universidade de aveiro

Gestão de uma Rede de Transporte de Doentes não urgentes

João Vieira nº mec 50458

Lara Matos nº mec 95228

Grupo P10G9

Base de Dados – 2022/23

Licenciatura em Engenharia em Computadores e Informática

Análise de Requisitos

Considerando um sistema de gestão de transporte de doentes, este sistema terá que oferecer:

- Adicionar **Pacientes** ao sistema. Estes **Pacientes** são pessoas que têm que ter Número de Cartão de Cidadão, Nº de Utente, Nome, Morada e Data de Nascimento. Um determinado paciente possui um Historial Clínico.
- Adicionar **Bombeiros** ao sistema. Estes **Bombeiros** são os responsáveis pelo **Transporte** de doentes não urgentes e um **Bombeiro** está associado a um **Transporte**. Um **Bombeiro** só pode pertence a um quartel e tem que ter a si associado um Nº interno, patente e valências.
- Um **Transporte** está associado a apenas um **Bombeiro** e possui um Nº identificador, matrícula, lotação, tipologia e a lista de utentes que o Bombeiro tem que recolher.
- O **Cuidador** Informal é responsável por um **Paciente** e pode efetuar um pedido de transporte para o mesmo, de acordo com as suas capacidades motoras/físicas (**necessidades**).
- O **Paciente** pode efetuar um pedido de transporte de acordo com as suas capacidades motoras/físicas (**Necessidades**).
- A **Unidade de Tratamento** é quem disponibiliza os serviços para o **Paciente** – **Consultas** – que são marcas por um **Profissional** com autorização para fazer as marcações de acordo com a sua função.
- Durante a **Unidade de Tratamento**, o **Profissional** tem acesso ao **Historial Clínico** do **Paciente**.
- O **Profissional** pode ser responsável por uma **Consulta**.

Entidades

Pessoa
Cartão de Cidadão
Número de Utente (Chave Primária)
Nome
Morada
Data de Nascimento

Paciente
Cartão de Cidadão
Número de Utente
Nome
Morada
Data de Nascimento
ID Local

Quartel
ID Quartel (Chave Primária)
Nome
Localidade
Morada
ID Local

Unidade de Tratamento
ID Unidade (Chave Primária)
ID Local
Nome
Morada
GPS

Transporte
ID Transporte (Chave Primária)
ID Quartel
Matrícula
Lotação
Tipologia
Lista de Recolha
ID Local

Localidade
ID Local (Chave Primária)
Nome
Coordenadas

Cuidador Informal
Cartão de Cidadão (Chave Primária)
Número de Utente
Nome
Morada
Data de Nascimento

Bombeiro
Cartão de Cidadão
Número de Utente
Nome
Morada
Data de Nascimento
Número Interno Bombeiro (Chave Primária)
ID Quartel
Patente
Valências

Profissional
Cartão de Cidadão
Número de Utente
Nome
Morada
Data de Nascimento
Cédula Profissional (Chave Primária)
ID Unidade
Função

Requisição
ID NEC (Chave Primária)
Nome
Tipologia

Necessidades
ID Requisição (Chave Primária)
Tipologia

Evento Clínico
Número de Utente (Chave Primária)
Data
Causa
Diagnóstico
Tratamentos
Vacinas
Receitas

