Top5 Futebol App

André Marques, 47468
Lucas Seabra, 80323
P8G7

Análise de Requisitos

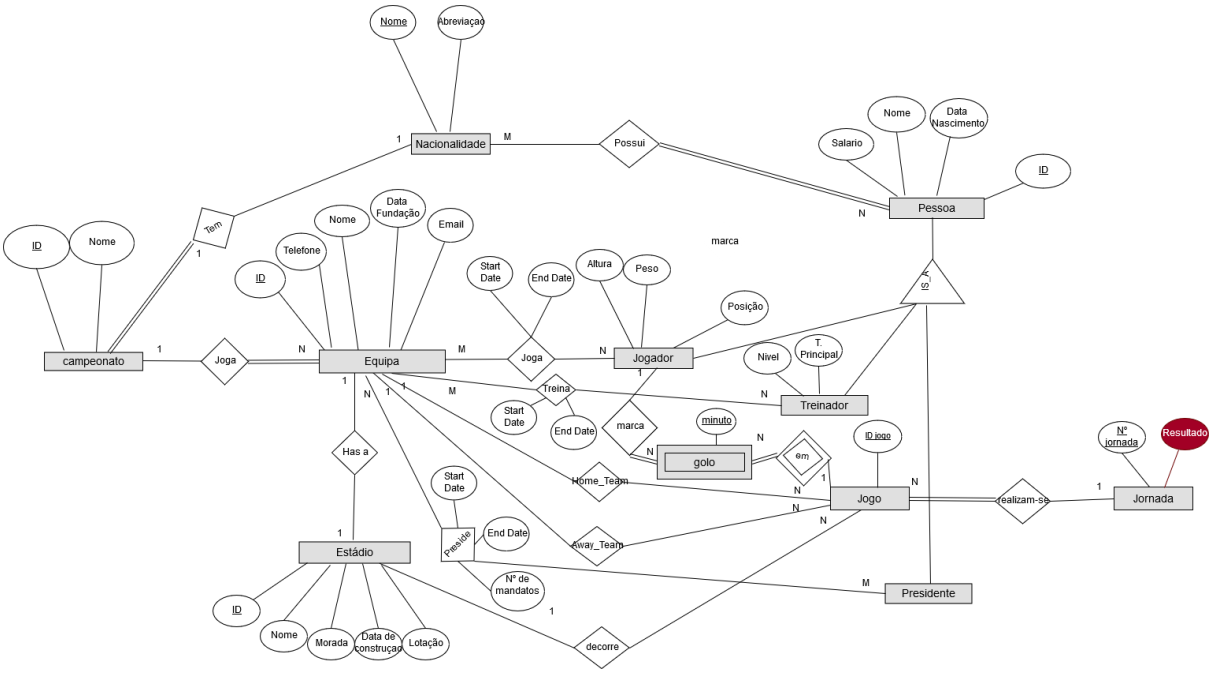
A nossa aplicação permite consultar, no contexto de ligas de futebol, os melhores marcadores, visualizar as equipas que pertencem a uma determinada liga, visualizar os resultados dos jogos decorridos, os pontos de cada equipa, para além de poder consultar alguns atributos sobre os jogadores, treinadores, estádio e presidente.

Para tal o sistema possui as seguintes entidades:

- ➔ Um campeonato tem como atributos um nome e um ID de identificação
- ➔ Uma equipa é identificada pelo seu nome, e é possível verificar a data em que foi fundada, o seu telefone, email, classificação, golos marcados e sofridos, para além do número de vitórias, empates e derrotas
- ➔ Uma pessoa é identificada pelo seu ID, e tem um nome e uma data de nascimento
- ➔ Um jogador, sub-entidade de pessoa, tem altura, peso, posição, ordenado, golos marcados e assistências
- ➔ O presidente, sub-entidade de pessoa, tem o número de mandatos
- ➔ Do treinador é possível saber o seu ordenado, e pode ser especializado em principal ou adjunto
- ➔ O estádio é caracterizado pelo seu nome e é possível consultar a sua lotação, morada e data de construção
- ➔ A nacionalidade é caracterizado pelo nome e possui também uma abreviatura

