universidade de aveiro

Departamento de Eletrónica, Telecomunicações e Informática

Trabalho Prático Final

Desenvolvimento de BD e camada interação

Autores:

Nome: Diogo Filipe Amaral Carvalho	**Nº**92969	**Participação:** 1/2
Nome: Pedro Miguel Loureiro Amaral	**Nº**93283	**Participação:** 1/2

Curso: Engenharia Informática

Disciplina: Base de Dados

Aveiro
2020

1. Introdução

O presente relatório visa descrever e complementar o trabalho realizado ao longo do semestre no desenvolvimento do trabalho prático final associado à disciplina de Base de Dados. Tem como principal objetivo clarificação da metodologia utilizado e apresentação dos resultados obtidos ao longo das diversas etapas deste projeto.

2. Conceito

Este projeto aponta para o desenvolvimento de uma Base de Dados que fornece o suporte a um jogo fictício com o nome de "Fantasy League".

Utilizadores de "Fantasy League" têm a capacidade de formar uma equipa fictícia de 11 jogadores, que são na realidade, jogadores reais, isto é, são os jogadores que atuam nas equipas reais dos campeonatos de futebol atuais suportados pela BD. Os utilizadores podem também criar ou aderir a ligas fictícias, onde disputam com outros utilizadores por quem consegue ter uma melhor equipa. Esta avaliação de melhor equipa é baseada numa classificação por pontos acumulados.

Sempre que é realizado um jogo na vida real, consoante a prestação do jogador, este obtêm uma classificação, que é adicionada a cada equipa fictícia que continha este mesmo jogador.

3. Objetivos

Tendo por base o conceito descrito anteriormente, os principais objetivos deste projeto são:

- Desenvolvimento de uma Base de Dados capaz de dar suporte a este jogo, isto é, permita armazenamento de informação relativa aos campeonatos de futebol, equipas reais que atuam nesses mesmos campeonatos, jogadores que pertencem a estas equipas, estádios, jogos, utilizadores do jogo "Fantasy League", equipas fictícias e ligas fictícias criadas por estes mesmos utilizadores.
- Desenvolvimento de uma Interface que permita inserção, atualização, pesquisa e eliminação de dados da Base de Dados de forma segura e simples.

4. Análise de Requisitos

O sistema deve ter por base as seguintes características:

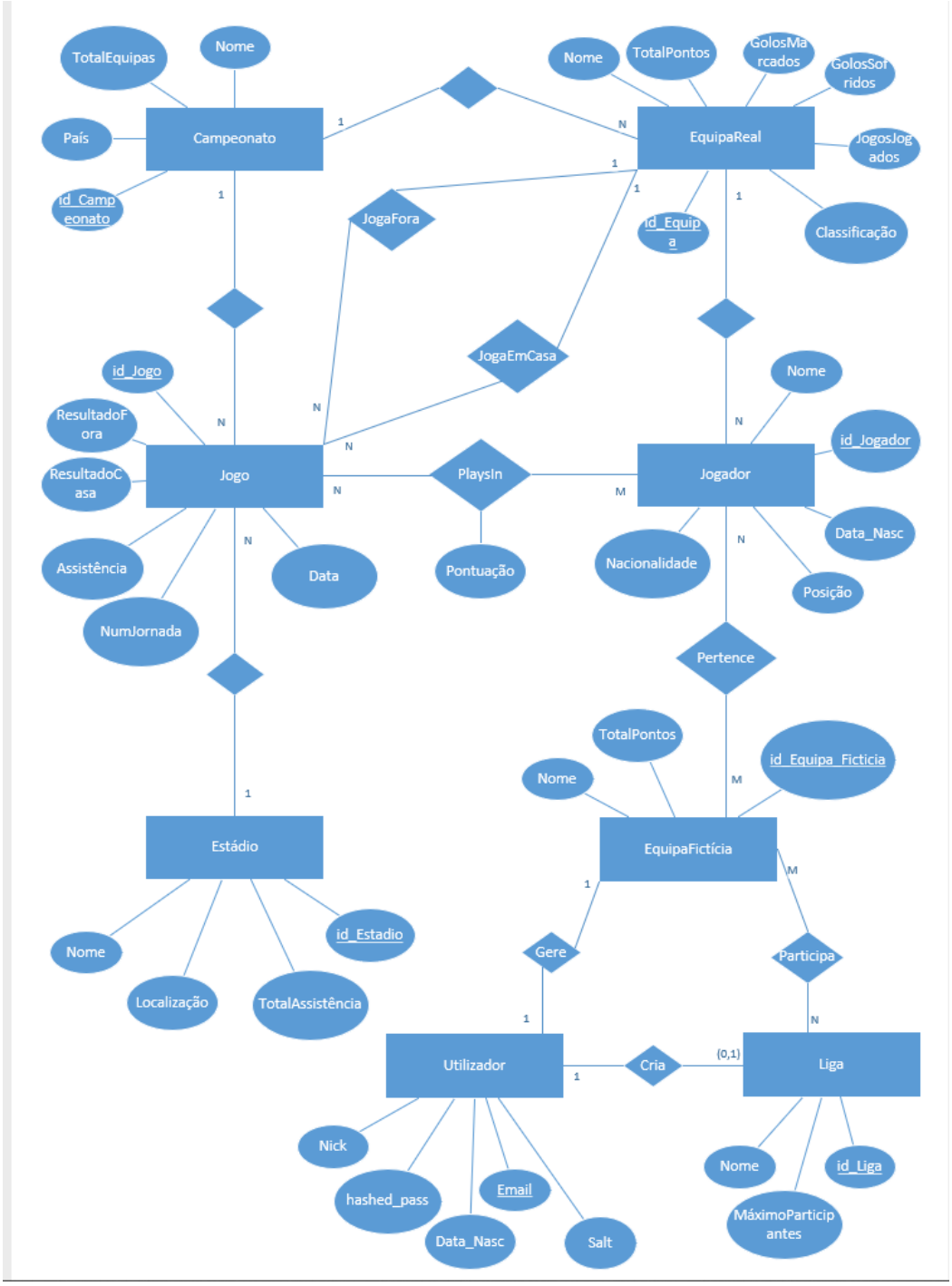
1. Um **campeonato** deve ter um nome, um país de origem, e um total de equipas participantes.
2. Um **Jogo** deve pertencer a um **campeonato**, ter duas **equipas reais**, realizar-se num determinado **estádio**, ter um resultado, uma data realização e o número de assistência.
3. Um **Estádio** contém um nome, um número máximo de assistência e uma localização.
4. Um **Jogador** deve possuir um nome, uma nacionalidade, uma posição e idade.

   Deve pertencer a:
   - Uma **Equipa Real** que é composta por jogadores, participa num **campeonato**, compete em **jogos**, tem um nome, uma classificação, um número de pontos e golos marcados e sofridos. Cada **jogador** deve receber uma pontuação pela sua prestação em cada **jogo**.
   - N **Equipas Fictícias** que também são compostas por jogadores mas participam em **Ligas** fictícias. Cada Equipa Fictícia é comandada por um **utilizador** e possuí um nome e um número de pontos total.
5. Uma **Liga** é composta por **Equipas Fictícias**, possuí um nome e um número máximo de participantes.
6. Um **Utilizador** gere uma **Equipa Fictícia,** possuí um nick, uma password, um email e uma idade.

