br> Numero de registos do utente: '||
to_char(ute%rowcount) || '.');
 close ute;
```

```
 htp.p('<br> ----------cursor2---------- <br>');
 open forn;
 loop
 fetch forn into nome_for,mora_for,cod_posta;
 exit when forn%notfound;
 htp.p('<br> O nome do fornecedor é: '||nome_for
||'<br> A morada: '||mora_for||'<br> Codigo-Postal: '||cod_posta);
 end loop;
 htp.p('<br> O numero de registos dos fornecedores '||
to_char(forn%rowcount)||'.');
 close forn;
end trab_7_2;
```

Criar um procedimento que utilize cursores com subquerys

```
create or replace procedure trab_7_3 as
reg_ute utente%rowtype;
reg_post codigo_postal%rowtype;
nregistos number;
cursor cursor1 is
select utente.nome,
utente.morada,codigo_postal.localidade,utente.data_de_nascimento
from utente,codigo_postal
where utente.codigo_postal_codigopostal=codigo_postal.codigopostal
and (utente.codigo_postal_codigopostal,utente.data_de_nascimento) in
 (select utente.codigo_postal_codigopostal, min
 (utente.data_de_nascimento)
 from utente
 group by utente.codigo_postal_codigopostal);
 begin
 open cursor1;
 htp.p('<br> ----------cursor1---------- <br>');
 loop
 fetch cursor1
 into reg_ute.nome,reg_ute.morada, reg_post.localidade,reg_ute.data_de_nascimento;
 exit when cursor1%notfound;
 htp.p('O utente '||reg_ute.nome||' ,com a morada '||reg_ute.morada|| ' é o mais novo da
localidade: ' ||reg_post.localidade || '<br>');
end loop;
nregistos:=cursor1%rowcount;
htp.p('<br> Total de utentes: '||nregistos);
close cursor1;
end trab_7_3;
```

Criar um procedimento que utilize REF CURSOR data type

```
create or replace procedure trab_7_4(num_func in number) as type func_refcur_typ is ref cursor
return

funcionarios%rowtype;

func_cursor func_refcur_typ;

procedure process_func_cv(func_cursor in func_refcur_typ) is

 func funcionarios%rowtype;

 begin

 htp.p('Cursor da tabela Funcionarios'||'<br>');

loop

fetch func_cursor into func;

exit when func_cursor%notfound;

 htp.p( func.id_funcionarios || ', '|| func.nome || '<br>');

end loop;

end;

begin

open func_cursor for

select *

from funcionarios

where nome like '%r%';

process_func_cv(func_cursor);
```

```
close func_cursor;
end;
```

Criar um procedimento que permita inserir registos na Base de Dados

```
create or replace PROCEDURE TRAB_8_1 (id_cons consulta.id_consulta%TYPE,
val_cons consulta.valor_consulta%TYPE, tip_con
consulta.tipo_consulta_id_tipoconsulta%TYPE, id_equip
consulta.equipa_id_equipa%TYPE) IS
BEGIN
INSERT INTO consulta (id_consulta, valor_consulta, tipo_consulta_id_tipoconsulta,
equipa_id_equipa)
 VALUES(id_cons, val_cons, tip_con, id_equip);
 HTP.P('O registo da consulta foi inserido com sucesso');
EXCEPTION
 WHEN DUP_VAL_ON_INDEX THEN
 HTP.P('Já existe equipa com o mesmo id' || TO_CHAR(id_cons));
END TRAB_8_1
```

Criar um procedimento que permita eliminar registos na Base de Dados

```
CREATE OR REPLACE PROCEDURE TRAB_8_2 (IDD_FUNCIONARIOS
funcionarios.id_funcionarios%TYPE) IS
  registo_funcionarios funcionarios%ROWTYPE;
BEGIN
  SELECT *
  INTO registo_funcionarios
  FROM funcionarios
  WHERE id_funcionarios = IDD_FUNCIONARIOS;

  DELETE FROM funcionarios
  WHERE id_funcionarios = IDD_FUNCIONARIOS;

  HTP.P('O funcionario com o id ' || TO_CHAR(IDD_FUNCIONARIOS) || ' foi deletado.');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    HTP.P('O funcionario com o id ' || TO_CHAR(IDD_FUNCIONARIOS) || ' não existe.');
END TRAB_8_2;
```

Criar um procedimento que permita alterar registos na Base de Dados

```
CREATE OR REPLACE PROCEDURE TRAB_8_3 (
  id_funcion funcionarios.id_funcionarios%TYPE,
  nom_funcion funcionarios.nome%TYPE,
  id_carg funcionarios.cargos_id_cargos%TYPE,
  codigo_post funcionarios.codigo_postal_codigopostal%TYPE
) IS
BEGIN
  UPDATE funcionarios
  SET nome = nom_funcion,
      cargos_id_cargos = id_carg,
      codigo_postal_codigopostal = codigo_post
  WHERE id_funcionarios = id_funcion;

  HTP.P('O funcionario com o id ' || TO_CHAR(id_funcion) || ' foi corretamente atualizado.');

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    HTP.P('Este funcionario nao existe');
END TRAB_8_3;
```

Criar um procedimento que utilize estruturas de dados do tipo RECORD

```
CREATE OR REPLACE PROCEDURE trab_9_1(
   IDD NUMBER,
   nom VARCHAR2,
   tele VARCHAR2,
   mora VARCHAR2,
   data_nas DATE,
   cartao_cid NUMBER,
   codigo_post VARCHAR2
)
AS
   TYPE equi_rec IS RECORD (
      esp_equ NUMBER(2),
      codigo_post VARCHAR2(30)
   );