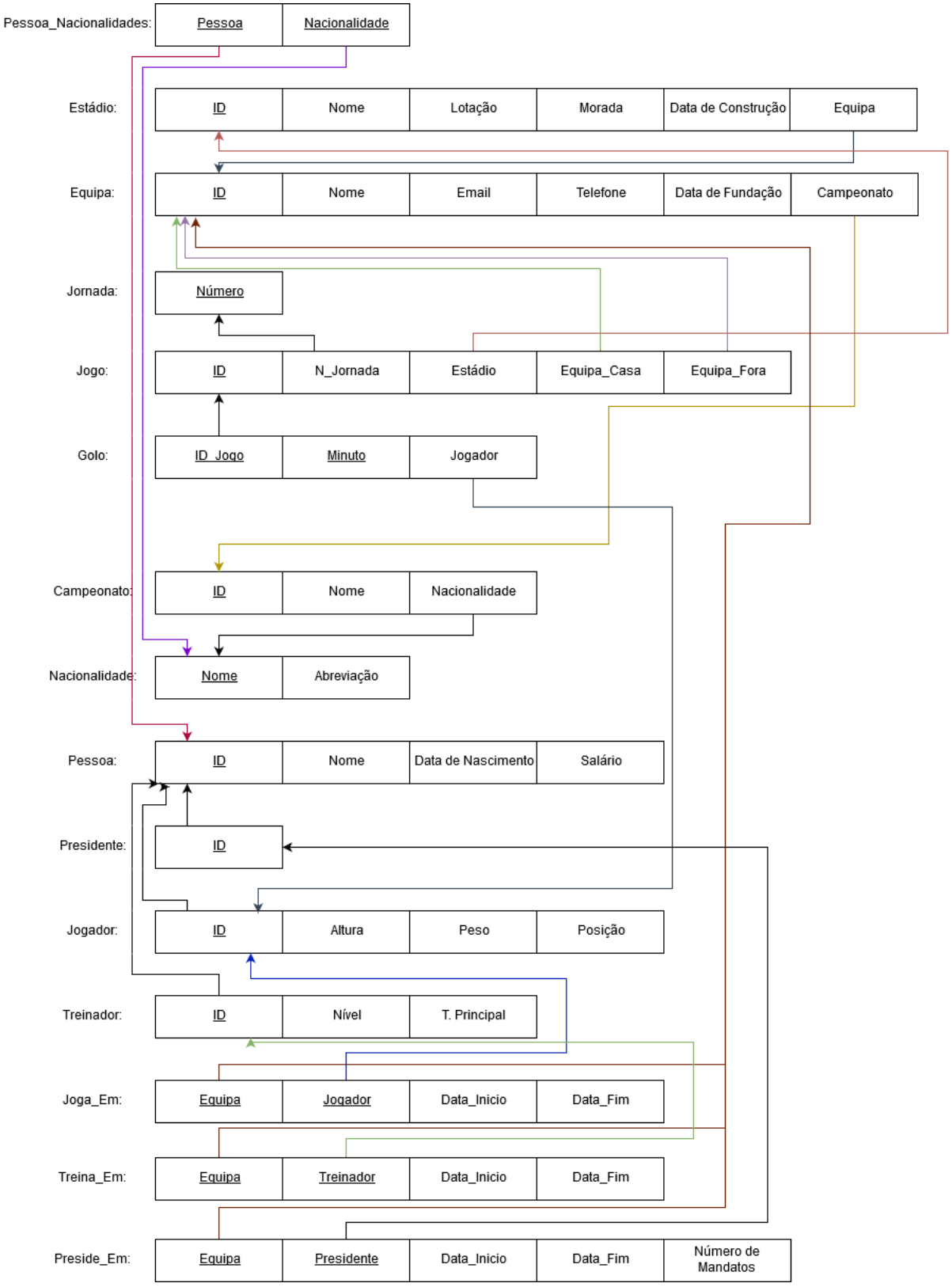
É de notar que um campeonato tem uma nacionalidade, e uma pessoa tem uma ou mais nacionalidades. Um campeonato tem várias equipas no qual jogam, e esta última tem um estádio, tem vários jogadores, para além de treinador principal, adjunto e presidente.

Diagrama Entidade-Relação



Usando a análise de requisitos chegámos a este diagrama entidade-relação.
De seguida, usando os métodos aprendidos nas aulas, realizou-se a transformação do diagrama entidade-relação para esquema relacional.
É de notar a forma como os golos estão modelados: um golo é marcado por um jogador num jogo, a um determinado minuto do mesmo, e o jogador joga numa equipa. É usando estas relações que conseguimos obter os resultados dos jogos e as pontuações das equipas, como será visto mais à frente.

