Base de Dados

Relatório Principal - LineUp

Luís Oliveira nº 98543
Arthur Monetto nº 102667

Prof. Joaquim Sousa Pinto
Prof. Regente Carlos Costa

Ano letivo
2024-2025

1 Introdução

O LineUp é uma aplicação web desenvolvida para auxiliar os utilizadores a registarem-se nos seus clubes de futebol, permitindo controlo de diversos aspectos administrativos. Dentro das funcionalidades está incluída a gestão dos jogadores e funcionários disponíveis, bem como a componente financeira do clube, através de compra e venda destes ativos.

O projeto foi inspirado em títulos reconhecidos como Football Manager e FIFA (agora EA Sports FC), nomeadamente nos seus modos de jogo orientados para a simulação de carreira.

2 Análise de Requisitos

2.1 Funcionalidades

- **Sign Up**: Cada utilizador regista-se na nossa aplicação, e assim, criar o seu próprio clube, ao qual ficará associado durante a sessão.
- **Log In**: O utilizador entra na aplicação com as credenciais com que se registou.
- **Adicionar Jogador**: O utilizador ao adicionar um jogador ao seu clube, irá criar um novo jogador que irá ser introduzido na base de dados.
- **Listar Jogadores e Funcionários**: Listagem de todos os jogadores e funcionários do clube.
- **Comprar Jogador**: O utilizador pode comprar um jogador que exista noutro clube. Neste processo, destacar que o orçamento de transferências será atualizado, e portanto, o valor do jogador será descontado nesse orçamento.
- **Histórico de Transferências**: Esta funcionalidade permite visualizar o histórico de transferências de jogadores que decorreram no clube, apresentado em forma de tabela, onde em cada linha é referido o nome do jogador, o clube de origem e o valor da transferência.
- **Escolher onze titular**: Com os jogadores disponíveis do clube, o utilizador pode escolher, para cada posição tática, o seu jogador predileto para assumir a titularidade.
- **Log Out**: O utilizador pode terminar e entrar noutra sessão para gerir outro clube.

2.2 Entidades e Atributos

- **AspNetUsers**: Id, Username, Email, PasswordHash, ClubId, NormalizedUserName, NormalizedEmail, SecurityStamp, ConcurrencyStamp, LockoutEnabled
- **Club**: Club_id, Club_name, TransferBudget, FoundationDate
- **Games**: Game_id, Game_location, team1_id, team2_id, Game_statistics
- **PlayerGameStats**: stat_id, Goals, Assists, YellowCards, RedCards, Game_id, Player_id, LeaveAt, EnterAt
- **GameStats**: Game_id, Goal_team1, Goal_team2
- **Player**: Player_id, Position, Rating, MarketValue, Person_id, Club_id, LastTransferValue, IsDeleted
- **Transfer**: Transf_id, Trans_value, Trans_player, Previous_club, Destination_club
- **Person**: Person_id, Person_name, BirthDate
- **Contracts**: Cont_id, Salary, StartDate, EndDate, Person_id
- **Employee**: Emp_id, Emp_function, Dep_id, Person_id
- **Department**: Dep_id, DepName, Club_id