   TYPE ute_rec IS RECORD (
      IDD_ute utente.id_utente%TYPE,
      nom_ute utente.nome%TYPE,
      tele_ute utente.telemovel%TYPE,
      mora_ute utente.morada%TYPE,
      data_nas_ute utente.data_de_nascimento%TYPE,
      equipamento equi_rec
   );

   utente ute_rec;
```

```
BEGIN
   utente.IDD_ute := IDD;
   utente.nom_ute := nom;
   utente.tele_ute := tele;
   utente.mora_ute := mora;
   utente.data_nas_ute := data_nas;
   utente.equipamento.esp_equ := cartao_cid;
   utente.equipamento.codigo_post := codigo_post;

   htp.p('ID do utente: ' || utente.IDD_ute || '<br> Nome: ' || utente.nom_ute || '<br>
telemovel: ' ||
      utente.tele_ute || '<br> Morada: ' || utente.mora_ute ||
      '<br> Data de Nascimento: ' || TO_CHAR(utente.data_nas_ute, 'DD-MON-YYYY') ||
      '<br> Especificaçao do Equipamento: ' || utente.equipamento.esp_equ ||
      '<br> Tipo de Colaborador: ' || utente.equipamento.codigo_post);
END trab_9_1;
```

Criar um procedimento que utilize estruturas de dados do tipo RECORD e CURSOR

```
CREATE OR REPLACE PROCEDURE trab_9_2 AS
  TYPE uten_rec IS RECORD(
    id_uten utente.id_utente%TYPE,
    nom_uten utente.nome%TYPE,
    tele_uten utente.telemovel%TYPE,
    mora_uten utente.morada%TYPE,
    data_nas_uten utente.data_de_nascimento%TYPE
  );

  utente uten_rec;
  CURSOR cursor_uten IS
    SELECT id_utente, nome, telemovel, morada, data_de_nascimento
    FROM utente;
BEGIN
  OPEN cursor_uten;
  LOOP
    FETCH cursor_uten INTO utente.id_uten, utente.nom_uten, utente.tele_uten,
utente.mora_uten, utente.data_nas_uten;
    EXIT WHEN cursor_uten%NOTFOUND;
    htp.p('ID do utente: '||utente.id_uten ||
      ' Nome: '||utente.nom_uten ||' Telemovel: '||utente.tele_uten ||
      ' Morada: '||utente.mora_uten|| ' Data de Nascimento: '||utente.data_nas_uten||
      '<br>');
  END LOOP;
  CLOSE cursor_uten;
```

```
END trab_9_2;
```

Criar um procedimento que utilize estruturas de dados do tipo ARRAY

```
create or replace procedure trab_10_1 as
type cod_array is array(13) of varchar2(20);
cod cod_array;
linhas number;
begin
cod:=cod_array('1600-404','1069-300','5500-535');
for i in cod.first..cod.last loop
select count(*) into linhas
from funcionarios
where codigo_postal_codigopostal=cod(i);
htp.p('codigo-postal: '||rpad(cod(i),50) ||'<br>'||
'numero de funcionarios: '||to_char(linhas)||'<br>');
end loop;
htp.p('numero de elementos do array:' ||cod.count||'<br>');
end trab_10_1;
```

Criar um procedimento que utilize estruturas de dados do tipo ARRAY e CURSOR

```
create or replace procedure trab_10_2 as
cursor cur is
 select *
 from funcionarios;
fun_rec cur%rowtype;
type fun_array is array(20) of cur%rowtype;
fun fun_array;
linhas number;
num number:=1;
begin
 fun:= fun_array();
 open cur;
 loop
 fetch cur into fun_rec;
 exit when cur%notfound;
 fun.extend(1);
 fun(num):=fun_rec;
 num:= num+1;
 end loop;
 close cur;
htp.p('Nome dos funcionarios: <br>');
for i in fun.first..fun.last loop
select count(*) into linhas
from funcionarios
where nome=fun(i).nome;
htp.p(rpad(fun(i).nome,100));
```

```
end loop;
htp.p(' <br> Numero de elementos do array: '||fun.count);
end trab_10_2;
```

Criar um trigger que utilize a declaração BEFORE

Crie um trigger que utilize a declaração AFTER

Criar um package que possua vários procedimentos (pelo menos 2) e pelo menos 1 funçao

Ecras do APEX

Conclusão

Em conclusão, este relatório abordou a implementação de uma base de dados no Apex utilizando a linguagem PL/SQL para a gestão de uma unidade móvel.

Por outras palavras, concluímos que a implementação de uma base de dados utilizando PL/SQL, traz benefícios significativos.

Essa abordagem permite o controle eficiente de informações críticas, agiliza os processos de atendimento, facilita a tomada de decisões e melhora a qualidade dos serviços prestados, se este sistema fosse mesmo posto em prática.

Em suma, a base de dados no Apex, utilizando a linguagem PL/SQL, é uma solução para a implementação do programa.

Para além disso, a criação das tabelas em si não teve grande dificuldade, mas a adptação à aplicação em questão, limita-nos ao básico e sentimos que muitos dos campos criados poderiam ser mais configurados, pois certas ideias não sabiamos como as resolver. A falta de tempo tambem foi um fator que nos projudicou para o que era pedido neste trabalho. No entanto, apenas a falta de conhecimento sobre o programa deixou em aberto muito daquilo que queriamos fazer.