Esquema Relacional



Organização dos ficheiros

A base de dados está organizada da seguinte forma:

- SQL_DDL.sql - crias as tabelas da base de dados
- Triggers.sql - cria e aplica os triggers às tabelas
- UDF.sql - cria as “user defined functions”
- StoredProcedures.sql - cria as stored procedures
- SQL_DML.sql - usa comandos sql e stored procedures para inserir dados
- comandos.sql - tem o mesmo propósido da SQL_DML.sql, mas é criado por um script

Data Definition Language

A DDL foi obtida através do esquema relacional.
O único ponto a reparar é o uso de `IDENTITY(1,1),` para que os ids das tabelas sejam criados automaticamente onde desejável, exemplificado de seguida:

```
CREATE TABLE gestao_futebol.campeonato(
    id                  INT NOT NULL IDENTITY(1,1),
    nome                VARCHAR(30) NOT NULL,
    nacionalidade       VARCHAR(50) NOT NULL
    PRIMARY KEY(id),
    FOREIGN KEY(nacionalidade) REFERENCES
gestao_futebol.nacionalidade(nome)
)

CREATE TABLE gestao_futebol.equipa(
    id                  INT NOT NULL IDENTITY(1,1),
    nome                VARCHAR(100) NOT NULL,
    email               VARCHAR(30),
    telefone            INT,
    data_fund           DATE NOT NULL,
    campeonato          INT NOT NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(campeonato) REFERENCES gestao_futebol.campeonato(id)
)

CREATE TABLE gestao_futebol.estadio(
    id                INT NOT NULL IDENTITY(1,1),
    nome              VARCHAR(100) NOT NULL,
```

```
    lotacao           INT NOT NULL,
    morada          VARCHAR(200),
    data_crons      DATE NOT NULL,
    equipa               INT NOT NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(equipa) REFERENCES gestao_futebol.equipa(id)
)
```

User Defined Functions

Existem duas UDFs, uma para obter uma tabela que relaciona cada jogo com as equipas participantes e os golos marcados pelas equipas que jogam em casa, e outra com os marcados fora. Ambas são consumidas por uma stored procedure, e foram divididas desta forma para uma maior organização e compreensão.

```
CREATE FUNCTION gestao_futebol.TabelaGolosPorJogoHome (

)
RETURNS TABLE
AS
RETURN

SELECT *, ISNULL(h_score_null, 0 ) as h_score_c
FROM gestao_futebol.jogo as jj left outer JOIN

    (SELECT  f.h_team as f_h_team, f.a_team as f_a_team, id_jogo,
COUNT(*) as h_score_null
    FROM gestao_futebol.TabelaGolosPorJogo() as f JOIN
gestao_futebol.jogo as j2 ON f.id=j2.id
    where team=f.h_team
    GROUP BY f.h_team, f.a_team, id_jogo) as comboio

ON comboio.id_jogo = jj.id
GO



CREATE FUNCTION gestao_futebol.TabelaGolosPorJogoAway (
```

```
)
RETURNS TABLE
AS
RETURN



SELECT *, ISNULL(a_score_null, 0 ) as a_score_c
FROM gestao_futebol.jogo as jj left outer JOIN

    (SELECT  f.h_team as f_h_team, f.a_team as f_a_team, id_jogo,
COUNT(*) as a_score_null


    FROM gestao_futebol.TabelaGolosPorJogo() as f JOIN
gestao_futebol.jogo as j2 ON f.id=j2.id
    where team=f.a_team
    GROUP BY f.h_team, f.a_team, id_jogo) as comboio

ON comboio.id_jogo = jj.id


GO
```