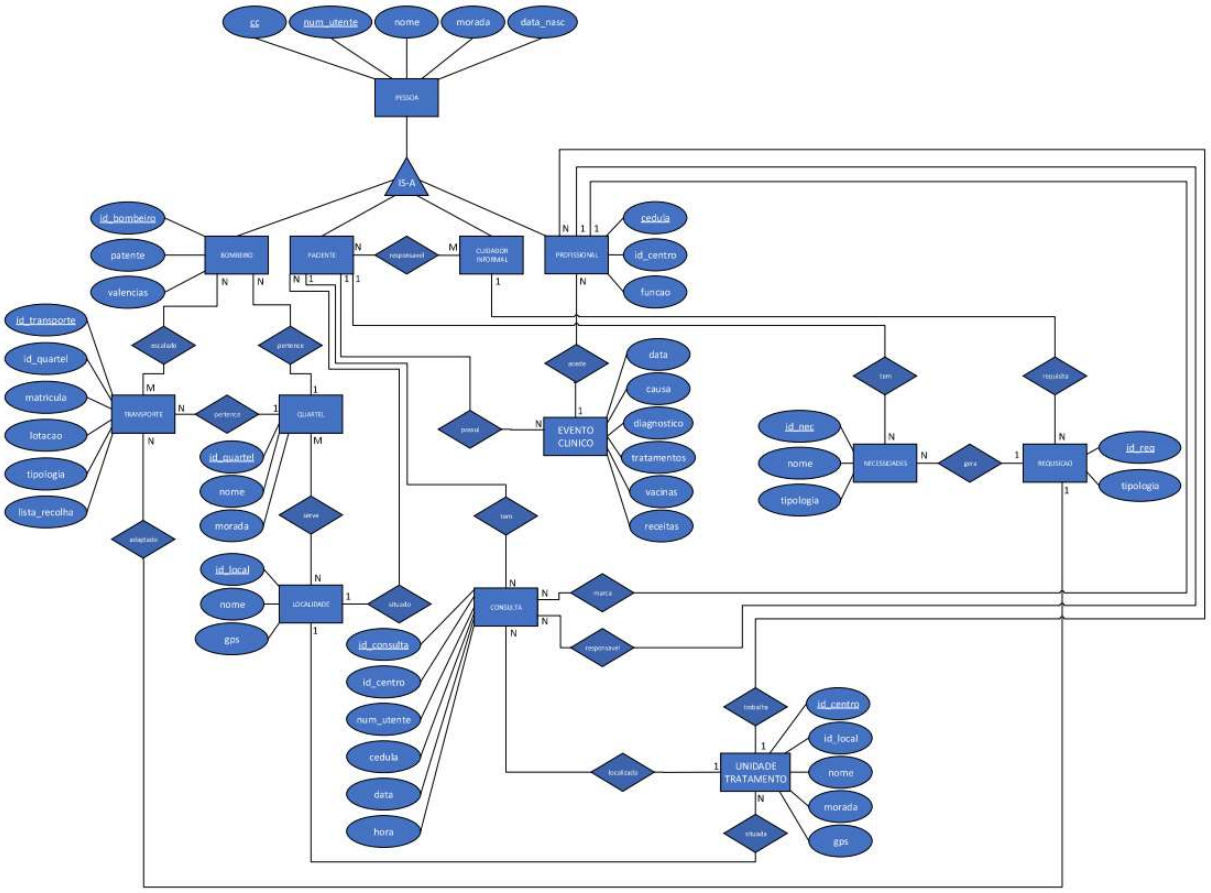
Consulta
ID Consulta (Chave Primária)
ID Unidade
Número de Utente
Cédula Profissional
Data
Hora