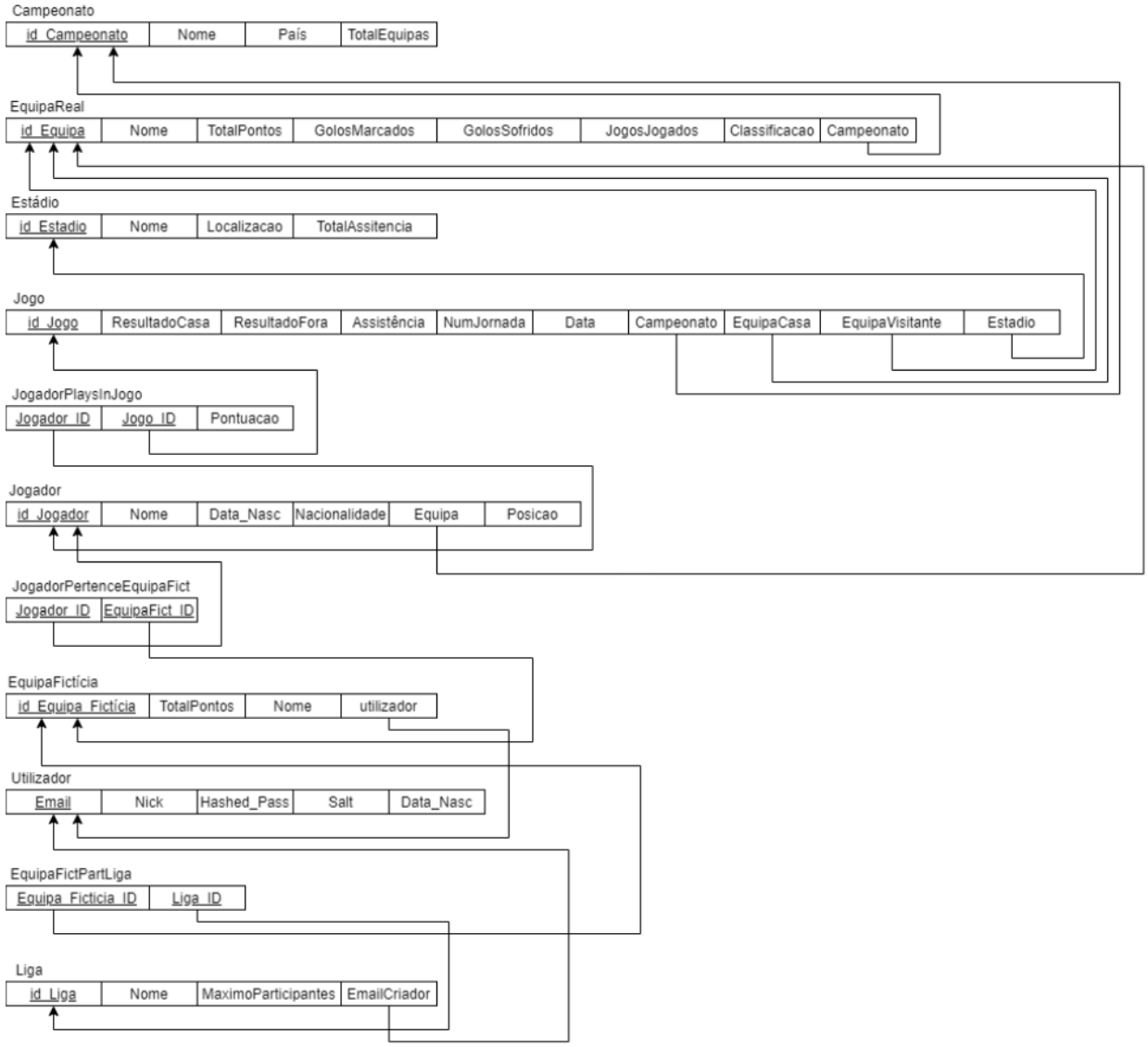
O sistema deve ainda respeitar as seguintes restrições de integridade:

1. Em cada **Jogo** não podem jogar mais de 14 **Jogadores** por equipa.
2. A **Pontuação** de um jogador num **Jogo** deve pertencer ao intervalo de 0 a 10.
3. O **Total** de **Pontos** de uma **Equipa Real** deve ser igual ou inferior a 3 vezes o **Número** de **Jogos** e igual ou superior a 0.
4. Uma **Liga** não pode ter mais **Equipas Fictícias** a participar nela do que o seu **Máximo Participantes**.
5. Um **Campeonato** não pode ter um número de **Equipas Reais** superior ao seu **Total Equipas**.
6. Num **Jogo**, a **Equipa Real** que joga fora tem de ser diferente da **Equipa Real** que joga em casa.
7. Uma **Equipa Fictícia** deve ser constituída por, no máximo, 11 **jogadores**.