Stored Procedures

O nosso projeto contém stored procedures com vários objetivos: inserir dados, garantindo que são válidos e inseridos corretamente, para editar dados, apagar e para inserir dados de forma automatizada. De seguida alguns exemplos:

```
CREATE PROC gestao_futebol.TabelaResultados AS
CREATE TABLE #resultados(
    id       INT,
    h_team    INT,
    a_team    INT,
    h_score    INT,
    a_score    INT
```

```
)

INSERT INTO #resultados (id)
SELECT id FROM gestao_futebol.jogo

UPDATE #resultados
SET #resultados.h_team = jogo.h_team
FROM #resultados
JOIN gestao_futebol.jogo as jogo
ON  #resultados.id = jogo.id

UPDATE #resultados
SET #resultados.a_team = jogo.a_team
FROM #resultados
JOIN gestao_futebol.jogo as jogo
ON  #resultados.id = jogo.id

UPDATE #resultados
SET #resultados.h_score = h.h_score_c
FROM #resultados
JOIN gestao_futebol.TabelaGolosPorJogoHome() as h
ON  h.id_jogo = #resultados.id

UPDATE #resultados
SET #resultados.a_score = h.a_score_c
FROM #resultados
JOIN gestao_futebol.TabelaGolosPorJogoAway() as h
ON  h.id_jogo = #resultados.id


SELECT #resultados.id, h_team, a_team, ISNULL(h_score, 0 ) as h_score,
ISNULL(a_score, 0 ) as a_score, equipa.nome as h_name, equipa2.nome as
a_name
FROM #resultados JOIN gestao_futebol.equipa as equipa on equipa.id =
h_team
JOIN gestao_futebol.equipa as equipa2 on equipa2.id = a_team


GO
```

Esta stored procedure utiliza as duas UDFs mencionadas anteriormente para obter uma tabela com o id do jogo, ids das equipas, os seus nomes e resultado do jogo. Esta SP foi especialmente complexa, e as abordagens anteriores a este resultado final foram

fracassadas. Isto porque ao contar o número de golos por equipa por jogo descartava os jogos onde uma das equipas não marcasse golo.
O que parecia uma solução simples, fazendo right join com a tabela de jogos, também se revelou um fracasso, por motivos inesperados: ao fazer SELECT de uma tabela com campos a *null* estes simplesmentes desapareciam, mesmo que no SELECT estivessem presentes todos os campos. Assim, voltou-se ao problema inicial.
A solução que acabou por funcionar foi uma mais modular, em que é criada uma tabela com os campos finais desejados, id do jogo, id da equipa que joga em casa e fora, e o resultado das mesmas.

Inicialmente são copiadas as colunas de id, e das equipas participantes. De seguida é utilizada a UDF *TabelaGolosPorJogoHome* para obter os resultados da equipa que joga em casa, e a *TabelaGolosPorJogoAway* para obter os resultados da equipa a jogar fora. Estes resultados são copiados para a tabela criada inicialmente.

Com esta abordagem, a tabela mantém os resultados a *null*, e assim é apenas necessário substituir estes pelo valor 0, obtendo assim os resultados dos jogos entre as equipas.

```
CREATE PROC gestao_futebol.CriaJogadorSimples (
    @nome               VARCHAR(100),
    @data_nasc     VARCHAR(50),
    @altura                  INT,
    @peso                    INT,
    @posicao       VARCHAR(100),
    @salario       VARCHAR(50)
) AS

SET NOCOUNT ON;

INSERT INTO gestao_futebol.pessoa([nome],[data_nasc],[salario])
VALUES(@nome,@data_nasc,@salario);

DECLARE @IncID int
SET @IncID = SCOPE_IDENTITY();



INSERT INTO gestao_futebol.jogador([id_jogador], [altura], [peso],
[posicao]) VALUES (@IncID, @altura, @peso, @posicao);

SET NOCOUNT OFF
GO
```

