Easy Intern

Bruno Pinto nº73156
André Gomes nº84811

Bases de Dados

10/06/2020

Licenciatura em Engenharia em Informática
Universidade de Aveiro

Índice

Proposta de Trabalho e Análise de Requisitos

A nossa proposta de trabalho consiste na criação de uma base de dados juntamente com a sua interface gráfica para uma rede de estágio de fisioterapia.

Os requisitos para esta base de dados desta rede de estágio são:

- A rede possui um staff composto por coordenadores, orientadores e estagiários, caracterizados pelo seu nif.
- Todo o staff possui passwords para terem acesso à plataforma consoante as suas permissões
- Orientadores e estagiários possuem horários de trabalhos.
- Orientadores são coordenados por um coordenador e os estagiários são orientados por orientadores
- O staff também possui um sistema de e-mails para comunicar entre si
- A plataforma também possui pacientes que visitam a rede de fisioterapia
- Cada paciente é caracterizado pelo nif, número de processo, descrição, altura , peso, a sua condição, o número de sessões a que já foi, o dia da primeira sessão e o plano que segue.
- O plano exercido por um determinado paciente deve conter o número total de sessões e o horário de cada sessão
- Cada sessão terá uma data, o plano que o paciente está a seguir, o estagiário que trata do paciente e o orientador que supervisiona a sessão.
- A sessão também deve seguir um tratamento para o paciente com o título e a descrição do mesmo.

