Relatório BD Projeto Final

Sistema de Gestão de Jogos Coletivos entre Universidades

Rafael Fernandes - 118956 **Luís Tojal - 119636**

2024/2025

Prof. Joaquim Sousa Pinto

deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Índice

1 Introdução

No âmbito da UC de Base de Dados, desenvolvemos um sistema de Gestão de Jogos Coletivos entre Universidades. Neste sistema conseguimos gerir Inscritos (Atletas, Treinadores e Árbitros), Equipas, Jogos, Fases, Universidades e Associações Académicas.

Ao longo deste relatório iremos explicar a organização do nosso projeto, o processo de desenvolvimento e também irá conter uma explicação detalhada dos processos que utilizamos para ligação entre a interface e a base de dados, visto que optamos por utilizar Python, Flask, HTML, CSS e JavaScript para o desenvolvimento da mesma.

2 Estruturação do Projeto

Na pasta submetida está todo o código desenvolvido neste projeto. Na pasta principal do projeto temos os ficheiros `app.py`, `querrys.py`, `.env`, `DDL.sql`, `INSERVALUES.sql`, `storeProcedure.sql`, `triggers.sql`, `UDF.sql`. Na pasta templates temos todas as páginas `.html`, na pasta static temos a pasta com o ficheiro de estilização `style.css` e a pasta com todos os scripts `.js` utilizados nas páginas html.

- O ficheiro `app.py` contém as funções correspondentes às solicitações HTTP, este retorna páginas HTML ou dados JSON.
- O ficneiro `querrys.py` contém algumas das queries utilizadas pelo `app.py`.
- O ficheiro `.env` é o ficheiro com as configurações da base de dados que é utilizado no ficheiro `app.py` para conectar o Flask com a base de dados. Esta é a configuração do `.env`:

```
DB_DRIVER = 'ODBC Driver 17 for SQL Server'
DB_SERVER = 'mednat.ieeta.pt'
DB_PORT = '8101'
DB_DATABASE = 'p8g2'
DB_USER = 'p8g2'
DB_PASSWORD = 'XXXXXXXXX'
```

- Os ficheiros `.sql` têm cada um a sua funcionalidade.
  - `DDL.sql` - Contém o DDL utilizado para criar as tabelas da nossa base de dados.
  - `INSERTVALUES.sql` - Contém os inserts para as tabelas da nossa base de dados.
  - `storeProcedure.sql` - Contém a lista de Stored Procedures utilizadas na nossa base de dados
  - `triggers.sql` - Contém a lista de triggers utilizados na nossa base de dados
- Também incluímos a imagem do nosso DER em `der.png`.

3 Análise de Requisitos

3.1 Requisitos Funcionais

1. O sistema deve permitir o registo de Fases, cada uma identificada por um nome.
2. O sistema deve permitir o registo de Modalidades, cada uma com o seu nome e número máximo de jogadores.
3. O sistema deve permitir associar Organizações a Fases.
4. O sistema deve permitir associar Associações Académicas a Organizações.
5. O sistema deve permitir associar Modalidades a Associações Académicas, com possibilidade de registar o número de medalhas.
6. O sistema deve permitir o registo de Equipas associadas a Modalidades e Associações Académicas.
7. O sistema deve permitir o registo de Jogos, incluindo data, duração, resultado, local, fase, modalidade e equipas participantes.
8. O sistema deve permitir o registo de Pessoas, incluindo nome, número de cartão de cidadão, data de nascimento, email, telefone e Associação Académica.
9. O sistema deve permitir registar Pessoas como Árbitros, Atletas ou Treinadores.
10. O sistema deve permitir associar Pessoas a Equipas, registando a data de inscrição.
11. O sistema deve permitir associar Pessoas a Modalidades.
12. O sistema deve permitir o registo de Tipos de Medalhas.
13. O sistema deve permitir registar Medalhas para uma Modalidade, Associação Académica e Ano, associando-as a um Tipo de Medalha.
14. O sistema deve permitir associar Medalhas a Pessoas específicas.
15. O sistema deve permitir o registo de Universidades, identificadas por nome e morada, associadas a uma Associação Académica.
16. O sistema deve permitir associar uma Pessoa a um login, garantindo a autenticação com palavra-passe.
17. O sistema deve permitir pesquisar Pessoas por nome ou por Associação Académica de forma eficiente.

