

Base de Dados

Relatório Secundário - LineUp

Luís Oliveira nº 98543
Arthur Monetto nº 102667

Prof. Joaquim Sousa Pinto
Prof. Regente Carlos Costa

Ano letivo
2024-2025

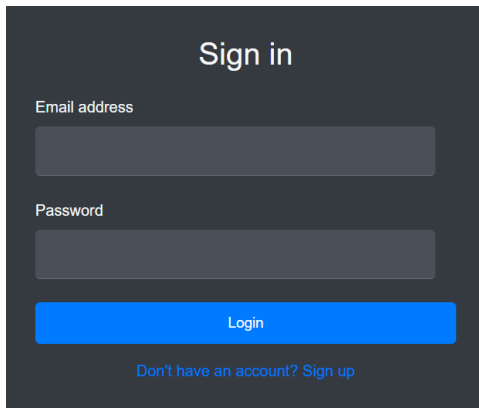
1 Introdução

Este relatório tem como objetivo descrever o funcionamento e a implementação da Web App desenvolvida como interface gráfica de apoio à base de dados criada no âmbito do projeto final da Unidade Curricular.

A aplicação foi concebida com o intuito de proporcionar uma experiência de utilização intuitiva, permitindo interação com dados de forma prática e eficiente. A interface foi desenvolvida com recurso a tecnologias HTML, CSS e JavaScript.

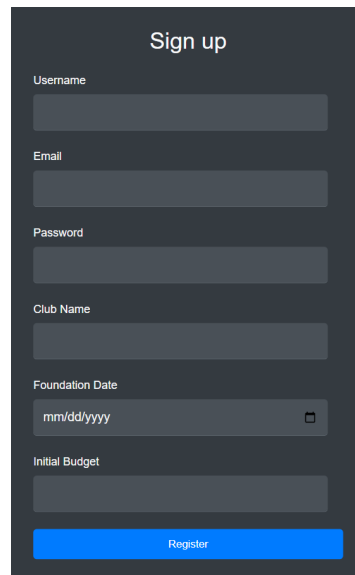
Para estabelecer ligação entre interface do utilizador e base de dados, foi desenvolvida uma API em ASP.NET Core. Esta API é responsável por processar as requisições provenientes do frontend, consultar a base de dados e devolver as informações tratadas de forma adequada.

2 Login/Register



The login screen has a dark background. At the top, it says "Sign in" in white. Below that, there are two input fields: "Email address" and "Password". Both fields are empty. Below the password field is a blue button labeled "Login". At the bottom, there is a link that says "Don't have an account? Sign up" in a smaller, lighter font.

Figure 1: Login Screen



The registration screen has a dark background. At the top, it says "Sign up" in white. Below that, there are five input fields: "Username", "Email", "Password", "Club Name", and "Foundation Date". The "Foundation Date" field has a date picker icon. Below the "Initial Budget" field is a blue button labeled "Register".

Figure 2: Registration Screen

Cada utilizador pode criar seu próprio clube, fornecendo como informações nome do clube, data de fundação e orçamento inicial. As informações sensíveis dos utilizadores são armazenadas de forma segura na base de dados. Para isso, a aplicação foi desenvolvida utilizando a framework ASP.NET Core, que inclui uma camada dedicada à segurança, por meio da biblioteca ASP.NET Core Identity. Essa biblioteca facilita o registo, autenticação e autorização dos utilizadores, garantindo que as senhas sejam sempre armazenadas de forma cifrada (utilizando técnicas como hashing e salting) e validadas conforme as melhores práticas de segurança.

	UserName	Email	PasswordHash	ClubId
1	user0	user0@test.pt	AQAAAAIAAYagAAAAEKb8nSx/0tk59wdnG3llg9I4NL50a8/3bi+...	3

Figure 3: Senha de utilizador cifrada após registo

3 Club

A gestão de um clube envolve diversas áreas, desde a contratação de jogadores até a administração dos funcionários e dos diferentes departamentos que compõem a organização. Para facilitar esse processo, desenvolvemos uma página simples que oferece uma visão geral completa do clube.