Diagrama Entidade/Relação

Esquema Relacional

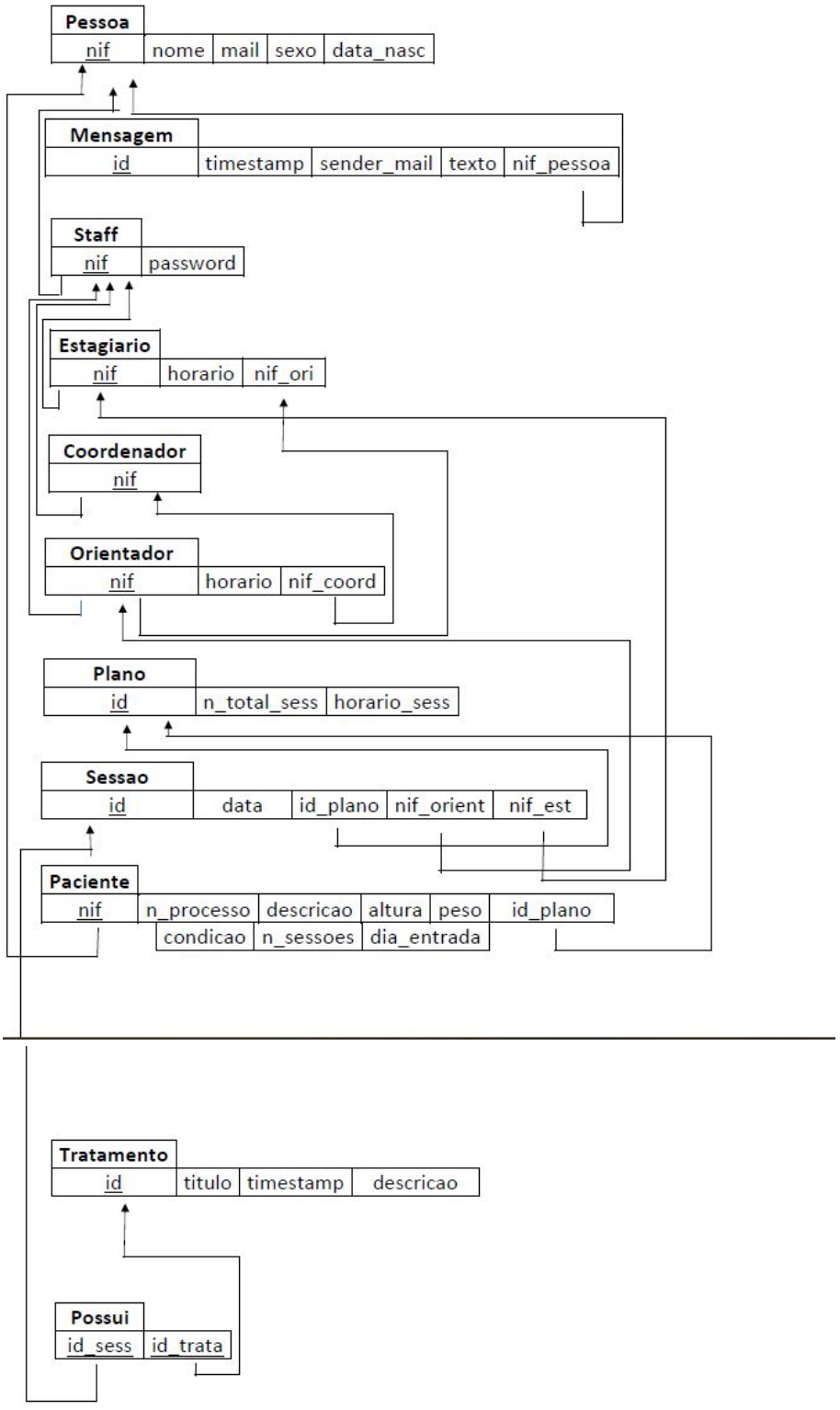

Figura 2 - Esquema Relacional

Criação de Tabelas

A primeira coisa a ser feita na nossa base de dados é a criação de um esquema a quem vamos associar as tabelas onde por sua vez iremos a seguir introduzir os nossos dados.

Todo o processo a partir deste momento(excluindo a criação e desenvolvimento da interface gráfica) será desenvolvido no Microsoft Sql Server Management Studio 18.

A criação das tabelas são definidos os atributos com os seus tipos de dados respectivos, respeitando algumas restrições de integridade.

Com a definição de chaves primárias , estrangeiras e usando constraints, conseguimos obter as ligações necessárias entre si.

Todo o código da criação das tabelas irá ser entregue juntamente com este relatório.

```
CREATE SCHEMA [fisioterapia];

create table [fisioterapia].Pessoa(
    nif         int             not null,
    nome        varchar(30)             ,
    mail        varchar(30)             ,
    sexo        char                    ,
    data_nasc   date
    primary key (nif)
    );

create table [fisioterapia].Mensagem(
    id          int             not null,
    timestamp   time                    ,
    sender_mail varchar(30)             ,
    texto       varchar(100)            ,
    nif_pessoa  int             not null
    primary key (id)
    );

create table [fisioterapia].Staff(
    nif             int             not null,
    password        varchar(15)             ,
    primary key (nif)
    );
```

Figura 3 - Exemplo de Criação de Esquema e Tabelas a ele Associado

```
ALTER TABLE [fisioterapia].Mensagem
ADD CONSTRAINT MENSPESS
FOREIGN KEY (nif_pessoa) REFERENCES [fisioterapia].Pessoa(nif);

ALTER TABLE [fisioterapia].Staff
ADD CONSTRAINT STAFFPESS
FOREIGN KEY (nif) REFERENCES [fisioterapia].Pessoa(nif);

ALTER TABLE [fisioterapia].Estagiario
ADD CONSTRAINT ESTSTAFF
FOREIGN KEY (nif) REFERENCES [fisioterapia].Staff(nif);
ALTER TABLE [fisioterapia].Estagiario
ADD CONSTRAINT ESTORI
FOREIGN KEY (nif_ori) REFERENCES [fisioterapia].Orientador(nif);
```