3.2 Requisitos Não Funcionais

1. O sistema deve ser intuitivo e fácil de usar, centrado na gestão das diferentes entidades (Fases, Modalidades, Organizações, etc.).
2. O sistema deve ser seguro, garantindo a confidencialidade dos dados pessoais e restrições de acesso às operações sensíveis (ex.: atribuição de medalhas).
3. O sistema deve ser eficiente, garantindo tempos de resposta rápidos nas operações de registo, pesquisa e consulta de dados.
4. O sistema deve suportar múltiplos utilizadores em simultâneo sem degradação significativa de desempenho.

4 Entidades

O nosso projeto consta destas entidades:

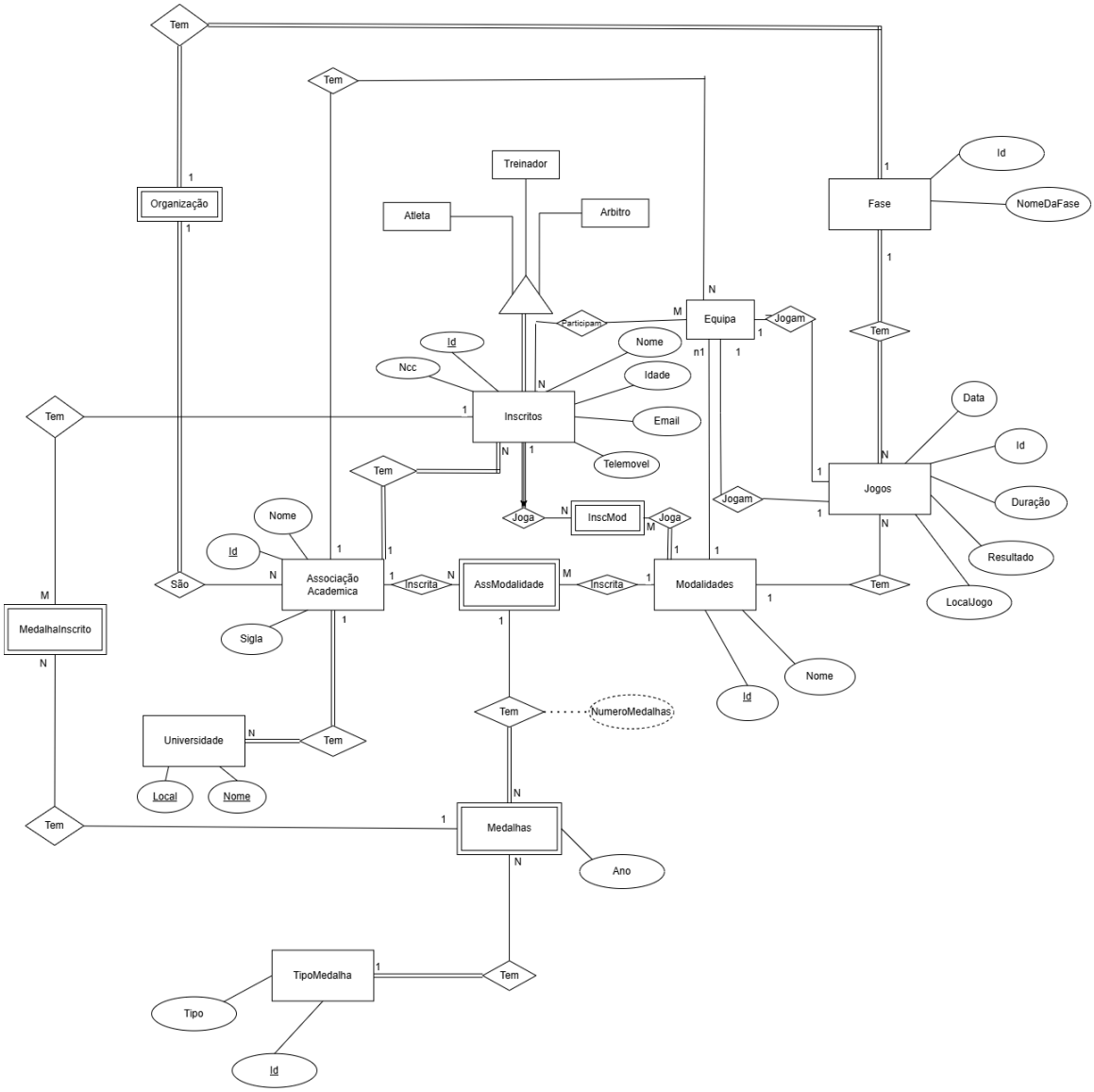
- **Inscrito** - Representa os inscritos, que podem ser Atletas, Treinadores e Árbitros
  - **Atleta**
  - **Treinador**
  - **Árbitro**
- **Universidades** - Representa as diferentes Universidades
- **Associação Académica** - Representa as diferentes Associações Académicas
- **Modalidades** - Representa as diferentes Modalidades
- **Medalhas** - Representa as medalhas ganhas pelas associações
- **TipoMedalha** - Representa o tipo de medalha (Ouro, Prata, Bronze)
- **Equipa** - Representa as equipas preenchidas por inscritos de cada modalidade das Associações
- **Jogos** - Representa os jogos realizados entre 2 equipas
- **Fase** - Representa as fases onde os jogos decorrem
- **Organização** - Representa as Associações responsáveis pela organização das fases
- **AssModalidade** - Representa modalidades por associação
- **InscMod** - Representa os inscritos por modalidade
- **MedalhaInscrito** - Representa as medalhas por inscrito

