

Universidade do Minho
Mestrado em Engenharia Informática
Projeto de Informática (2023/24)

HUB GVERREIRO

PROJETO 7 - SCBRAGA



Gonçalo Braz (a93178)



Tiago Silva(a93277)



Simão Cunha (a93262)



Gonçalo Pereira (a93168)



Hugo Fernandes (pg50419)



Marisa Soares (pg50643)



David Duarte (pg50315)

Braga, 24 de janeiro de 2024

Repositório GitHub: <https://github.com/realRunlo/BragaScoutingPlatform>

Conteúdo

1	Introdução	2
1	Contextualização	2
2	Prefácio do Projeto	2
2	Projeto	3
1	Introdução e objetivos	3
2	Primeira Componente	3
3	Segunda Componente	3
3	Requisitos	4
1	Requisitos não Funcionais	4
2	Requisitos Funcionais	6
4	Plano da solução	7
1	Arquitetura	8
2	Ciclo de trabalho	8
3	Solução final	9
5	Infraestrutura	10
1	Especificação	10
1.1	Base de dados	10
1.2	Máquina virtual	10
2	Critérios de escolha	10
3	Terraform (IaC)	11
4	Extra	11
6	ETL e Base de Dados	11
1	Base de dados	11
2	ETL	12
3	Extra	14
7	Microsoft Power BI	15
1	Competições	16
2	Competição	17
3	Jogo	18
4	Equipa	18
5	Jogador	19
6	Comparação de jogadores	19
7	Filtragem de jogadores	20
8	Conclusão e Trabalho Futuro	20
A	Anexos	21

1. Introdução

1 Contextualização

A unidade curricular de Projeto de Informática tem como objetivo a criação de um ponto de partida para a realização de projetos num contexto mais realista, intervindo diretamente - no caso dos projetos de empresa - com uma empresa que, fazendo o papel de cliente, apresenta uma proposta de um projeto para o grupo realizar. Deste modo, os alunos são incentivados para um ambiente que, embora já simulado em cadeiras anteriores - e ainda num espírito académico -, permite colocar em prática as aprendizagens sobre a planificação, discussão e desenvolvimento de um projeto com um cliente desde a sua génese e envolvendo maior responsabilidade por parte dos alunos.

Neste sentido, espera-se que, tendo como base uma descrição de um projeto de uma empresa, se estabeleça comunicação com esta de modo a criar um ciclo de desenvolvimento eficaz e que satisfaça o objetivo do cliente: desde uma fase inicial de levantamento de objetivos e requisitos; até ao planeamento e desenvolvimento da solução, mantendo sempre contacto com o cliente de modo cumprir o trabalho satisfazendo os desejos deste sempre que possível.

Os projetos de empresa foram divididos em duas componentes, sendo que ambas podiam variar, mas em geral a primeira envovia a generalidade do projeto, sendo a segunda uma tarefa mais orientada para a documentação, ou algum outro encargo complementar à primeira componente.

2 Prefácio do Projeto

O nosso grupo demonstrou interesse e foi encarregado do projeto proposto pelo **SCBraga** (projeto 7), intitulado, "*Hub Gverreiro*". Como contactos/coordenadores do lado do cliente, assim como interessados na utilização da ferramenta proposta, foram Luís Dias e Miguel do Carmo da equipa de análise de dados.

É de notar que o **SCBraga** já realizou antes projetos com a UC de Projeto de Informática, sendo que o atual tem a sua formação como uma espécie de continuação ou complemento do projeto realizado no ano passado. O anterior projeto, embora com um ciclo de trabalho semelhante e objetivos relativamente no mesmo contexto, apresenta-se concluído no seu intuito, assim, devido a mudanças nas ferramentas utilizadas e requisitos pedidos (que serão mais à frente apresentados), não serve como base ou para reaproveitamento no projeto deste ano. Não deixando de relevar, claro, o facto de que em certos aspetos que correram bem no trabalho anterior e se aplicavam no nosso, tenhamos tirado inspiração deste, como é o caso da utilização de *views* para diminuir a carga de computação do lado do *Power BI* (algo que exploraremos mais tarde); ou por outro lado quando vimos que havia espaço para melhorar, como é o caso de realizar as chamadas à API de forma paralela ao invés de corrotinas, de modo a permitir controlar mais facilmente a quantidade de pedidos durante horas de trabalho.