Figura 4 - Exemplo de Constraints criadas para referenciar chaves estrangeiras

Diagrama das Tabelas

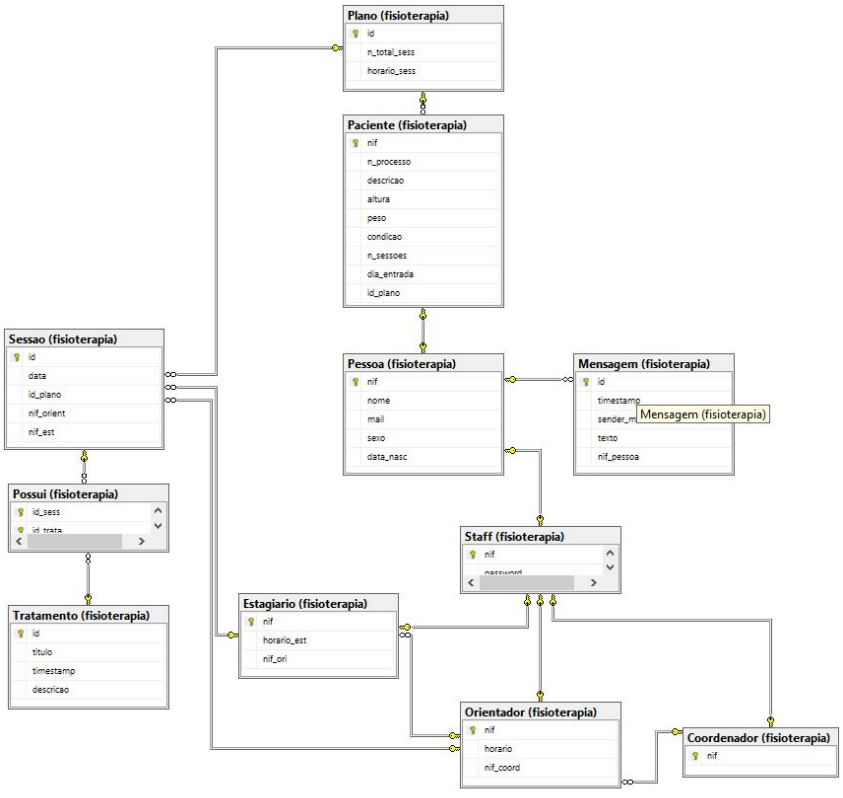

Figura 5 - Diagrama das Tabelas

Inserção de Valores

A inserção de valores foi feita através de de Inserts nas tabelas que foram criadas previamente.

O script usado com todas as inserções também irá ser entregue juntamente com este relatório.

```
-- VALUES FOR TABLE MENSAGEM
INSERT INTO [fisioterapia].Mensagem (id,timestamp,sender_mail, texto, nif_pessoa) VALUES
    (1, '2019-10-12 20:23:21', 'andregomes00@gmail.com', 'Nao esquecer marcar a consulta de quarta-feira!', 876598675),
    (2, '2020-04-12 17:00:00', 'pedrobastos10@gmail.com', 'Mudar plano de Rui Torres.', 345612345),
    (3, '2020-04-13 08:21:40', 'pereirita@gmail.com', 'Desmarcar a sessao de Joana Pires.', 489347234),
    (4, '2020-05-12 11:10:23', 'luiscarvalho@gmail.com', ' Reunir o staff na terca-feira.', 234527643);

SELECT * FROM [fisioterapia].Mensagem;
DELETE FROM [fisioterapia].Mensagem WHERE id=1;
```

Figura 6 - Exemplo de uma Inserção de Valores usando INSERT

Criação das UDF'S

Recorremos ao uso de udf´s para realizarmos várias pesquisas assim como validação de dados para o sistema de login. As udf´s usadas estão divididas em 2 tipos.

Funções de Valores Escalares

```
GO
CREATE FUNCTION [fisioterapia].UDF_LOGINORI(@nif int, @password varchar(15))
RETURNS varchar(20)
AS
BEGIN
    Declare @tipo varchar(20);
    IF( EXISTS (SELECT [fisioterapia].Orientador.nif, password
                FROM [fisioterapia].Staff JOIN [fisioterapia].Orientador ON [fisioterapia].Staff.nif=[fisioterapia].Orientador.nif
                WHERE @nif = [fisioterapia].Orientador.nif AND  @password= password))
        BEGIN
            SET @tipo ='Orientador'
        END
    ELSE
        BEGIN
            SET @tipo ='Nada'
        END
    RETURN @tipo;
END
GO
```

Figura 7 - Função de Validação de um Orientador

```
GO
CREATE FUNCTION [fisioterapia].UDF_LOGINCOOR(@nif int, @password varchar(15))
RETURNS varchar(20)
AS
BEGIN
    Declare @tipo varchar(20)
    IF( EXISTS (SELECT [fisioterapia].Coordenador.nif, password
                FROM [fisioterapia].Staff JOIN [fisioterapia].Coordenador ON [fisioterapia].Staff.nif=[fisioterapia].Coordenador.nif
                WHERE @nif = [fisioterapia].Coordenador.nif AND  @password= password))
        BEGIN
            SET @tipo ='Coordenador'
        END
    ELSE
        BEGIN
            SET @tipo ='Nada'
        END
    RETURN @tipo;
END
GO
```

Figura 8 - Função de Validação de um Coordenador

```
GO
CREATE FUNCTION [fisioterapia].UDF_LOGINEST(@nif int, @password varchar(15))
RETURNS varchar(20)
AS
BEGIN
    Declare @tipo varchar(20)
    IF( EXISTS (SELECT [fisioterapia].Estagiario.nif, password
                FROM [fisioterapia].Staff JOIN [fisioterapia].Estagiario ON [fisioterapia].Staff.nif=[fisioterapia].Estagiario.nif
                WHERE @nif = [fisioterapia].Estagiario.nif AND  @password= password))
        BEGIN
            SET @tipo ='Estagiario'
        END
    ELSE
        BEGIN
            SET @tipo ='Nada'
        END
    RETURN @tipo;
END
GO
```