5 Normalização

Durante o desenho das relações e entidades, para garantir um melhor funcionamento, seguimos as regras que nos permitiram alcançar a Terceira Forma Normal (3FN)

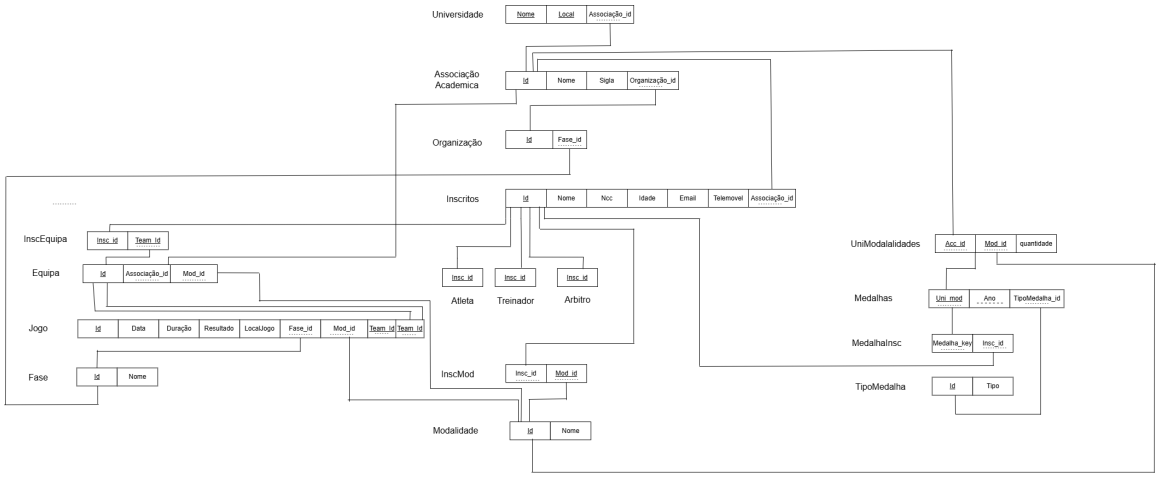
6 Diagrama Entidade-Relação

Baseado nas entidades e requisitos que já mostramos, construímos este DER.