A stored procedure CriaJogadorSimples é utilizada, como o nome indica, para adicionar um jogador. Para isso, primeiro é criada uma pessoa, e de seguida, usando o id da pessoa que foi criada, adiciona-se também um registo na tabela de jogadores.

```
CREATE PROC gestao_futebol.GetPoints AS

CREATE TABLE #resultados(
    id       INT,
    h_team    INT,
    a_team    INT,
    h_score    INT,
    a_score    INT
)

INSERT INTO #resultados (id)
SELECT id FROM gestao_futebol.jogo

UPDATE #resultados
SET #resultados.h_team = jogo.h_team
FROM #resultados
JOIN gestao_futebol.jogo as jogo
ON  #resultados.id = jogo.id

UPDATE #resultados
SET #resultados.a_team = jogo.a_team
FROM #resultados
JOIN gestao_futebol.jogo as jogo
ON  #resultados.id = jogo.id

UPDATE #resultados
SET #resultados.h_score = h.h_score_c
FROM #resultados
JOIN gestao_futebol.TabelaGolosPorJogoHome() as h
ON  h.id_jogo = #resultados.id

UPDATE #resultados
SET #resultados.a_score = h.a_score_c
FROM #resultados
JOIN gestao_futebol.TabelaGolosPorJogoAway() as h
ON  h.id_jogo = #resultados.id
```

```
SELECT #resultados.id, h_team, a_team, ISNULL(h_score, 0 ) as h_score,
ISNULL(a_score, 0 ) as a_score, equipa.nome as h_name, equipa2.nome as
a_name INTO #res
--SELECT *
FROM #resultados JOIN gestao_futebol.equipa as equipa on equipa.id =
h_team
JOIN gestao_futebol.equipa as equipa2 on equipa2.id = a_team

SELECT equipa.id, equipa.nome , sum
    (
       case
             when equipa.id=h_team and h_score > a_score then 3
             when equipa.id=a_team and h_score < a_score then 3
             when equipa.id=h_team and h_score = a_score then 1
             when equipa.id=a_team and h_score = a_score then 1
             else 0
       end
    ) pontos
FROM #res as r
JOIN gestao_futebol.equipa as equipa ON equipa.id=r.h_team OR
equipa.id=r.a_team
GROUP BY equipa.id, equipa.nome
ORDER BY pontos DESC

GO
```

A SP GetPoints devolve uma tabela com os pontos de cada equipa, usando os resultados dos jogos de cada equipa e comparando o número de golos marcados por cada uma.

```
CREATE PROC gestao_futebol.GetTeamInfo AS

SELECT equipa.id, equipa.nome, equipa.email, equipa.data_fund, camp.nome
as campeonato, pessoa.nome as presidente, est.nome as estadio
FROM gestao_futebol.equipa as equipa JOIN gestao_futebol.campeonato as
camp ON equipa.campeonato = camp.id
JOIN gestao_futebol.preside_em as preside_em ON preside_em.team =
equipa.id
JOIN gestao_futebol.pessoa as pessoa ON pessoa.id = preside_em.president
JOIN gestao_futebol.estadio as est ON est.equipa = equipa.id

GO
```

```
CREATE PROC gestao_futebol.GetPlayerInfo AS

SELECT id, nome, data_nasc, salario, altura, peso, posicao
FROM gestao_futebol.pessoa as pessoa JOIN gestao_futebol.jogador as
jogaror ON pessoa.id = id_jogador

GO
```

A GetTeamInfo e GetPlayerInfo devolvem os dados das equipas e dos jogadores prontos a serem consumidos pela interface gráfica.

```
CREATE PROC gestao_futebol.EditaJogador (
    @id                 INT,
    @nome               VARCHAR(100),
    @data_nasc      VARCHAR(50),
    @altura                 INT,
    @peso                   INT,
    @posicao        VARCHAR(100),
    @salario        VARCHAR(50)
) AS

SET NOCOUNT ON;


UPDATE gestao_futebol.pessoa
SET gestao_futebol.pessoa.nome = @nome
FROM gestao_futebol.pessoa
WHERE gestao_futebol.pessoa.id = @id


UPDATE gestao_futebol.pessoa
SET gestao_futebol.pessoa.data_nasc = @data_nasc
FROM gestao_futebol.pessoa
WHERE gestao_futebol.pessoa.id = @id

UPDATE gestao_futebol.pessoa
SET gestao_futebol.pessoa.salario = @salario
FROM gestao_futebol.pessoa
WHERE gestao_futebol.pessoa.id = @id
```

```
    --
UPDATE gestao_futebol.jogador
SET gestao_futebol.jogador.altura = @altura
FROM gestao_futebol.jogador
WHERE gestao_futebol.jogador.id_jogador = @id

    --
UPDATE gestao_futebol.jogador
SET gestao_futebol.jogador.peso = @peso
FROM gestao_futebol.jogador
WHERE gestao_futebol.jogador.id_jogador = @id

    --
UPDATE gestao_futebol.jogador
SET gestao_futebol.jogador.posicao= @posicao
FROM gestao_futebol.jogador
WHERE gestao_futebol.jogador.id_jogador = @id


SET NOCOUNT OFF
GO
```

A EditaJogadpr edita os dados de um jogador de futebol, incluindo os seus atributos enquanto pessoa.