Serve
ID Quartel
ID Local

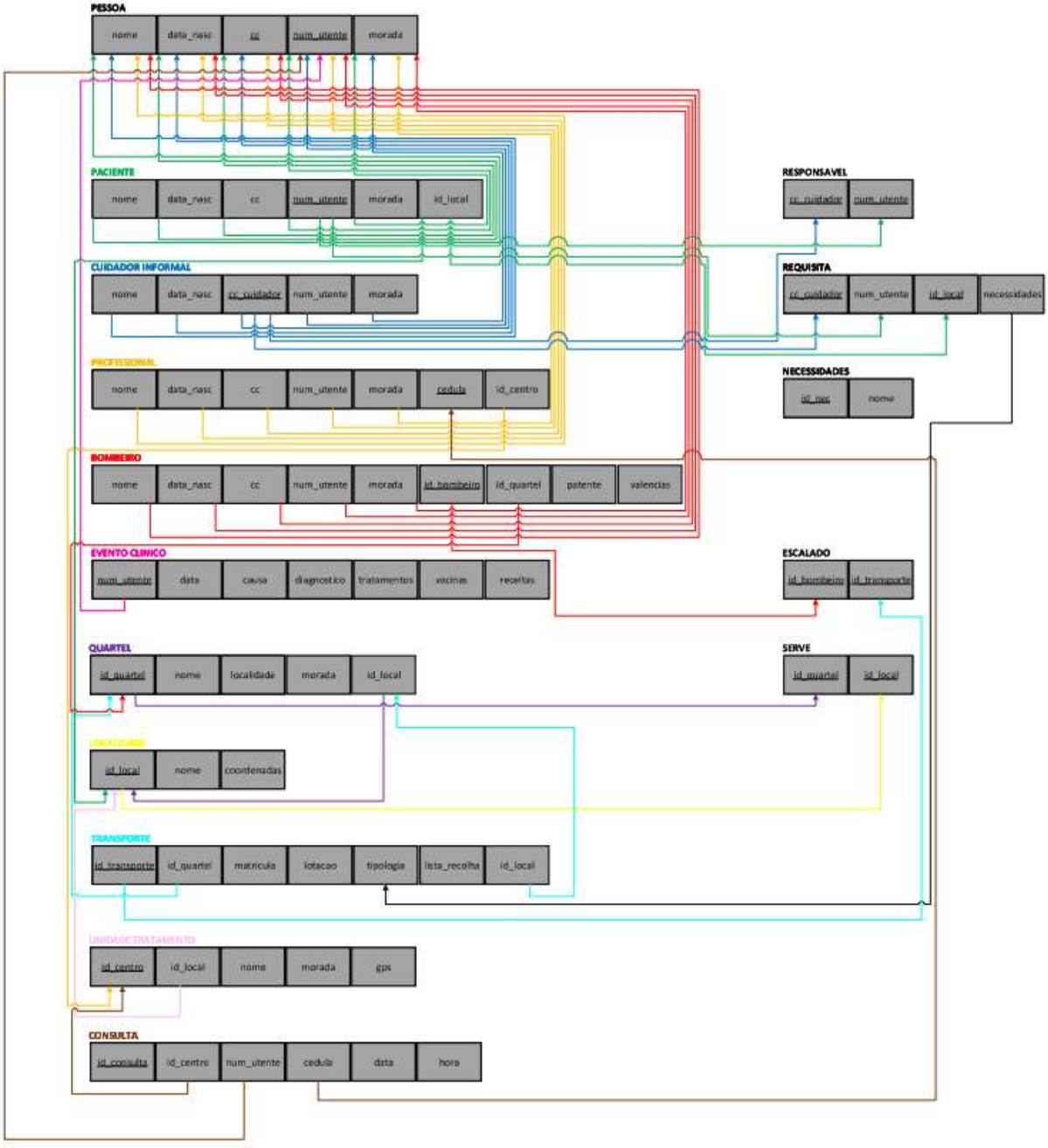
Escalado
Número Interno Bombeiro
ID Transporte

Responsável
Cartão de Cidadão - Cuidador
Número de Utente - Paciente

Diagrama Entidade – Relação



Modelo Relacional



SQL DLL

Para criar a base de dados deste projeto foram utilizadas as funcionalidades de **Create**, **Drop** e de comandos SQL DLL.

Exemplos:

Drop Tables	Drop Schema

```
DROP TABLE IF EXISTS RTD.Responsavel;
DROP TABLE IF EXISTS RTD.Escalado;
DROP TABLE IF EXISTS RTD.Serve;
DROP TABLE IF EXISTS RTD.Consulta;
DROP TABLE IF EXISTS RTD.EventoClinico;
DROP TABLE IF EXISTS RTD.Necessidades;
DROP TABLE IF EXISTS RTD.Requisicao;
DROP TABLE IF EXISTS RTD.Transporte;
DROP TABLE IF EXISTS RTD.Profissional;
DROP TABLE IF EXISTS RTD.UnidadeTratamento;
DROP TABLE IF EXISTS RTD.Bombeiro;
DROP TABLE IF EXISTS RTD.Quartel;
DROP TABLE IF EXISTS RTD.Cuidador;
DROP TABLE IF EXISTS RTD.Paciente;
DROP TABLE IF EXISTS RTD.Localidade;
DROP TABLE IF EXISTS RTD.Pessoa;
GO
```

```
DROP SCHEMA IF EXISTS RTD;
GO
```