7 Esquema Relacional

Baseado no diagrama do DER apresentado anteriormente construímos este ER



8 Data Definition Language (DDL)

De modo a criar a nossa base de dados criámos as tabelas e relações seguindo as regras acima indicadas.

```
1
2  CREATE TABLE dbo.FADU_FASE (
3      Id INT IDENTITY PRIMARY KEY,
4      Name VARCHAR(64)
5  )
6  GO
7
8  CREATE TABLE dbo.FADU_MODALIDADE (
9      Id INT IDENTITY PRIMARY KEY,
10     Name VARCHAR(64) NOT NULL,
11     MaxPlayers INT NOT NULL
12 )
13 GO
14
15 CREATE TABLE dbo.FADU_ORGANIZACAO (
16     Id INT IDENTITY PRIMARY KEY,
17     Fase_id INT CONSTRAINT FK_ORG_FASE REFERENCES dbo.FADU_FASE
18 )
19
20 ...
```

Estes são alguns exemplos resto das tabelas encontram-se no ficheiro `DDL.sql`.

9 Data Manipulation Language (DML)

No desenvolvimento do nosso projeto, utilizámos vários métodos:

- Trigger
- Cursor
- Index
- User Defined Function (UDF)
- Stored Procedure
- Transaction

Para além destes métodos, como foi solicitado pelo Professor no feedback da apresentação final, adicionamos um formulário de LogIn com hashing na password.

9.1 Table Inserts

Inserimos alguns dados na nossa tabela de modo a conseguir realizar testes através destes inserts:

```
1
2
3  INSERT INTO FADU_ASSOCIA AO_ACADEMICA ([Name], [Sigla], Org_Id) VALUES
4  ('Associa  o Acad mica do Norte Virtual', 'AANV', 1),
5  ('Associa  o Estudantil T cnica Avan ada', 'AETA', 1),
6  ('N cleo Universit rio Digital do Sul', 'NUDS', 1),
7  ('Federa  o Acad mica Virtual Atl ntica', 'FAVA', 1),
8  ('Liga Acad mica do Centro', 'LAC', 1),
9  ('Uni o de Estudantes Insulares', 'UEI', 1),
10 ('Associa  o Polit cnica Digital', 'APD', 2),
11 ('Conselho Universit rio Litoral', 'CUL', 2),
12
13
14
15 INSERT INTO FADU_UNIVERSIDADE ([Address], [Name], [Ass_Id]) VALUES
16 ('Rua do Saber 21', 'Universidade L cida', 1),
17 ('Avenida do Conhecimento 42', 'Instituto Omega', 2),
18 ('Pra a Acad mica 88', 'Escola Superior Nova Era', 3),
19 ('Largo das Ci ncias 99', 'Universidade Mar tima Digital', 4),
20 ('Alameda das Artes 15', 'Universidade Criativa', 5),
21 ('Travessa da Inova  o 77', 'Instituto Polit cnico Global', 6),
22 ('Rotunda Cient fica 33', 'Universidade de Ci ncias Aplicadas', 7),
23 ('Boulevard Tecnol gico 12', 'Faculdade de Engenharia Avan ada', 8),
24 ...
```