2. Projeto

1 Introdução e objetivos

O Sporting Clube de Braga é um clube de futebol de renome, não só em Portugal como internacionalmente, já com uma longa história que data desde 1921 e que tem demonstrado ao longo deste período e inclusive recentemente a razão para o seu renome, vindo a apresentar resultados notórios mesmo nas competições de mais alto nível no panorama internacional.

Este sucesso envolve várias facetas da modalidade, sendo uma destas a análise dos dados produzidos pelo desporto. Estes dados envolvem aspectos mais englobantes, como no caso do projeto proposto pelo SC Braga no ano passado, com estatísticas sobre as diferentes competições, tanto a nível nacional como global, a mais focadas, como dados sobre os diferentes jogadores, não só atualmente como no decorrer da sua carreira. É este último ponto que motiva o SC Braga para realizar a proposta do atual projeto, mais especificamente: a extração, limpeza e análise de dados de interesse para a área de *scouting*, e posterior criação de meios de visualização destes de forma interativa, útil e simples.

O projeto do ano passado envolvia, num traço geral, a criação de um conjunto de estatísticas que permitisse uma análise mais facilitada das diferenças entre diferentes competições e o desempenho do clube nestas. Do mesmo modo como o deste ano, esse projeto envolvia uma primeira fase de extração, limpeza e armazenamento destes dados, antes da sua posterior manipulação para a criação de *views*.

2 Primeira Componente

A primeira componente do projeto engloba a totalidade do ciclo para criação das *views* e as próprias, isto é: criação de *scripts* para recolha e limpeza de dados da API para armazenar numa base de dados externa; aplicação destas *scripts* de forma rotineira numa máquina externa para atualizar os dados com uma dada frequência; criação e otimização da base de dados mencionada; criação das *views* no ambiente *Power BI* especificamente.

3 Segunda Componente

A segunda componente é composta pela criação de um guia de utilização do projeto, i.e., explicação da informação disponibilizada nas diferentes *views* e o modo da sua utilização e navegação; assim como de algumas questões mais pertinentes fora da área do Power BI como no caso dos utilizadores gostarem de adicionar novas épocas para recolha de dados, ou variar a frequência com que as *scripts* colecionam os dados.

Este guia será entregue em simultâneo com o resto do projeto e num documento independente do atual.

3. Requisitos

Neste capítulo iremos enumerar e descrever os diferentes requisitos base do projeto. Alguns destes não foram pedidos invariantes ou essenciais, sendo que nos foi dada bastante liberdade por parte do SC Braga para tomar certas decisões, particularmente em questões mais técnicas; enquanto que outros pontos não totalmente fixos foram sendo moldados de acordo com o *feedback* dado. Estas exceções estão devidamente anotadas na listagem.

1 Requisitos não Funcionais

Requisito	O produto final para utilização do consumidor, i.e. as <i>views</i> , têm de ser criadas utilizando Power BI .
Justificação	Ferramenta familiar no ambiente da empresa e suficiente para a criação das <i>views</i> esperadas.

Tabela 3.1: Requisito não funcional 1

Requisito	A API a utilizar para a coleta dos dados necessários tem de ser WyScout .
Justificação	Ferramenta de trabalho dos próprios contactos do SC Braga, sendo que são dados com que eles já estão familiarizados e dispõe da informação necessária para o pretendido no projeto.
Nota	Como anteriormente mencionado, a API é distinta do projeto do ano anterior (sendo a do ano passado usada OPTA). A chave da API não é única para o projeto, sendo partilhada com os membros do SC Braga. A API permite até 12 pedidos por segundo e a chave disponibilizada dava acesso a todos os dados da API (existem diferentes níveis de acesso).