3.1 Adicionar Jogador

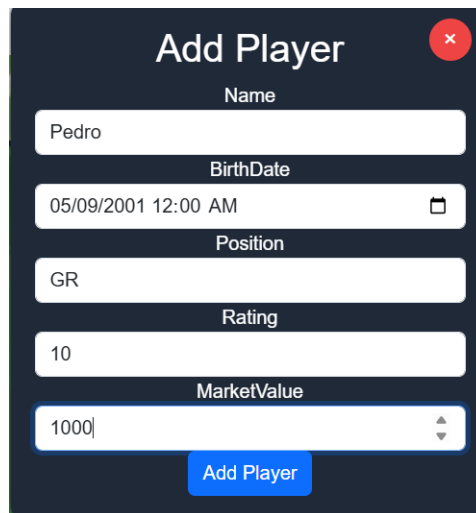
A screenshot of a web form titled "Add Player" with a dark blue background. The form contains several input fields: "Name" with the value "Pedro", "BirthDate" with the value "05/09/2001 12:00 AM" and a calendar icon, "Position" with the value "GR", "Rating" with the value "10", and "MarketValue" with the value "1000" and a spinner icon. At the bottom of the form is a blue button labeled "Add Player". A red close button with a white 'x' is in the top right corner.

Figure 4: Adicionar jogador

Esta opção permite que o utilizador adicione algum jogador à equipa sem qualquer custo ao clube. Assim que os campos sejam preenchidos e submetidos, os dados são enviados para a API que chama um Procedure para fazer a adesão do novo jogador ao clube.

```

1      ALTER PROCEDURE [dbo].[sp_add_player]
2      (
3          @ClubId int,
4          @PlayerName nvarchar(100),
5          @BirthDate date,
6          @Position nvarchar(100),
7          @Rating int,
8          @MarketValue int
9      )
10     AS
11     BEGIN
12         SET NOCOUNT ON;
13
14         DECLARE @NewPersonId INT;
15
16         INSERT INTO ConfManagement.Person (Person_Name, BirthDate)
17         VALUES (@PlayerName, @BirthDate);
18
19         SET @NewPersonId = SCOPE_IDENTITY();
20
21         INSERT INTO ConfManagement.Player (Person_id, Position, Rating,
22             MarketValue, Club_id, IsDeleted)
23         VALUES (@NewPersonId, @Position, @Rating, @MarketValue, @ClubId,
24             0);
25     END;

```

3.2 Comprar Jogador

A adição de um jogador à equipa sem custos ao clube torna a experiência monótona e foge ao que foi idealmente proposto pelo grupo, na medida em que se pretende que a aplicação aproxime a experiência do utilizador com um cenário mais realista. Deste modo, desenvolvemos a funcionalidade de poder comprar um jogador existente na base de dados e que esteja fora do clube que o utilizador esteja a gerir.



Figure 5: Comprar jogador

Através desse menu é possível filtrar de acordo com os valores de mercado, posição do jogador, e ainda, através da classificação (*rating*) do jogador. A lógica de compra do jogador é toda feita através de um Procedure que altera todos os campos necessários para que o jogador seja transferido para o clube comprador, e ainda cria de maneira automática entradas na tabela de transações do clube.

```

1 BEGIN
2     SET NOCOUNT ON;
3
4     DECLARE @oldClub INT;
5
6     BEGIN TRY
7         BEGIN TRANSACTION;
8
9         -- Get old club
10        SELECT @oldClub = Club_id
11        FROM ConfManagement.Player
12        WHERE Player_id = @PlayerId;
13
14        -- Add transfer value to old club's budget
15        UPDATE ConfManagement.Club
16        SET TransferBudget = TransferBudget + @TransValue
17        WHERE Club_id = @oldClub;
18
19        -- Update player transfer
20        UPDATE ConfManagement.Player
21        SET Club_id = @ClubId,
22            LastTransferValue = @TransValue
23        WHERE Player_id = @PlayerId;
24
25        -- Deduct transfer value from new club's budget
26        UPDATE ConfManagement.Club
27        SET TransferBudget = TransferBudget - @TransValue
28        WHERE Club_id = @ClubId;
29
30        COMMIT;
31    END TRY
32    BEGIN CATCH
33        ROLLBACK;
34
35    END CATCH
36 END