Data Manipulation Language

Para preencher a base de dados foi usado um dataset contendo jogadores e as suas equipas. Para isso, criámos algumas stored procedures para agilizar essa tarefa, de forma a que o dataset, formatado em CSV, pudesse ser usado diretamente.

Como exemplo, esta é uma das linhas do ficheiro:

```
Iker Casillas Fernández,FC Porto,Portuguese Primeira
Liga,1981-05-20,185.0,84.0,Spain,3500000.0
```

Esta linha é facilmente transformada na seguinte:

```
EXEC gestao_futebol.CriaJogador 'Iker Casillas
Fernández','Porto','Primeira
Liga','1981-05-20',185.0,84.0,'Spain',3500000.0, '2018-01-01',
'2020-01-01'
```

Desta forma, foi criada uma stored procedure que adiciona uma pessoa, com os seus campos (nome, data de nascimento e salário) e usando o mesmo id adiciona-a também na tabela de jogadores. De seguida, com o nome da competição e da equipa onde joga obtém os respetivos ids e adiciona-os nas tabelas de relacionamento com equipas e campeonatos.

```
CREATE PROC gestao_futebol.CriaJogador (
    @nome               VARCHAR(100),
    @clube              VARCHAR(100),
    @liga               VARCHAR(100),
    @data_nasc     VARCHAR(50),
    @altura                 INT,
    @peso                   INT,
    @nacionalidade    VARCHAR(100),
    @salario       VARCHAR(50),
    @data_ini     DATE,
    @data_fim     DATE
) AS

SET NOCOUNT ON;

INSERT INTO gestao_futebol.pessoa([nome],[data_nasc],[salario])
VALUES(@nome,@data_nasc,@salario);

DECLARE @IncID int
DECLARE @ClubID int
SET @IncID = SCOPE_IDENTITY();



IF NOT EXISTS (SELECT * FROM gestao_futebol.nacionalidade WHERE [nome] =
@nacionalidade)
    SELECT @nacionalidade;


INSERT INTO gestao_futebol.pessoa_nacionalidade(pessoa, nacionalidade)
VALUES (@IncID, @nacionalidade);

INSERT INTO gestao_futebol.jogador([id_jogador], [altura], [peso])
VALUES (@IncID, @altura, @peso);

SELECT @ClubID = id FROM [top_5_futebol].[gestao_futebol].[equipa] where
nome=@clube;
```

```
INSERT INTO gestao_futebol.joga_em([team],[player], [data_ini],
[data_fim]) VALUES (@ClubID ,@IncID,@data_ini, @data_fim);
SET NOCOUNT OFF
GO
```

Para prencher a tabela de golos foi necessária uma abordagem mais criativa. Usando um dataset com o esquema

```
10/08/2018,Benfica,Vitoria de Guimaraes,
```

é possível obter os resultados das equipas, no entanto a nossa base de dados não suporta esta estrutura. Assim, usando o dataset anterior, escrevemos um script em *Python* que cria um jogo na base de dados, e de seguida insere o número de golos de acordo com o dataset usado, com um minuto escolhido de forma aleatória, e um jogador da equipa escolhido da mesma forma. Ambas as operações são realizadas com stored procedures, e tanto o script como os dataset estão presentes na submissão do trabalho. O script cria um ficheiro com o nome *comandos.sql*, e na submissão do trabalho também é enviado, junto com o script que o cria.

Conclusão

Com este trabalho conseguimos atingir os objetivos que considerámos mais importantes, nomeadamente aquilo que consideramos ser uma boa modulação da base de dados, que conseguimos atingir usando as ferramentas aprendidas aprendidas nesta disciplina, e a capacidade de, usando queries, user defined functions e stored procedures, conseguir adicionar, editar, remover e consultar os dados presentes na base de dados de forma escalável e eficiente. Infelizmente não tivemos a oportunidade de realizar tudo aquilo a que nos auto propusemos com este trabalho, pois gostaríamos de ter adicionado mais algumas funcionalidades para garantir uma melhor integridade de dados, e de ter melhorado a interface gráfica.
Apesar disso, estamos contentes com o trabalho realizado, pois para além dos objetivos que conseguimos alcançar, aprendemos imenso com a sua realização, e sabemos que no próximo projeto que envolver bases de dados teremos mais facilidade e mais conhecimentos para o realizar de uma forma mais correta e eficiente.