Create Tables:

```
CREATE SCHEMA RTD;
GO


CREATE TABLE RTD.Pessoa (
    [cc]            INT             NOT NULL        UNIQUE              ,
    [num_utente]    INT             NOT NULL        PRIMARY KEY         ,
    [nome]          VARCHAR(32)     NOT NULL        UNIQUE              ,
    [morada]        VARCHAR(64)                     UNIQUE              ,
    [data_nasc]     DATE            NOT NULL        UNIQUE
    )
GO

CREATE TABLE RTD.Paciente (
    [cc]            INT             NOT NULL                    FOREIGN KEY     REFERENCES RTD.Pessoa(cc)           ,
    [num_utente]    INT             NOT NULL    UNIQUE          FOREIGN KEY     REFERENCES RTD.Pessoa(num_utente)   ,
    [nome]          VARCHAR(32)     NOT NULL                    FOREIGN KEY     REFERENCES RTD.Pessoa(nome)         ,
    [morada]        VARCHAR(64)                                 FOREIGN KEY     REFERENCES RTD.Pessoa(morada)       ,
    [data_nasc]     DATE            NOT NULL                    FOREIGN KEY     REFERENCES RTD.Pessoa(data_nasc)    ,
    [id_local]      INT             NOT NULL
    )
GO
```

SQL DML

Para inserir elementos nas tabelas da base de dados utiliza-se o **Insert**.

Exemplos:

```
-- INSERT - TABELA RTD.UnidadeTratamento
INSERT INTO RTD.UnidadeTratamento ([id_centro], [id_local], [nome], [morada], [GPS])
VALUES (11111111, 1, 'Centro Hospitalar de Lisboa', 'Rua Bela Moça', '38.7223º N, 9.1393º W'),
       (22222222, 2, 'Centro Hospitalar do Porto', 'Rua da Boa Tripa', '41.1691º N, 8.6145º W'),
       (33333333, 3, 'Centro Hospitalar de Coimbra', 'Avenida Direita', '40.2118º N, 8.4298º W'),
       (44444444, 4, 'Hospital da Praia de Faro', 'Rua das Maçãs', '37.0169º N, 7.9332º W'),
       (55555555, 5, 'Hospital de Aveiro', 'Rua da Boa Hora', '40.6381º N, 8.6513º W');

-- INSERT - TABELA RTD.Localidade
INSERT INTO RTD.Localidade ([id_local], [nome], [coordenadas])
VALUES (1, 'Lisboa', '38.7071º N, 9.1355º W'),
       (2, 'Porto', '41.1620º N, 8.6217º W'),
       (3, 'Coimbra', '40.2035º N, 8.4106º W'),
       (4, 'Faro', '37.0157º N, 7.9350º W'),
       (5, 'Aveiro', '40.6362º N, 8.6487º W');
```

Views

```
/* VIEW - Todos os Profissionais */
CREATE VIEW TodosProfissionais AS SELECT * FROM RTD.Profissional;
--
--
GO
/* VIEW - Todos os Transportes */
CREATE VIEW TodosTransportes AS SELECT * FROM RTD.Transporte;
--
--
GO
/* VIEW - Todos os Quarteis */
CREATE VIEW TodosQuarteis AS SELECT * FROM RTD.Quartel;
--
--
GO
/* VIEW - Todos as Unidades de Tratamento */
CREATE VIEW TodasUnidades AS SELECT * FROM RTD.UnidadeTratamento;
--
--
GO

/* VIEW - Profissionais com mais de 1 consulta */
CREATE VIEW RTD.ConsultasProfissionais AS
SELECT pr.cedula, p.nome AS nome_profissional, COUNT(c.id_consulta) AS total_consultas
FROM RTD.Profissional pr
JOIN RTD.Pessoa p ON pr.cc = p.cc
JOIN RTD.Consulta c ON pr.cedula = c.cedula
GROUP BY pr.cedula, p.nome
HAVING COUNT(c.id_consulta) >= 2;
--
--
GO
```