2.3 DER

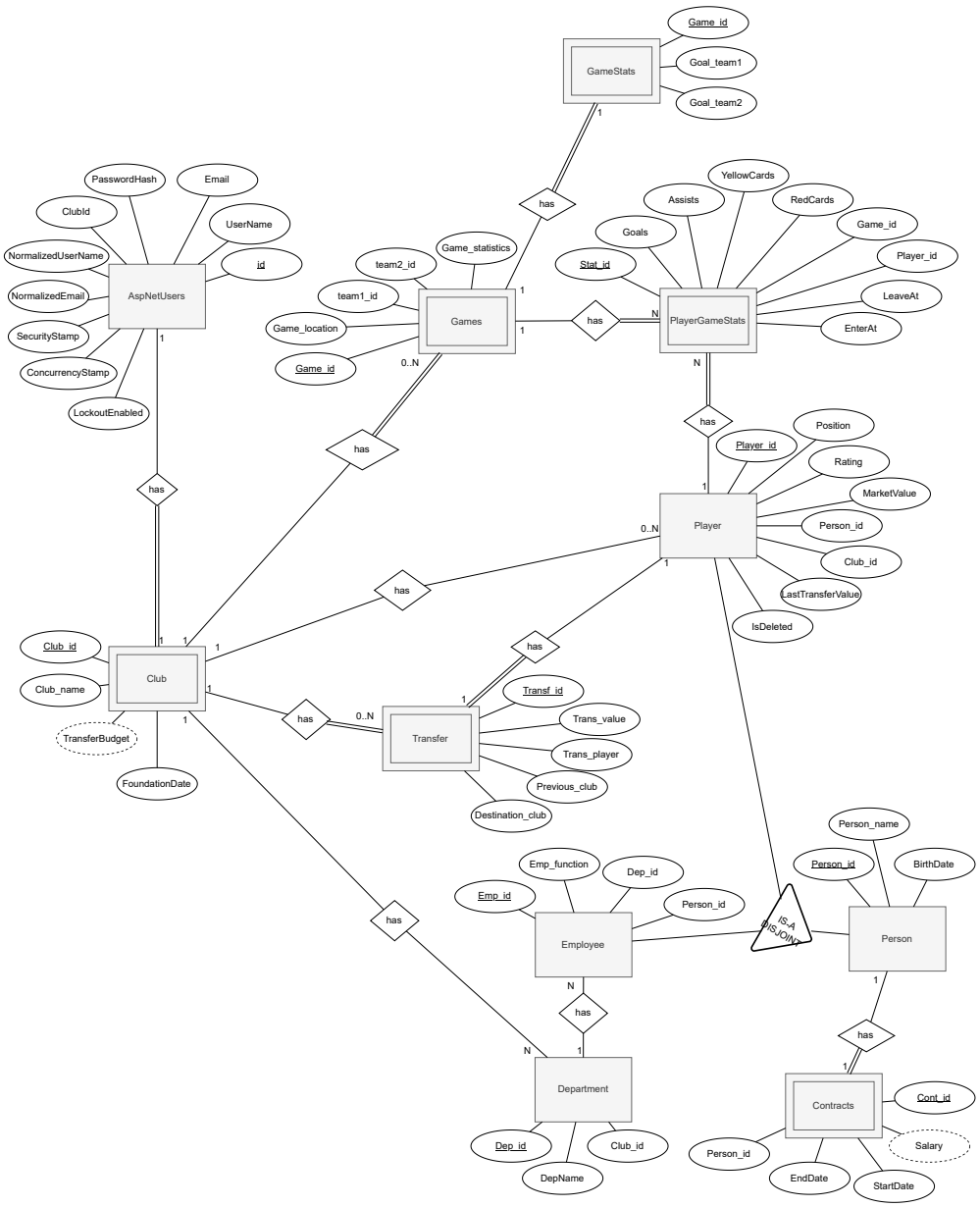

Figure 1: Diagrama Entidade Relação

2.4 ER

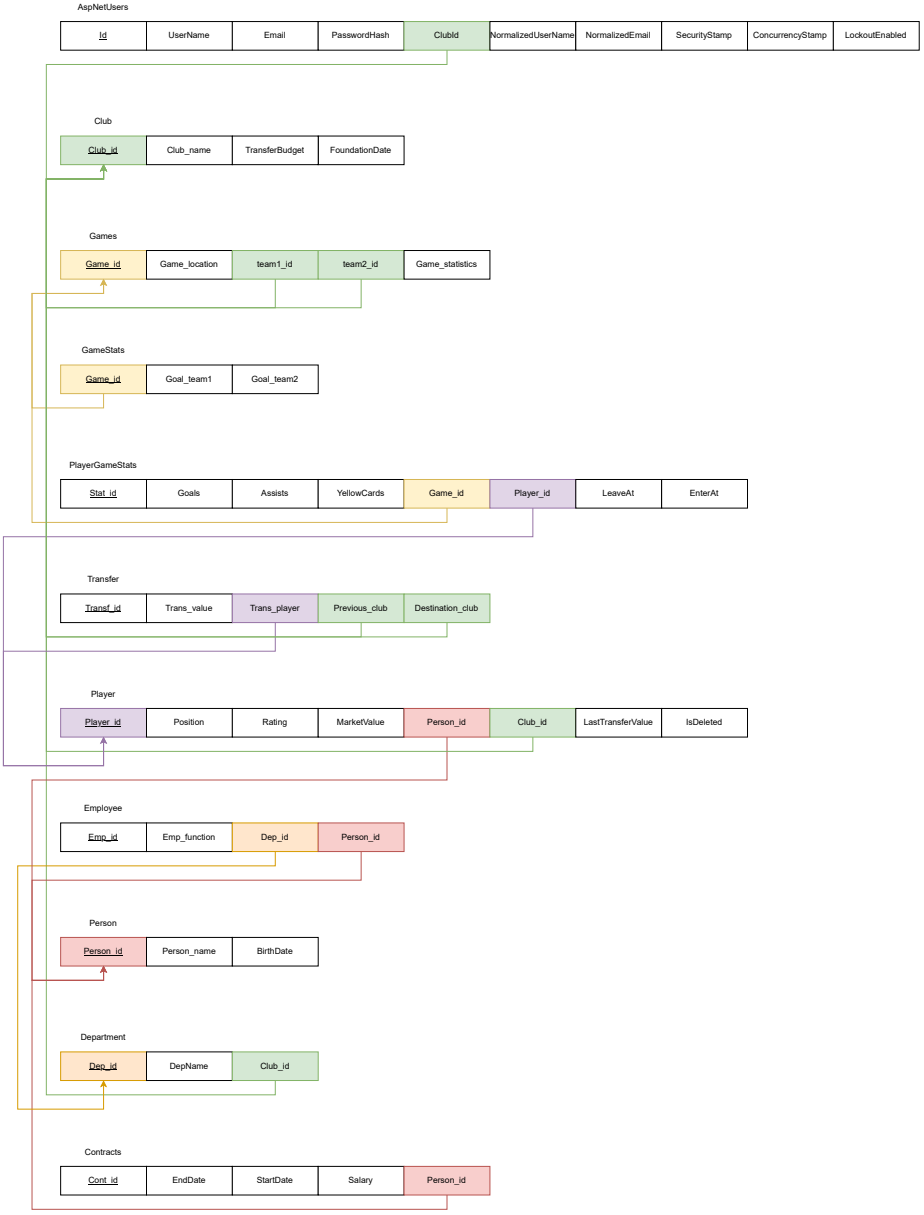

Figure 2: Esquema Relacional

3 Estrutura

3.1 Arquitetura do Projeto

Para este projeto, optou-se pelo desenvolvimento de uma WebApp interativa, utilizando as tecnologias *HTML*, *CSS* e *JavaScript*. Esta abordagem visa fornecer ao utilizador uma interface amigável, permitindo uma interação eficiente com os dados da aplicação.

A ligação à base de dados é estabelecida através de uma API desenvolvida em *.NET* (C), seguindo o padrão *MVC* (*Model-View-Controller*). Este padrão permite uma separação clara entre a lógica de apresentação, controlo e acesso a dados, facilitando a manutenção, escalabilidade e organização do projeto.

3.2 API e Camada de Acesso aos Dados

A estrutura da API é composta por:

- **Controllers**: Responsáveis por receber as requisições HTTP, coordenar a lógica da aplicação e retornar respostas adequadas (geralmente views ou JSON).
- **Services/Repositories**: Camada intermédia que realiza a lógica de negócio e gerencia o acesso a dados.
- **Models**: Representam os objetos de domínio da aplicação, refletindo a estrutura dos dados.

Relativamente ao acesso aos dados, todos os acessos são abstraídos através da camada de abstração, garantindo maior segurança, reusabilidade de código e separação de responsabilidades. Essa camada comunica com a base de dados exclusivamente através de *Stored Procedures* e *User-Defined Functions* (*UDF's*), evitando a escrita direta de queries SQL no código da aplicação.

```
public class UserRepository : IUserRepository
{
    private readonly AppDbContext _context;

    public UserRepository(AppDbContext context)
    {
        _context = context;
    }

    public async Task<AppUser> GetUserByEmailAsync(string email)
    {
        var userInfo = _context.Users
            .FromSqlRaw("EXEC dbo.sp_get_name_by_email @p0", email)
            .AsEnumerable()
            .FirstOrDefault();

