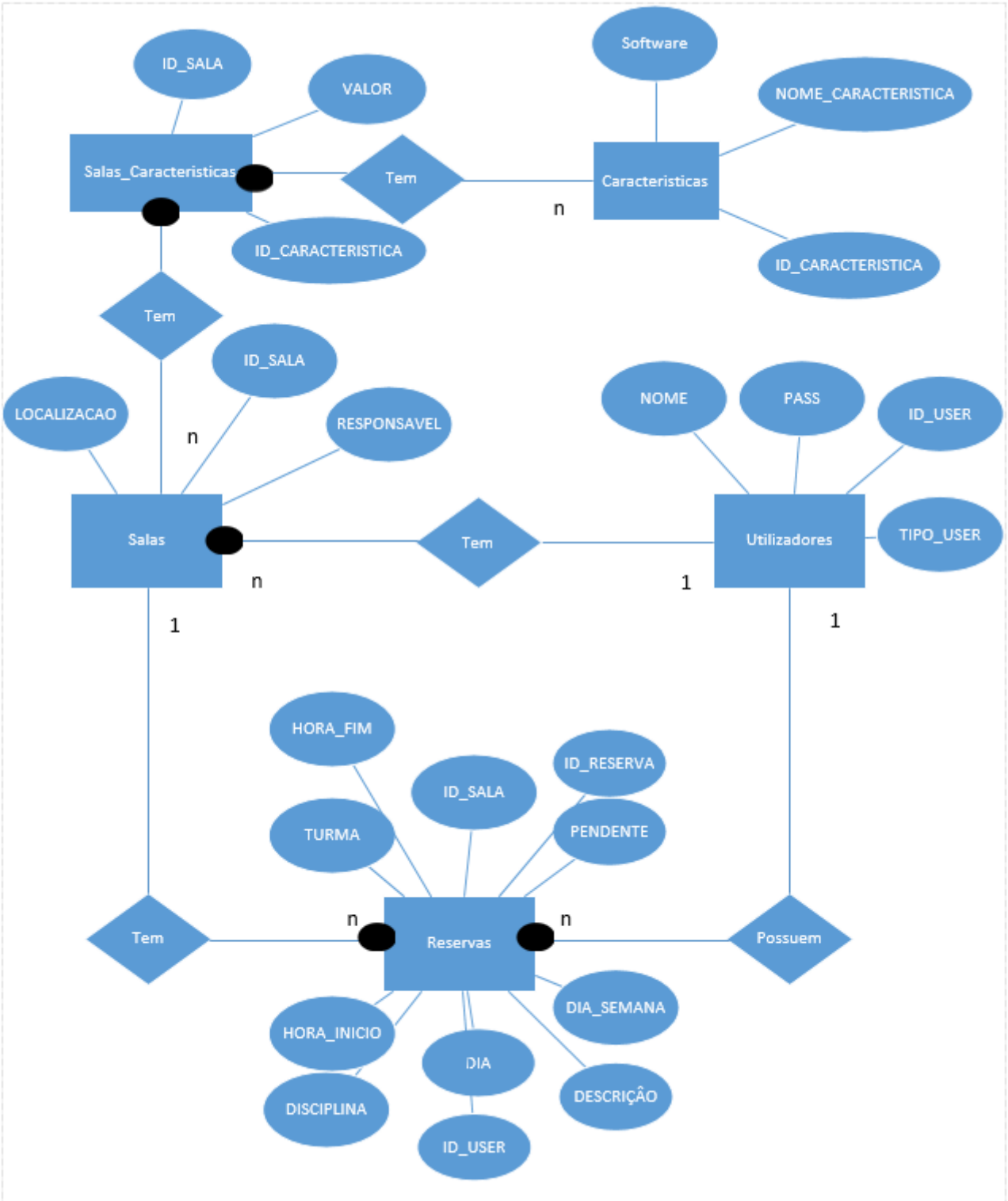
*DER-Diagrama Entidade-Relacionamento



```
CREATE TABLE Salas(
   ID_SALA NUMBER(4) PRIMARY KEY,
   LOCALIZACAO NVARCHAR2(6)NOT NULL,
   RESPONSAVEL NUMBER(4) NOT NULL
);

CREATE TABLE Carateristicas(
   ID_CARATERISTICA NUMBER(3) PRIMARY KEY,
   NOME_CARATERISTICA NVARCHAR2(20),
   SOFTWARE CHAR(1) CHECK (SOFTWARE IN (0, 1))
);

CREATE TABLE Salas_Carateristicas(
   SALA NUMBER(4),
   CARATERISTICA NUMBER(3),
   VALOR NUMBER(2)
);
```

```
CREATE TABLE Utilizadores(

   ID_UTILIZADOR NUMBER(4) PRIMARY KEY,

   NOME NVARCHAR2(60) NOT NULL,

   PALAVRA_PASSE NVARCHAR2(20) NOT NULL
CHECK(LENGTH(PALAVRA_PASSE) > 2),

   EMAIL NVARCHAR2(30) NOT NULL UNIQUE,

   TIPO_USER CHAR(1) CHECK (TIPO_USER = 'P' OR TIPO_USER
= 'R' OR TIPO_USER = 'F' OR TIPO_USER = 'A')

);

CREATE TABLE Reservas(

   ID_RESERVA NUMBER(6) PRIMARY KEY,

   SALA NUMBER(4) NOT NULL,

   HORA_INICIO NUMBER(4) NOT NULL,

   HORA_FIM NUMBER(4) NOT NULL,

   DIA_SEMANA NVARCHAR2(3),

   TURMA NVARCHAR2(30),

   DISCIPLINA NVARCHAR2(30),

   DOCENTE NVARCHAR2(40),

   DIA DATE,

   PENDENTE CHAR(1) CHECK (PENDENTE IN (0, 1)), --NAO
ACEITA BOOLEAN NEM BIT?

   DESCRICAO NVARCHAR2(50)

);
```

```
ALTER TABLE Reservas ADD CONSTRAINT RESERVA_SALA FOREIGN
KEY (SALA) REFERENCES Salas(ID_SALA);

ALTER TABLE Salas ADD CONSTRAINT SALA_RESPONSAVEL FOREIGN
KEY (RESPONSAVEL) REFERENCES Utilizadores(ID_UTILIZADOR);

ALTER TABLE Salas_Carateristicas ADD CONSTRAINT
PK_Salas_Carateristicas PRIMARY KEY (CARATERISTICA,
SALA);

ALTER TABLE Salas_Carateristicas ADD CONSTRAINT
Qual_CARATERISTICA FOREIGN KEY (CARATERISTICA) REFERENCES
Carateristicas(ID_CARATERISTICA);

ALTER TABLE Salas_Carateristicas ADD CONSTRAINT Qual_SALA
FOREIGN KEY (SALA) REFERENCES Salas(ID_SALA);
```