Estes são alguns dos imports no ficheiro `INSERTVALUES.sql`

9.2 Trigger + Transaction

Um exemplo de Trigger utilizado no desenvolvimento do projeto é este trigger que, ao apagar uma equipa, em vez de apagá-la diretamente, apaga-a dos Inscritos que lá estavam, os jogos onde estava e, por fim, da tabela das equipas. Neste Trigger está presente uma Transaction que permite voltar ao estado que estava antes do delete caso dê erro em algum dos deletes.

```
1
2  CREATE OR ALTER TRIGGER trg_DeleteTeam
3  ON FADU_EQUIPA
4  INSTEAD OF DELETE
5  AS
6  BEGIN
7      BEGIN TRY
8          BEGIN TRANSACTION;
9
10         DELETE FROM FADU_PERSONEQUIPA
11         WHERE EQUIPA_Id IN (SELECT Id FROM deleted);
12
13         DELETE FROM FADU_JOGO
14         WHERE Equipa_id1 IN (SELECT Id FROM deleted) OR Equipa_id2 IN (
               SELECT Id FROM deleted);
15
16         DELETE FROM FADU_EQUIPA
17         WHERE Id IN (SELECT Id FROM deleted);
18
19         COMMIT TRANSACTION;
20     END TRY
21     BEGIN CATCH
22         ROLLBACK TRANSACTION;
23         PRINT 'Error occurred while deleting the team and related records.';
24         THROW;
25     END CATCH
26 END;
```

9.3 Stored Procedure + Cursor

Um exemplo de Cursor é este seguinte, que verifica na tabela dos jogos quantas vitórias têm as equipas e faz um ranking:

```
1
2  CREATE OR ALTER PROCEDURE dbo.ranking_cursor
3  AS
4  BEGIN
5      SET NOCOUNT ON;
6
7      DECLARE @Equipa_Id1 INT, @Equipa_Id2 INT, @Resultado NVARCHAR(20)
8      DECLARE @Golos1 INT, @Golos2 INT
9      DECLARE @Vencedor_Id INT
10
11     DECLARE @Ranking TABLE (
12         Ass_Id INT,
13         Total_Jogos_Ganhos INT
14     )
15     DECLARE jogo_cursor CURSOR FOR
16     SELECT Equipa_id1, Equipa_id2, Resultado
17     FROM FADU_JOGO
18     WHERE Resultado LIKE '%-%'
19
20     OPEN jogo_cursor
21     FETCH NEXT FROM jogo_cursor INTO @Equipa_Id1, @Equipa_Id2, @Resultado
22
23     WHILE @@FETCH_STATUS = 0
24     BEGIN
25
26         SET @Golos1 = CAST(SUBSTRING(@Resultado, 1, CHARINDEX('-',
               @Resultado) - 1) AS INT)
27         SET @Golos2 = CAST(SUBSTRING(@Resultado, CHARINDEX('-', @Resultado)
               + 1, LEN(@Resultado)) AS INT)
28
29         IF @Golos1 > @Golos2
30             SET @Vencedor_Id = @Equipa_Id1
31         ELSE IF @Golos2 > @Golos1
32             SET @Vencedor_Id = @Equipa_Id2
33         ELSE
34             SET @Vencedor_Id = NULL
35
36         IF @Vencedor_Id IS NOT NULL
37         BEGIN
38             DECLARE @Ass_Id INT
39             SELECT @Ass_Id = Ass_Id FROM FADU_EQUIPA WHERE Id = @Vencedor_Id
40             ...
```

```
            ...
            IF EXISTS (SELECT 1 FROM @Ranking WHERE Ass_Id = @Ass_Id)
            BEGIN
                UPDATE @Ranking
                SET Total_Jogos_Ganhos = Total_Jogos_Ganhos + 1
                WHERE Ass_Id = @Ass_Id
            END
            ELSE
            BEGIN
                INSERT INTO @Ranking (Ass_Id, Total_Jogos_Ganhos)
                VALUES (@Ass_Id, 1)
            END
        END

        FETCH NEXT FROM jogo_cursor INTO @Equipa_Id1, @Equipa_Id2,
            @Resultado
    END

    CLOSE jogo_cursor
    DEALLOCATE jogo_cursor

    SELECT
        R.Ass_Id,
        A.Name AS Association_Name,
        R.Total_Jogos_Ganhos
    FROM @Ranking R
    JOIN FADU_ASSOCIACAO_ACADEMICA A ON R.Ass_Id = A.Id
    ORDER BY R.Total_Jogos_Ganhos DESC
END
```