```

```

1 CREATE TRIGGER [ConfManagement].[trg_LogTransfer]
2 ON [ConfManagement].[Player]
3 AFTER UPDATE
4 AS
5 BEGIN
6     SET NOCOUNT ON;
7
8     IF UPDATE(Club_id)
9     BEGIN
10         INSERT INTO ConfManagement.Transference (
11             Trans_value,
12             Trans_player,
13             Previous_club,
14             Destination_club
15         )
16         SELECT
17             i.LastTransferValue,
18             i.Player_id,
19             d.Club_id,
20             i.Club_id
21         FROM inserted i
22         JOIN deleted d ON i.Player_id = d.Player_id
23         WHERE i.Club_id <> d.Club_id; -- only if actually changed
24     END
25 END

```


3.2.1 Filtrar Jogadores

Conforme referido na secção anterior, através do menu de compra é possível pesquisar pelo jogador de acordo com as características que o utilizador procura, através de um outro Procedure que faz a validação dos inputs por questões de segurança.

```
1 CREATE PROCEDURE [dbo].[sp_get_players_avaliabile]
2 (
3     @ClubId int,
4     @Offset int,
5     @PageSize int,
6     @Position nvarchar(2),
7     @FilterType nvarchar(100),
8     @FilterOption nvarchar(100)
9 )
10 AS
11
12 BEGIN
13     SET NOCOUNT ON;
14
15     -- Whitelist check
16     DECLARE @allowedColumns TABLE (ColumnName NVARCHAR(50));
17     INSERT INTO @allowedColumns VALUES ('MarketValue'), ('Rating');
18
19     DECLARE @allowedOrders TABLE (SortOrder NVARCHAR(10));
20     INSERT INTO @allowedOrders VALUES ('ASC'), ('DESC');
21
22     -- Defaults if input is invalid
23     IF NOT EXISTS (SELECT 1 FROM @allowedColumns WHERE ColumnName =
24         @FilterType)
25         SET @FilterType = 'MarketValue';
26
27     IF NOT EXISTS (SELECT 1 FROM @allowedOrders WHERE SortOrder =
28         @FilterOption)
29         SET @FilterOption = 'ASC';
30
31     -- Build ORDER BY clause safely
32     DECLARE @OrderByClause NVARCHAR(200) = QUOTENAME(@FilterType) + '
33         ' + @FilterOption;
34
35     -- Build SQL string
36     DECLARE @sql NVARCHAR(MAX) = '
37         SELECT Person_name, Player_id AS PlayerId, BirthDate,
38             MarketValue, Rating
39         FROM ConfManagement.vw_PlayerDetails
40         WHERE Club_id != @ClubId
41         AND (@Position IS NULL OR Position = @Position)
42         ORDER BY ' + @OrderByClause + '
43         OFFSET @Offset ROWS FETCH NEXT @PageSize ROWS ONLY;
44     ';
```

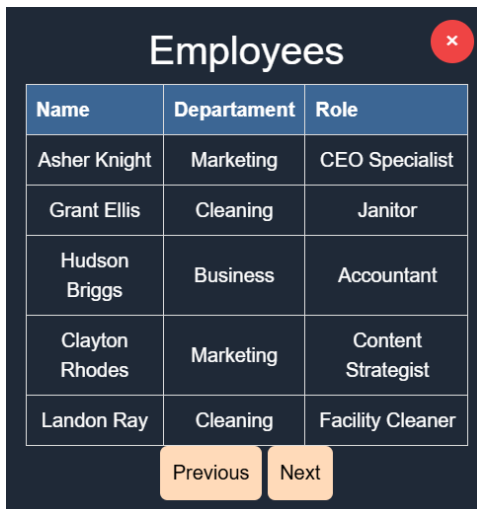
```

42      -- Execute with parameters
43      EXEC sp_executesql
44          @sql,
45          N'@ClubId INT, @Position NVARCHAR(2), @Offset INT, @PageSize
           INT',
46          @ClubId = @ClubId,
47          @Position = @Position,
48          @Offset = @Offset,
49          @PageSize = @PageSize;
50  END