Implementação dos Subprogramas Armazenados

#Funcionalidade 1

Esta funcionalidade tem como objetivo retornar a lista de salas existentes e carateristicas, o código de criação é o seguinte:

```
CREATE OR REPLACE VIEW getSalas AS

  SELECT  SALAS.ID_SALA, SALAS.LOCALIZACAO,
SALAS.RESPONSAVEL, UTILIZADORES.NOME,
CARATERISTICAS.ID_CARATERISTICA,
CARATERISTICAS.NOME_CARATERISTICA,
CARATERISTICAS.SOFTWARE, SALAS_CARATERISTICAS.VALOR

    FROM SALAS

    INNER JOIN SALAS_CARATERISTICAS ON SALAS.ID_SALA =
SALAS_CARATERISTICAS.SALA

    INNER JOIN CARATERISTICAS ON
CARATERISTICAS.ID_CARATERISTICA =
SALAS_CARATERISTICAS.CARATERISTICA

    INNER JOIN UTILIZADORES ON SALAS.RESPONSAVEL =
UTILIZADORES.ID_UTILIZADOR

    ORDER BY SALAS.LOCALIZACAO;
```

+Funcionalidade 2

Esta funcionalidade tem como objetivo listar salas que cumpram os requesitos, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE
getSalas_Carateristicas(v_carateristicas IN tabc_valor,
lista OUT SYS_REFCURSOR)