9.4 User Defined Function (UDF)

Um exemplo de UDF é esta que faz o hashing das passwords através da função HASHBYTES do SQL:

```
CREATE OR ALTER FUNCTION dbo.HashPass(@Password NVARCHAR(50))
RETURNS VARBINARY(32)
AS
BEGIN
    DECLARE @HashThis NVARCHAR(4000);
    DECLARE @Hash VARBINARY(32);

    SET @HashThis = @Password;
    SET @Hash = HASHBYTES('SHA2_256', @HashThis);

    RETURN @Hash;
END;
```

9.5 Index

Um exemplo de index na nossa implementação é este, que é utilizado para ter uma pesquisa mais eficiente nas barras de pesquisa:

```
CREATE INDEX IDX_PERSON_NAME ON dbo.FADU_PERSON (Name);

CREATE INDEX IDX_PERSON_ASS_ID ON dbo.FADU_PERSON (Ass_Id);
```

10 Interação entre a interface e a Base de Dados

10.1 Interface

A interface do utilizador foi desenvolvida em **HTML** (Estrutura do Conteúdo) e **CSS** (Estilização da página).

10.2 Configuração do Backend

Para configuração do Backend utilizámos **Flask**, que é uma framework em **Python**. Esta framework é responsável por processar as requisições feitas pela interface, interagindo com a base de dados e retornando as respostas apropriadas para cada caso específico.

A conexão entre o Flask e a base de dados SQL Server é realizada através da biblioteca **pyodbc**, que permite que o Python se conecte a bases de dados SQL Server e tenha interações com as mesmas.

10.3 Requests

Quanto ao processamento de requests, quando o utilizador interage com os formulários da interface, os dados inseridos nos mesmos são enviados para o backend, onde serão recebidos num endpoint Flask, que os irá receber e inserir na base de dados.

10.4 Uso de AJAX para Requisições Dinâmicas

Utilizamos **AJAX**, que é uma biblioteca de **JavaScript** para poder enviar e receber dados sem recarregar a página inteira, proporcionando uma experiência mais fluida e rápida para o utilizador.

Ao submeter um formulário para adicionar um inscrito, por exemplo, o AJAX vai enviar um request para o endpoint do Flask que trata da adição de inscritos. Após isso, o request é processado pelo Flask, o inscrito é adicionado à base de dados e vai ser retornada uma resposta na página sem necessidade de a recarregar.

11 Conclusão

Em suma, acreditamos que este projeto cumpriu com êxito os requisitos propostos. Implementamos uma base de dados robusta, utilizando métodos seguros e eficientes, fundamentados nos princípios de formalização estudados. Além disso, desenvolvemos uma interface limpa e intuitiva que permite gerenciar as informações da base de dados de forma prática e eficiente.

Para alcançar esses objetivos, aplicamos os conceitos de UDFs, stored procedures e triggers que aprendemos durante as aulas. As tecnologias utilizadas incluem SQL Server para o gerenciamento da base de dados e Python (com Flask), HTML, JavaScript e AJAX para a construção da aplicação, garantindo assim um fluxo de ações coerente e eficaz.

Este projeto evidenciou a importância das boas práticas no desenvolvimento de bases de dados, tanto na manutenção da eficiência quanto na prevenção de erros futuros. O relatório apresentado não inclui todos os triggers e stored procedures implementados, a fim de evitar maçar o leitor. No entanto, recomendamos fortemente a consulta aos ficheiros SQL incluídos na submissão para uma análise mais completa.