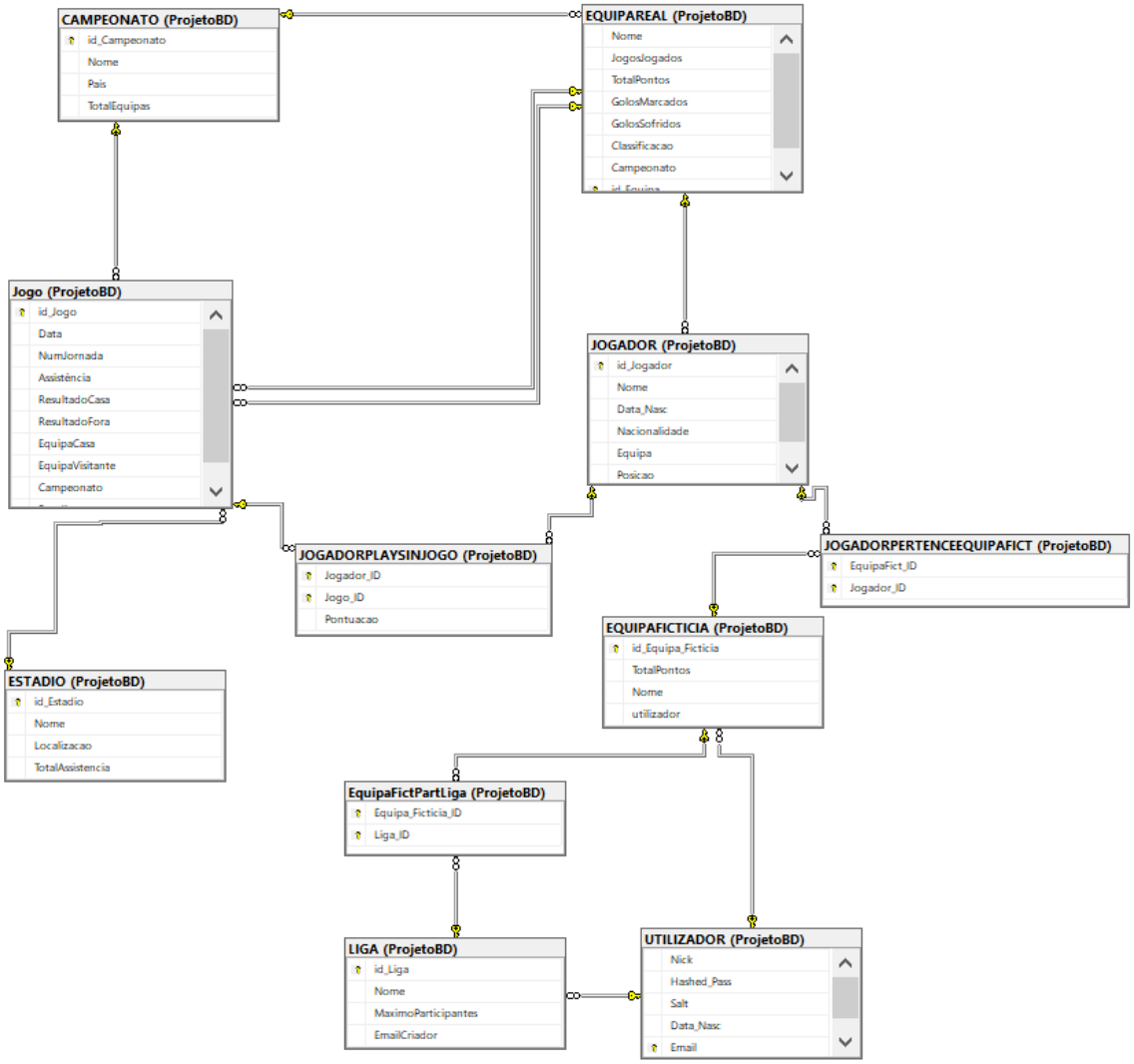
5. Diagrama de Entidade-Relação



6. Esquema Relacional da BD



7. Diagrama da Base de Dados



8. SQL DDL Estrutura da BD

```
CREATE TABLE ProjetoBD.CAMPEONATO (
    id_Campeonato        INT                NOT NULL IDENTITY(1,1),
    Nome                 VARCHAR(30)        NOT NULL,
    País                 VARCHAR(15)        NOT NULL,
    TotalEquipas         INT,
    PRIMARY KEY (id_Campeonato),
);

CREATE TABLE ProjetoBD.ESTADIO (
    id_Estadio                  INT         NOT NULL IDENTITY(1,1),
    Nome                 VARCHAR(55)        NOT NULL,
    Localizacao          VARCHAR(25)        NOT NULL,
    TotalAssistencia     INT                CHECK (TotalAssistencia>=0),
    PRIMARY KEY (id_Estadio)
);

CREATE TABLE ProjetoBD.EQUIPAREAL (
    id_Equipa                   INT                NOT NULL IDENTITY(1,1),
    Nome                        VARCHAR(20)        NOT NULL,
    JogosJogados         INT                CHECK(JogosJogados >= 0) Default(0),
    TotalPontos                 INT                Default(0),
    GolosMarcados        INT                CHECK(GolosMarcados >=0) Default(0),
    GolosSofridos        INT                CHECK(GolosSofridos >=0) Default(0),
    Campeonato                  INT,
    Classificacao        INT,
    PRIMARY KEY (id_Equipa),
    FOREIGN KEY (Campeonato) REFERENCES ProjetoBD.CAMPEONATO(id_Campeonato),
    CHECK (TotalPontos >=0 AND TotalPontos <= 3*JogosJogados)
);

CREATE TABLE ProjetoBD.JOGADOR (
    id_Jogador           INT                NOT NULL IDENTITY(1,1),
    Nome                 VARCHAR(20)        NOT NULL,
    Data_Nasc            DATE,
    Nacionalidade VARCHAR(35),
    Equipa               INT                NOT NULL,
    Posicao              VARCHAR(10)        NOT NULL,
    PRIMARY KEY (id_Jogador),
    FOREIGN KEY (Equipa) REFERENCES ProjetoBD.EQUIPAREAL(id_Equipa),
);

CREATE TABLE ProjetoBD.EQUIPAFICTICIA (
    id_Equipa_Ficticia   INT                NOT NULL IDENTITY(1,1),
    TotalPontos                 DECIMAL(6,1)  CHECK(TotalPontos >= 0) Default(0),
    Nome                        VARCHAR(40)        NOT NULL,
    utilizador                  VARCHAR(40),
    PRIMARY KEY (id_Equipa_Ficticia),
    FOREIGN KEY (utilizador) REFERENCES ProjetoBD.UTILIZADOR(Email),
);

CREATE TABLE ProjetoBD.Jogo (
    id_Jogo              INT           NOT NULL IDENTITY(1,1),
    Data                 DATE   NOT NULL,
    NumJornada           INT           NOT NULL,
    Assistência          INT,
    ResultadoCasa INT           NOT NULL,