Indexes

```
DROP INDEX IF EXISTS RTD.Transporte.idx_transporte;
DROP INDEX IF EXISTS RTD.UnidadeTratamento.idx_unidade;
DROP INDEX IF EXISTS RTD.Quartel.idx_quartel;
DROP INDEX IF EXISTS RTD.Profissional.idx_profissional;
DROP INDEX IF EXISTS RTD.Bombeiro.idx_bombeiro;
DROP INDEX IF EXISTS RTD.Cuidador.idx_cuidador;
DROP INDEX IF EXISTS RTD.Paciente.idx_paciente;


CREATE INDEX idx_paciente       ON RTD.Paciente(cc);
CREATE INDEX idx_cuidador       ON RTD.Cuidador(cc_cuidador);
CREATE INDEX idx_bombeiro       ON RTD.Bombeiro(id_bombeiro);
CREATE INDEX idx_profissional   ON RTD.Profissional(cedula);
CREATE INDEX idx_quartel        ON RTD.Quartel(id_quartel);
CREATE INDEX idx_unidade        ON RTD.UnidadeTratamento(id_centro);
CREATE INDEX idx_transporte     ON RTD.Transporte(id_transporte);
```

Triggers

Foi criado um *trigger* que define que apenas os Profissionais cuja função de ‘Médico’ e cédula profissional válida é que têm autorização para marcar consultas a um paciente.

```
CREATE TRIGGER consulta_insert_trigger
BEFORE INSERT ON RTD.Consulta
FOR EACH ROW
BEGIN
    DECLARE prof_funcao VARCHAR(20);
    DECLARE prof_cedula INT;

    -- Get the professional's function and cedula
    SELECT funcao, cedula INTO prof_funcao, prof_cedula
    FROM RTD.Profissional
    WHERE cedula = NEW.cedula;

    -- Check if the professional is a "Médico"
    IF prof_funcao = 'Médico' THEN
        -- Check if the cedula matches
        IF prof_cedula = NEW.cedula THEN
            -- Allow the insert
            SET NEW.id_consulta = (SELECT COALESCE(MAX(id_consulta), 0) + 1 FROM RTD.Consulta);
        ELSE
            -- Raise an error if the cedula doesn't match
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid professional cedula for "Médico".';
        END IF;
    ELSE
        -- Raise an error if the professional is not a "Médico"
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Only "Médico" professionals can insert into "consulta".';
    END IF;
END;
```

Stored Procedures

As Procedures foram desenvolvidas para:

- Ver todos os pacientes
- Pesquisar pacientes
- Inserir paciente
- Remover paciente

Exemplos:

```
/*----- Ver todos os pacientes -----*/
create procedure [dbo].[getPacientes]
as
    begin
            select * from RTD.Paciente
    end
go

--exec dbo.getPacientes

/*------- Pesquisar Pacientes ------*/
create procedure [dbo].[searchPacientes](@num_utente INT)
as
    select * from RTD.Paciente where num_utente = @num_utente
go
```

UDFs

Foram criadas as seguintes UDFs:

- Dado um paciente devolve os diagnósticos dos eventos clínicos

```
DROP FUNCTION IF EXISTS getEventsByName;

-- Dado um paciente devolve os diagnosticos dos eventos clinicos
GO
CREATE FUNCTION getEventsByName (@num_utente INT) RETURNS TABLE AS
    RETURN (SELECT * FROM RTD.EventoClinico
                WHERE num_utente = @num_utente)
GO
```

- Devolver o ID do Quartel que um dado veículo (nomeadamente transporte de doentes não urgentes) pertence

```
-- Devolver o Quartel que um dado veiculo pertence
CREATE FUNCTION getTransport (@id_quartel INT) RETURNS TABLE AS
    RETURN (SELECT * FROM RTD.Transporte
                WHERE id_quartel = @id_quartel)
GO
```

- Devolver o Quartel que um dado Bombeiro pertence

```
-- Devolver o Quartel que um dado bombeiro pertence
CREATE FUNCTION getFirefighter (@id_quartel INT) RETURNS TABLE AS
     RETURN (SELECT * FROM RTD.Transporte
                 WHERE id_quartel = @id_quartel)
GO
```

- Devolver o nome de um Quartel

```
-- Devolver o nome de um quartel
CREATE FUNCTION getHeadquartersName (@nome VARCHAR(32)) RETURNS TABLE AS
    RETURN (SELECT * FROM RTD.Quartel
                WHERE nome = @nome)
GO
```

Interface

Ver vídeo da demonstração da aplicação em anexo.