Figura 9 - Função de Validação de um Estagiário

Funções de Valores de Tabela

Criação dos Stored Procedures

Foram usados procedures para para fazer updates de parâmetros nas tables.

```
Go
Create PROCEDURE [fisioterapia].ALTMAIL(@nif int,
                                        @mail varchar(30)
                                        )
AS
BEGIN
    BEGIN
        UPDATE [fisioterapia].Pessoa
            SET   mail  =@mail
        WHERE nif = @nif
    END

END
Go
```

Figura X - Procedimento usado para alterar o e-mail de uma pessoa

```
Go
Create PROCEDURE [fisioterapia].ALTORIEST(@nifest int,
                                          @nif_ori int
                                          )
AS
BEGIN
    BEGIN
        UPDATE [fisioterapia].Estagiario
            SET nif = @nifest,
                nif_ori = @nif_ori
            WHERE nif=@nifest
    END
END
GO
```

Figura X - Procedimento de alteração de orientadores encarregues de estagiários

```
Go
Create PROCEDURE [fisioterapia].MAILSEND(@nif_pessoa int,
                                         @sender_mail varchar(30),
                                         @texto varchar(100)
                                         )
AS
BEGIN
    BEGIN
        INSERT INTO [fisioterapia].Mensagem (nif_pessoa, timestamp, sender_mail, texto)
                                    VALUES  (@nif_pessoa,CURRENT_TIMESTAMP, @sender_mail, @texto)
    END
END
GO
```

Figura X - Procedimento para envio de uma mensagem

Criação dos Triggers

Apenas foi criado um trigger para salvaguardar toda a informação relativa a dados de todas as entidades pessoais caso a tabela Pessoa seja apagada. Este trigger irá criar uma tabela backup com toda a informação que caso contrário iria ser perdida.

```
--- TRIGGER DE BACKUP DE TODA INFORMACAO SOBRE PESSOAS
Go
CREATE TRIGGER BACKUP_PESSOA ON [fisioterapia].Pessoa INSTEAD OF DELETE AS
BEGIN
    IF  NOT (EXISTS (
                SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = 'fisioterapia' AND TABLE_NAME = 'PESSOA_BACKUP'))
                CREATE TABLE [fisioterapia].PESSOA_BACKUP (
                                                        nif INT NOT NULL,
                                                        nome INT NOT NULL,
                                                        mail VARCHAR(50) NOT NULL,
                                                        sexo char NOT NULL,
                                                        data_nasc date NOT NULL)
    INSERT INTO [fisioterapia].PESSOA_BACKUP SELECT * FROM DELETED
    Delete FROM [fisioterapia].Pessoa Where [fisioterapia].Pessoa.nif=( SELECT nif FROM DELETED)
END
Go
```

Figura X - Exemplo da Criação do trigger de Backup da Tabela Pessoa

Criação da Interface Gráfica

Tarefas Não Implementadas

Após termos debatido , chegamos à conclusão que as nossas tabelas não necessitavam de passar pelo processo de normalização. Apenas tivemos dúvidas na tabela Sessão, onde acabámos por concordar que não iríamos aplicar normalização.

Também não houve implementação de índices devido à base de dados não ser extensa o suficiente para se notar diferença na sua performance.

Para além do único trigger que temos, gostaríamos de ter implementado mais dois ou três triggers para satisfazer as necessidades do projeto.

Também tentamos implementar um procedure para atualizar um estagiário encarregue de um funcionário, mas não a conseguimos pôr a funcionar.

Conclusão

Com a realização de deste trabalho , conseguimos explorar todas as etapas do que é a concepção de uma base de dados e o seu desenvolvimento do início ao fim.

Apesar de termos tido algumas dificuldades, penso que conseguimos alcançar grande parte dos objetivos e tarefas propostas, obtendo assim no final um produto bem formado e coerente que interage bem com o utilizador.

Com a interface criada em C# e usando o Microsoft Sql Server Management Studio 18 para trabalhar com a parte SQL do trabalho conseguimos notar uma certa harmonia entre as duas componentes.

Bibliografia

- Slides Teóricos de BD
- Guiões Práticos de BD