    ResultadoFora INT           NOT NULL,
    EquipaCasa           INT           NOT NULL,
    EquipaVisitante      INT           NOT NULL,
    Campeonato           INT           NOT NULL,
    Estadio              INT           NOT NULL,
    PRIMARY KEY (id_Jogo),
    FOREIGN KEY (EquipaCasa) REFERENCES ProjetoBD.EQUIPAREAL(id_Equipa),
    FOREIGN KEY (EquipaVisitante) REFERENCES ProjetoBD.EQUIPAREAL(id_Equipa),
    FOREIGN KEY (Estadio) REFERENCES ProjetoBD.ESTADIO(id_Estadio),
```

```
    FOREIGN KEY (Campeonato) REFERENCES ProjetoBD.CAMPEONATO(id_Campeonato)
);

CREATE TABLE ProjetoBD.JOGADORPLAYSINJOGO (
    Jogador_ID          INT                 NOT NULL,
    Jogo_ID             INT                 NOT NULL,
    Pontuacao           DECIMAL(2,1)  NOT NULL,
    PRIMARY KEY (Jogador_ID, Jogo_ID),
    FOREIGN KEY (Jogador_ID) REFERENCES ProjetoBD.JOGADOR(id_Jogador),
    FOREIGN KEY (Jogo_ID) REFERENCES ProjetoBD.JOGO(id_Jogo),
    CONSTRAINT CHK_Pontuacao CHECK (Pontuacao>=0.0 AND Pontuacao<=10.0)
);

CREATE TABLE ProjetoBD.JOGADORPERTENCEEQUIPAFICT (
    EquipaFict_ID INT                   NOT NULL,
    Jogador_ID          INT                 NOT NULL,
    PRIMARY KEY (EquipaFict_ID, Jogador_ID),
    FOREIGN KEY (EquipaFict_ID) REFERENCES ProjetoBD.EQUIPAFICTICIA(id_Equipa_Ficticia),
    FOREIGN KEY (Jogador_ID) REFERENCES ProjetoBD.JOGADOR(id_Jogador),
);

CREATE TABLE ProjetoBD.UTILIZADOR (
    Email                       VARCHAR(40)         NOT NULL,
    Nick                        VARCHAR(20)         NOT NULL,
    Hashed_Pass                 INT                         NOT NULL,
    Salt                        INT                 Default(256),
    Data_Nasc                   DATE,
    PRIMARY KEY (Email),
);

CREATE TABLE ProjetoBD.LIGA (
    id_Liga                     INT                 NOT NULL IDENTITY(1,1),
    Nome                        VARCHAR(40)         NOT NULL,
    MaximoParticipantes   INT CHECK(MaximoParticipantes>=2 AND MaximoParticipantes <=10),
    EmailCriador          VARCHAR(40)           NOT NULL,
    PRIMARY KEY (id_Liga),
    FOREIGN KEY (EmailCriador) REFERENCES ProjetoBD.UTILIZADOR(Email)
);

CREATE TABLE ProjetoBD.EquipaFictPartLiga (
    Equipa_Ficticia_ID    INT                   NOT NULL,
    Liga_ID                     INT                 NOT NULL,
    PRIMARY KEY (Equipa_Ficticia_ID, Liga_ID),
    FOREIGN KEY (Equipa_Ficticia_ID) REFERENCES ProjetoBD.EQUIPAFICTICIA(id_Equipa_Ficticia),
    FOREIGN KEY (Liga_ID) REFERENCES ProjetoBD.LIGA(id_Liga)
);
```