```

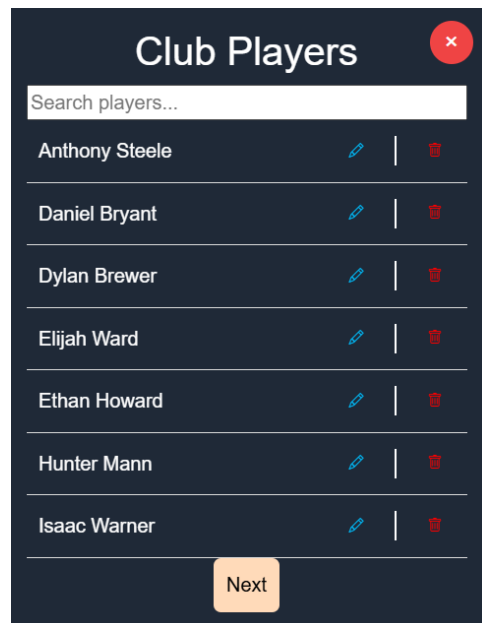
3.3 Gerir Funcionários

Para facilitar o processo de gestão, esta App permite ter um visão global dos membros que compõem o clube, sejam eles jogadores ou funcionários.



Name	Department	Role
Asher Knight	Marketing	CEO Specialist
Grant Ellis	Cleaning	Janitor
Hudson Briggs	Business	Accountant
Clayton Rhodes	Marketing	Content Strategist
Landon Ray	Cleaning	Facility Cleaner

Figure 6: Funcionários do clube



Club Players	
Search players...	
Anthony Steele	[edit] [delete]
Daniel Bryant	[edit] [delete]
Dylan Brewer	[edit] [delete]
Elijah Ward	[edit] [delete]
Ethan Howard	[edit] [delete]
Hunter Mann	[edit] [delete]
Isaac Warner	[edit] [delete]

Figure 7: Jogadores do clube

3.4 Update e Delete

Conforme ilustra a Figura 7, é possível fazer a atualização de dados (transitivos) de um jogador. Assim, dependendo da performance do jogador, o utilizador poderá atualizar a classificação do jogador e o valor de mercado. O *delete* de um jogador, por sua vez, é feito através de um "delete", pois a destruição de dados numa base de dados relacional pode comprometer a integridade dos dados nela contidos. Desta forma, preferiu-se adicionar um bit de estado a cada jogador, sinalizando assim se um jogador está ativo ou não.

```
1 ALTER PROCEDURE [dbo].[sp_soft_delete_player]
2     @PlayerId int
3 AS
4 BEGIN
5     -- SET NOCOUNT ON added to prevent extra result sets from
6     -- interfering with SELECT statements.
7     SET NOCOUNT ON;
8
9     UPDATE ConfManagement.Player
10        SET IsDeleted = 1
11        WHERE Player_id = @PlayerId;
12 END
```

4 Logout

Terminando as tarefas de gestão do clube, o utilizador pode efetuar um *logout*, e os dados armazenados na sessão são removidos automaticamente. Isso garante que, após o encerramento da sessão, não haja possibilidade de acesso não autorizado ao clube por meio de dados de sessão remanescentes, assegurando a integridade e a segurança da aplicação.

```
1 public async Task<IActionResult> Logout()
2 {
3     await HttpContext.SignOutAsync(); // Clears the authentication
4     cookie
5     HttpContext.Session.Clear(); // Optional: clears the session data
6     return RedirectToAction("Index", "Home");
7 }
```