IS BEGIN

  OPEN lista FOR SELECT getSalas.ID_SALA

    FROM getSalas, TABLE(v_carateristicas)
v_carateristicas

    WHERE getSalas.NOME_CARATERISTICA =
v_carateristicas.CARATERISTICA

      AND getSalas.VALOR >= v_carateristicas.VALOR

    GROUP BY getSalas.ID_SALA

    ORDER BY getSalas.ID_SALA;

END;

/
```

#Funcionalidade 3

Esta funcionalidade tem como objetivo listar todas as salas, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE getAllSalas(lista OUT
SYS_REFCURSOR)

IS BEGIN

  OPEN lista FOR SELECT * FROM getSalas

    ORDER BY getSalas.ID_SALA;

END;

/
```

#Funcionalidade 4

Esta funcionalidade tem como objetivo adicionar salas, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE addSala(LOCALIZACAO
NVARCHAR2, RESPONSAVEL NUMBER, v_carateristicas IN
tabc_valor)

IS l_index NUMBER;

id NUMBER;

BEGIN

  INSERT INTO SALAS VALUES(NULL, LOCALIZACAO,
RESPONSAVEL)

  RETURNING ID_SALA INTO id;

  l_index := v_carateristicas.FIRST;

  WHILE (l_index IS NOT NULL) LOOP

    addCar(id, v_carateristicas(l_index).CARATERISTICA,
v_carateristicas(l_index).VALOR);

    l_index := v_carateristicas.NEXT(l_index);

  END LOOP;

END addSala;

/
```

*Funcionalidade 5

Esta funcionalidade tem como objetivo apagar uma sala, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE popSala(v_idSala IN NUMBER)

IS BEGIN

  DELETE FROM SALAS WHERE ID_SALA = v_idSala;

END popSala;

/
```

*Funcionalidade 6

Esta funcionalidade tem como objetivo editar uma sala, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE editSala(LOCALIZACAO
NVARCHAR2, RESPONSAVEL NUMBER, v_idSala IN NUMBER,
v_carateristicas IN tabc_valor, lista OUT SYS_REFCURSOR)

IS BEGIN

  popSala(v_idSala);

  addSala(LOCALIZACAO, RESPONSAVEL, v_carateristicas);

END editSala;

/
```

#Funcionalidade 7

Esta funcionalidade tem como objetivo retornar o horário de uma certa sala numa certa data, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE getSalas_horario(v_sala IN
NVARCHAR2, v_dia IN DATE, horarios OUT SYS_REFCURSOR)

  IS BEGIN

      OPEN horarios FOR

        SELECT  RESERVAS.ID_RESERVA,
RESERVAS.HORA_INICIO, RESERVAS.HORA_FIM, RESERVAS.TURMA,
RESERVAS.DISCIPLINA, RESERVAS.DOCENTE, RESERVAS.PENDENTE,
RESERVAS.DESCRICAO

          FROM RESERVAS, SALAS

          WHERE SALAS.LOCALIZACAO = v_sala AND
RESERVAS.SALA = SALAS.ID_SALA AND

          (DIA = v_dia OR

          DIA_SEMANA = TO_CHAR(v_dia, 'DY'));

END getSalas_horario;

 /
```

+Funcionalidade 8

Esta funcionalidade tem como objetivo diz se a sala esta disponível num certo dia e hora, o código de criação é o seguinte:

```
CREATE OR REPLACE FUNCTION salaDisponivel(v_sala IN
NVARCHAR2, v_dia IN DATE, v_inicio IN NUMBER, v_fim IN
NUMBER)
RETURN NUMBER IS
  valor NUMBER := 0;
  CURSOR horarios (v_sala IN NVARCHAR2, v_dia IN DATE) IS
    SELECT RESERVAS.ID_RESERVA, RESERVAS.HORA_INICIO,
RESERVAS.HORA_FIM, RESERVAS.TURMA, RESERVAS.DISCIPLINA,
RESERVAS.DOCENTE, RESERVAS.PENDENTE, RESERVAS.DESCRICAO
      FROM RESERVAS, SALAS
      WHERE SALAS.LOCALIZACAO = v_sala AND RESERVAS.SALA =
SALAS.ID_SALA AND
      (DIA = v_dia OR
      DIA_SEMANA = TO_CHAR(v_dia, 'DY'));
BEGIN
  FOR horario in horarios(v_sala, v_dia) LOOP
    IF(horario.HORA_INICIO = v_inicio) THEN
      valor := 1;
    END IF;
    IF(valor = 1) THEN
      IF(horario.PENDENTE = 0) THEN
        RETURN 0;
      END IF;
      IF(horario.HORA_FIM = v_fim) THEN
        valor := 0;
      END IF;
    END IF;
  END LOOP;
  Return 1;
END salaDisponivel;
/
```