        return userInfo;
    }
}
```

Essa abordagem garante que:

- o código permaneça limpo e organizado;
- todas as interações com a base de dados sejam previamente definidas e controladas no lado do servidor (SQL Server);
- a aplicação esteja protegida contra injeção de SQL, uma vez que os parâmetros são devidamente tratados;
- o comportamento dos *Stored Procedures* sejam validados e testados separadamente da lógica de aplicação.

A validação dos dados é feita em camadas apropriadas, garantindo que apenas dados consistentes sejam enviados para a base de dados, contribuindo para a integridade da aplicação.

4 Normalização

Dado o contexto deste trabalho, dentro do que foi proposto, o grupo não achou que fosse oportuno a utilização de qualquer Normalização.

5 SQL Progamming

- Stored Procedures
- User Defined Functions
- Triggers
- Views

6 Informações relevantes

É importante realçar que o grupo desenvolveu um segundo relatório, solicitado pelos docentes da Unidade Curricular, com o objetivo de prestar esclarecimentos relativamente à interface da aplicação, uma vez que esta foi implementada com recurso a uma tecnologia diferente da inicialmente proposta.

Assim, para uma melhor compreensão do contexto e dos conteúdos, recomenda-se a leitura desse segundo relatório.

Os *Triggers*, *Stored Procedures* e *UDF's* usados foram disponibilizados no segundo relatório, uma vez que só faz sentido descrever a interação do frontend com a base de dados através das ferramentas de SQL, uma vez que toda a interação com a base de dados é feita em SQL.

Foi criado ainda dentro da API um arquivo que popula a base de dados quando está vazia. Quando se for proceder testes, basta utilizar as credenciais:

```
Email: user0@test.pt | Senha: Test1234!
Email: user1@test.pt | Senha: Test1234!
```

7 Conclusões

Ao longo do semestre, o desenvolvimento deste projeto proporcionou ao grupo a oportunidade de consolidar competências e adquirir conhecimentos valiosos, com impacto direto no percurso académico e profissional.