9.Normalização

O processo de normalização das tabelas da base de dados foi realizado com sucesso.

Depois de uma conversa com o professor na aula prática, acabamos por chegar à conclusão que não era necessário decompor qualquer tabela, visto que estas já se encontravam na BCNF.

10. User Defined Functions

Desenvolvidas **11** user defined functions com o principal objetivo de filtragem de informação para apresentação na interface de interação com a base de dados.

Mais especificamente:

Nome	Função
getCampeonatosFiltrados	Filtrar os campeonatos consoante os filtros escolhidos na interface
getEquipaFictPartLigaFiltradas	Filtrar as equipas pertencentes a uma determinada liga escolhida na interface
getEquipasFicticiasFiltradas	Filtrar as equipas fictícias consoante os filtros escolhidos na interface
getEquipasFiltradas	Filtrar as equipas reais consoante os filtros escolhidos na interface
getEstadiosFiltrados	Filtrar os estadios consoante os filtros escolhidos na interface
getJogadoresFiltradas	Filtrar os jogadores consoante os filtros escolhidos na interface
getJogadorPertEquipaFictFiltrados	Filtrar os jogadores pertencentes a uma determinada equipa fictícia escolhida na interface
getJogadorPlaysInJogoFiltrados	Filtrar os jogadores que jogaram um determinado jogo selecionado na interface
getJogosFiltrados	Filtrar os jogos consoante os filtros escolhidos na interface
getLigasFicticiasFiltradas	Filtrar as ligas fictícias consoante os filtros escolhidos na interface
getUtilizadoresFiltradas	Filtrar os utilizadores consoante os filtros escolhidos na interface

11. Triggers

Desenvolvidos **13** triggers com 2 principais objetivos, o de implementar as restrições de integridade definidas na análise de requisitos que se referem a duas tabelas e o de atualizar automatimente determinadas tabelas ao editar outras. Para implementar os triggers nós usamos cursores e, sempre que tínhamos de editar outras tabelas, transactions de modo a manter a integridade do sistema de base de dados.

Mais especificamente:

Nome	Tabela	Função
insertVerifications	EQUIPAFICTICIA	Verificar se o utilizador associado à equipa que queremos inserir não tem outra equipa
verificationsInsertEquipaFictPartLiga	EquipaFictPartLiga	Verificar se a liga onde a Equipa Fictícia irá entrar tem um número de equipas inferior ao seu máximo de participantes
insertEquipaReal	EQUIPAREAL	Verificar se o Campeonato onde a Equipa que queremos inserir seria colocada tem um número de equipas inferior ao seu número máximo de equipas
verificationsInsertJogadorPertenceEquipaFict	JOGADORPERTENCEEQUIPAFICT	Verificar se o número de jogadores da Equipa Fictícia em questão é inferior a 11(máximo na nossa aplicação)
deleteJogadorPlaysInJogoTrigger	JOGADORPLAYSINJOGO	Diminuir a pontuação das equipas fictícias com o jogador referenciado no tuplo a ser apagado
insertJogadorPlaysInJogoTrigger	JOGADORPLAYSINJOGO	Aumentar a pontuação das equipas fictícias com o jogador referenciado no tuplo a ser inserido
updateJogadorPlaysInJogoTrigger	JOGADORPLAYSINJOGO	Atualizar a pontuação das equipas fictícias com o jogador referenciado no tuplo a ser atualizado
verificationsTriggerJogadorPlaysInJogo	JOGADORPLAYSINJOGO	Verificar se o jogador no tuplo a inserir pertence a uma das equipas em jogo e se essa equipa tem menos de 14 jogadores nesse jogo até agora
deleteJogoTrigger	Jogo	Remover os efeitos que o jogo a ser apagado teve na classificação das equipas que nele participaram
insertJogoTrigger	Jogo	Atualizar nas equipas reais os pontos, jogos jogados, golos marcados, golos sofridos e classificação consoante o resultado do jogo

verificationsTrigger	Jogo	Verificar se a assistência do jogo é inferior ao limite do estádio, verificar se as equipas são do mesmo campeonato e se são equipas diferentes
insertLigaVerifications	LIGA	Verificar se o utilizador criador da liga não tem outra liga onde é criador
updateLigaVerifications	LIGA	Verificar se o novo máximo de participantes é igual ou superior ao atual número de participantes nessa liga

12. Stored Procedures

Desenvolvidos **8** stored procedures com o principal objetivo de apagar os tuplos que estão dependentes do tuplo que queremos apagar. De forma semelhante aos triggers desenvolvidos, sempre que era necessário efetuar alterações em mais do que uma tabela, foram usadas transactions de forma a assegurar a integridade dos dados da base de dados.

Mais especificamente:

Nome	Função
deleteCampeonato	Apaga um tuplo Campeonato e os tuplos que o têm o seu id como chave estrangeira.
deleteEquipaFicticia	Apaga um tuplo Equipa Fictícia e os tuplos que o têm o seu id como chave estrangeira.
deleteEquipaReal	Apaga um tuplo Equipa Real e os tuplos que o têm o seu id como chave estrangeira.
deleteEstadio	Apaga um tuplo Estadio e os tuplos que o têm como o seu id chave estrangeira.
deleteJogador	Apaga um tuplo Jogador e os tuplos que o têm como o seu id chave estrangeira.
deleteJogo	Apaga um tuplo Jogo e os tuplos que o têm como o seu id chave estrangeira.
deleteLiga	Apaga um tuplo Liga e os tuplos que o têm como o seu id chave estrangeira.
deleteUtilizador	Apaga um tuplo Utilizador e os tuplos que o têm o seu email como chave estrangeira.

13. Views

Desenvolvidos **7** views com o principal objetivo de servirem de base de informação para as comboBox utilizadas na interface de interação com a Base de Dados.

Mais especificamente:

Nome	Função
Emails_Nick_Utilizador	Utilizada para conversão Email -> Nick do Utilizador
ID_Jogador	Utilizada para conversão ID -> Nome do Jogador
ID_Liga	Utilizada para conversão ID -> Nome da Liga
ID_Nome_Campeonato	Utilizada para conversão ID -> Nome do Campeonato
ID_Nome_Equipa	Utilizada para conversão ID -> Nome da Equipa Real
ID_Nome_Equipa_Fict	Utilizada para conversão ID -> Nome da Equipa Fícticia
ID_NomeEstadio	Utilizada para conversão ID -> Nome do Estádio

14. Conclusão

Concluímos que o presente trabalho foi benéfico para o desenvolvimento das capacidades e conhecimentos ligados à área de Base de Dados, desde a preparação e realização dos diagramas mais úteis nesta área (DER, Modelo Relacional) até aos conhecimentos ligados à linguagem SQL. Adicionalmente, foi proveitoso na expansão das capacidades de trabalho em equipa dos diferentes elementos do grupo.

É de realçar que os objetivos deste trabalho foram alcançados com sucesso.