*Funcionalidade 9

Esta funcionalidade tem como objetivo mostrar os pedidos de reserva de um docente, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE getPedido(v_id IN NVARCHAR2,
pedidos OUT SYS_REFCURSOR)

  IS BEGIN

    OPEN pedidos FOR

      SELECT RESERVAS.ID_RESERVA, RESERVAS.SALA,
SALAS.LOCALIZACAO, RESERVAS.HORA_INICIO,
RESERVAS.HORA_FIM, RESERVAS.DIA_SEMANA, RESERVAS.TURMA,
RESERVAS.DISCIPLINA, RESERVAS.DOCENTE, RESERVAS.DIA,
RESERVAS.PENDENTE, RESERVAS.DESCRICAO

        FROM RESERVAS INNER JOIN SALAS ON SALAS.ID_SALA =
RESERVAS.SALA

        WHERE RESERVAS.DOCENTE = (SELECT ID_UTILIZADOR
FROM UTILIZADORES WHERE ID_UTILIZADOR = v_id);

END getPedido;

/
```

#Funcionalidade 10

Esta funcionalidade tem como objetivo mostrar os pedidos de reserva pendentes de um responsável de sala, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE getPedidoPendente(v_id IN
NVARCHAR2, pedidos OUT SYS_REFCURSOR)

  IS BEGIN

    OPEN pedidos FOR

      SELECT RESERVAS.ID_RESERVA, RESERVAS.SALA,
SALAS.LOCALIZACAO, RESERVAS.HORA_INICIO,
RESERVAS.HORA_FIM, RESERVAS.DIA_SEMANA, RESERVAS.TURMA,
RESERVAS.DISCIPLINA, RESERVAS.DOCENTE, RESERVAS.DIA,
RESERVAS.PENDENTE, RESERVAS.DESCRICAO

        FROM RESERVAS INNER JOIN SALAS ON SALAS.ID_SALA =
RESERVAS.SALA

        WHERE SALAS.RESPONSAVEL = (SELECT ID_UTILIZADOR
FROM UTILIZADORES WHERE ID_UTILIZADOR = v_id);

END getPedidoPendente;

/
```

#Funcionalidade 11

Esta funcionalidade tem como objetivo aceitar um pedido de reserva, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE aceitarPedido(v_pedido IN
NUMBER)

  IS BEGIN

    UPDATE RESERVAS SET RESERVAS.PENDENTE = 0

    WHERE RESERVAS.ID_RESERVA = v_pedido;

END aceitarPedido;

/
```

*Funcionalidade 12

Esta funcionalidade tem como objetivo inserir um pedido de reserva, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE addPedido(SALA IN NUMBER,
HORA_INICIO IN NUMBER, HORA_FIM NUMBER, DIA_SEMANA IN
NVARCHAR2, TURMA IN NVARCHAR2, DISCIPLINA IN NVARCHAR2,
DOCENTE IN NVARCHAR2, DIA IN DATE, PENDENTE IN CHAR,
DESCRICAO IN NVARCHAR2)

  IS BEGIN

    INSERT INTO RESERVAS VALUES(NULL, SALA, HORA_INICIO,
HORA_FIM, DIA_SEMANA, TURMA, DISCIPLINA, DOCENTE, DIA,
PENDENTE, DESCRICAO);

END addPedido;

/
```

#Funcionalidade 13

Esta funcionalidade tem como objetivo apagar um pedido de reserva, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE popPedidoPendente(v_pedido IN
NUMBER)

  IS BEGIN

    DELETE FROM RESERVAS

    WHERE RESERVAS.ID_RESERVA = v_pedido;

END popPedidoPendente;

/
```

+Funcionalidade 14

Esta funcionalidade tem como objetivo editar um pedido de reserva pendente, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE editPedidoPendente(v_pedido
IN NUMBER, SALA IN NUMBER, HORA_INICIO IN NUMBER,
HORA_FIM NUMBER,

  DIA_SEMANA IN NVARCHAR2, TURMA IN NVARCHAR2, DISCIPLINA
IN NVARCHAR2, DOCENTE IN NVARCHAR2, DIA IN DATE,
DESCRICAO IN NVARCHAR2)

  IS BEGIN

    popPedidoPendente(v_pedido);

    addPedido(SALA, HORA_INICIO, HORA_FIM, DIA_SEMANA,
TURMA, DISCIPLINA, DOCENTE, DIA, 1, DESCRICAO);

END editPedidoPendente;

/
```

#Funcionalidade 15

Esta funcionalidade tem como objetivo obter um utilizador, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE getUser(v_email IN VARCHAR2,
v_pass IN VARCHAR2, v_lista OUT SYS_REFCURSOR) IS

BEGIN

  OPEN v_lista FOR

  SELECT * FROM Utilizadores WHERE(EMAIL = v_email AND
PALAVRA_PASSE = v_pass);

END;

/
```

+Funcionalidade 16

Esta funcionalidade tem como objetivo adicionar utilizadores, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE AddUser(V_NOME IN NVARCHAR2,
V_PALAVRA_PASSE IN NVARCHAR2, V_EMAIL IN NVARCHAR2,
V_TIPO_USER IN CHAR)

IS BEGIN

  INSERT INTO Utilizadores VALUES(null, V_NOME,
V_PALAVRA_PASSE, V_EMAIL, V_TIPO_USER);

END;

/
```

#Funcionalidade 17

Esta funcionalidade tem como objetivo apagar utilizadores, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE PopUser(v_id IN NUMBER)

IS BEGIN

  DELETE FROM SALAS WHERE RESPONSAVEL = v_id;

  DELETE FROM Utilizadores WHERE ID_UTILIZADOR = v_id;

END;

/
```

+Funcionalidade 18

Esta funcionalidade tem como objetivo editar utilizadores, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE EditUser(v_id IN NUMBER,
V_NOME IN NVARCHAR2, V_PALAVRA_PASSE IN NVARCHAR2,
V_EMAIL IN NVARCHAR2, V_TIPO_USER IN CHAR)

IS BEGIN

  PopUser(v_id);

  AddUser(V_NOME, V_PALAVRA_PASSE, V_EMAIL, V_TIPO_USER);

  UPDATE Utilizadores

  SET ID_UTILIZADOR = v_id

  WHERE ID_UTILIZADOR = (SELECT MAX(ID_UTILIZADOR) FROM
Utilizadores);

END;/
```

#Funcionalidade 19

Esta funcionalidade tem como objetivo alterar o responsável de sala de uma determinada sala, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE addSala_User(v_user IN
NUMBER, v_sala IN NUMBER)

IS BEGIN

  UPDATE SALAS SET RESPONSAVEL = v_user

  WHERE ID_SALA = v_sala;

END;

/
```

*Funcionalidade 20

Esta funcionalidade tem como objetivo listar todos os utilizadores, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE listaUsers(v_cursor OUT
SYS_REFCURSOR)

  IS BEGIN

  OPEN v_cursor FOR

    SELECT * FROM UTILIZADORES ORDER BY ID_UTILIZADOR;

END;

/
```

#+*Funcionalidade 21

Esta funcionalidade tem como objetivo adicionar uma carateristica a uma sala, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE addCar(sala IN NUMBER, id IN
NVARCHAR2, valor IN NUMBER)

IS BEGIN

  INSERT INTO Salas_Carateristicas VALUES(sala,

     (SELECT ID_CARATERISTICA FROM CARATERISTICAS WHERE
NOME_CARATERISTICA = id), valor);

END;

/
```

```
CREATE OR REPLACE PROCEDURE addCarat(sala IN NVARCHAR2,
id IN NUMBER, valor IN NUMBER)

IS BEGIN

  INSERT INTO Salas_Carateristicas VALUES(

    (SELECT ID_SALA FROM Salas WHERE Salas.LOCALIZACAO =
sala), id, valor);

END;

/
```

```
CREATE OR REPLACE PROCEDURE addCarSala(v_sala IN
NVARCHAR2, id_car IN NVARCHAR2, valor IN NUMBER)

IS BEGIN

  INSERT INTO Salas_Carateristicas VALUES(

    (SELECT ID_SALA FROM SALAS WHERE LOCALIZACAO =
v_sala),

    (SELECT ID_CARATERISTICA FROM CARATERISTICAS WHERE
NOME_CARATERISTICA = id_car),

    valor);

END;

/
```

+Funcionalidade 22

Esta funcionalidade tem como objetivo fazer o login, o código de criação é o seguinte:

```
CREATE OR REPLACE FUNCTION DoLogin(v_email IN NVARCHAR2,
v_pass IN NVARCHAR2)

  RETURN number IS cnumber number;

BEGIN

  SELECT COUNT(*) INTO cnumber FROM Utilizadores
WHERE(EMAIL = v_email AND PALAVRA_PASSE = v_pass);

  RETURN cnumber;

END;

/
```

#Funcionalidade 23

Esta funcionalidade tem como objetivo obter a sala, o código de criação é o seguinte:

```
CREATE OR REPLACE PROCEDURE getSala(v_id IN NUMBER, lista
OUT SYS_REFCURSOR)

IS BEGIN

  OPEN lista FOR SELECT * FROM SALAS

    WHERE ID_SALA = v_id

    ORDER BY ID_SALA;

END;

/
```

*Funcionalidade 24

Esta funcionalidade tem como objetivo gerar um novo id_sala, o código de criação é o seguinte:

```
CREATE OR REPLACE TRIGGER SALA

BEFORE INSERT ON SALAS

FOR EACH ROW

BEGIN

   SELECT N_SALA.NEXTVAL INTO :NEW.ID_SALA FROM DUAL;

END SALA;

/
```

#Funcionalidade 25

Esta funcionalidade tem como objetivo gerar uma nova id_caracteristica, o código de criação é o seguinte:

```
CREATE OR REPLACE TRIGGER CARATERISTICA

BEFORE INSERT ON CARATERISTICAS

FOR EACH ROW

BEGIN

   SELECT N_CARATERISTICA.NEXTVAL INTO
:NEW.ID_CARATERISTICA FROM DUAL;

END CARATERISTICA;

/
```

*Funcionalidade 26

Esta funcionalidade tem como objetivo gerar uma nova id_utilizador, o código de criação é o seguinte:

```
CREATE OR REPLACE TRIGGER GERA_ID_UTILIZADOR

BEFORE INSERT ON UTILIZADORES

FOR EACH ROW

BEGIN

   SELECT N_UTILIZADOR.NEXTVAL INTO :NEW.ID_UTILIZADOR
FROM DUAL;

   SELECT UPPER(:NEW.TIPO_USER) INTO :NEW.TIPO_USER FROM
DUAL;

END GERA_ID_UTILIZADOR;

/
```

#Funcionalidade 27

Esta funcionalidade tem como objetivo gerar uma nova id_reserva, o código de criação é o seguinte:

```
CREATE OR REPLACE TRIGGER GERA_ID_RESERVA

BEFORE INSERT ON RESERVAS

FOR EACH ROW

BEGIN

   SELECT N_RESERVA.NEXTVAL INTO :NEW.ID_RESERVA FROM
DUAL;

END GERA_ID_RESERVA;

/
```

+Funcionalidade 28

Esta funcionalidade tem como objetivo definir o responsável de cada sala, o código de criação é o seguinte:

```
CREATE OR REPLACE TRIGGER E_RESPONSAVEL

BEFORE INSERT OR UPDATE ON SALAS

FOR EACH ROW

   DECLARE V_TIPO_USER CHAR(1);

   user_invalido EXCEPTION;

   BEGIN

   SELECT TIPO_USER INTO V_TIPO_USER FROM UTILIZADORES
WHERE ID_UTILIZADOR = :NEW.RESPONSAVEL;

   IF V_TIPO_USER <> 'R' THEN

        RAISE user_invalido;

   END IF;



EXCEPTION

   WHEN user_invalido THEN

   DBMS_OUTPUT.PUT_LINE('Utilizador invalido');

END GERA_ID_RESERVA;

/
```