Tabela 3.2: Requisito não funcional 2

* Não concreto	
Requisito	As <i>views</i> finais devem ser interativas e intuitivas.
Justificação	O produto final tem como alvo não apenas utilizadores acostumados com gráficos e estatísticas, sendo que uma apresentação com visuais de apoio e de fácil utilização é importante.
Nota	<p>Embora satisfeitos com o trabalho realizado no ano passado, os nossos contactos apontaram que a questão visual foi uma questão que gostavam que o nosso grupo melhorasse.</p> <p>No trabalho do ano passado, a grande maioria dos dados é apresentada na forma de gráficos, sem grande apoio visual para criar um ambiente mais "digerível".</p> <p>A equipa de análise de dados do SC Braga aproveitou em conjunto com este pedido para demonstrar alguns exemplos/<i>mockups</i> de <i>designs</i> do estilo que gostariam de ver implementados.</p>

Tabela 3.3: Requisito não funcional 3

* Não concreto	
Requisito	Os tempos de carregamento das <i>views</i> , na sua totalidade ou parcialmente (certos elementos de computação mais demorada) não deve ser excessivamente demorada.
Nota	Pela própria descrição é evidente que esta condição não está explícita, sendo que como alvo final alvejamos para um tempo máximo de carregamento de elemento de 2 minutos (considerando condições de ligação à Internet normais). No <i>feedback</i> recebido, foi retirado que fundamental seria a informação apresentada ser relevante e correta, e os tempos de espera estarem dentro de uma margem aceitável (não mais do que alguns minutos).

Tabela 3.4: Requisito não funcional 4

Requisito	Os dados devem ser diariamente atualizados utilizando a API.
------------------	--

Tabela 3.5: Requisito não funcional 5

Requisito	A base de dados deve ter os dados das últimas 3 épocas de cada competição.
Justificação	As últimas 3 épocas de uma competição foram o limite de relevância ditado pelos contactos do SC Braga, sendo que isto normalmente corresponderá aos últimos 3 anos. Mais do que isso e as divergências no estado dos jogadores podem tornar-se muito notáveis.

Tabela 3.6: Requisito não funcional 6

Requisito	Os dados antigos (antes da antepenúltima da dada competição) devem ser anualmente limpos da base de dados, mantendo apenas os relevantes.
Justificação	De forma a manter os custos do servidor baixo e, simultaneamente os tempos de resposta deste aceitáveis, os dados devem ser mantidos apenas enquanto estes forem relevantes.

Tabela 3.7: Requisito não funcional 7

Requisito	A extração de dados, i.e. a parte de ETL, deve ser realizada fora das horas de trabalho - entendidas como sendo entre 8h-21h.
Justificação	Como mencionado anteriormente, a chave da API é de uso partilhado com a equipa da empresa, sendo que este projeto não deve impedir ou restringir o seu acesso a este recurso.

Tabela 3.8: Requisito não funcional 8

* Não concreto	
Requisito	Os recursos de <i>cloud</i> utilizados devem tentar reduzir os seus custos ao mínimo. Como guia foi sugerido os custos do projeto do ano anterior, que rondam por volta de 50 euros por mês.

Tabela 3.9: Requisito não funcional 9

Requisito	A base de dados deve conseguir atender pedidos de cerca de 6 utilizadores em simultâneo.
Justificação	Número de utilizadores relativo ao conjunto de utilizadores máximo expectável do projeto, no presente e futuro.

Tabela 3.10: Requisito não funcional 10

Certas competições foram explicitamente pedidas para terem os seus dados guardados, estas estão listadas nos anexos.

2 Requisitos Funcionais

Requisito	O projeto de <i>Power BI</i> final deve disponibilizar acesso aos dados: <ul style="list-style-type: none"> • Estatísticas de uma partida. • Estatísticas das equipas numa partida. • Estatísticas dos jogadores numa partida. • Estatísticas de uma época. • Estatísticas de uma equipa numa época. • Estatísticas de um jogador numa época.
Nota	As estatísticas específicas são extensas e serão portanto discutidas com maior profundidade numa secção posterior.

Tabela 3.11: Requisito funcional 1

Requisito	O projeto de <i>Power BI</i> final deve disponibilizar uma navegação hierárquica pelos dados: <ul style="list-style-type: none">• região → competição• competição → época• época → partida• partida → equipa• partida → jogador• equipa → jogador
------------------	--

Tabela 3.12: Requisito funcional 2

Requisito	O projeto de <i>Power BI</i> final deve ter uma opção de pesquisa direta por jogadores.
------------------	---

Tabela 3.13: Requisito funcional 3

Requisito	A pesquisa direta por jogadores deve proporcionar um conjunto de filtros, como: <ul style="list-style-type: none">• nome• competição• minutos jogados• valor de mercado• pé favorito• posição mais jogada
------------------	--

Tabela 3.14: Requisito funcional 4

Requisito	O projeto de <i>Power BI</i> final deve disponibilizar uma <i>view</i> para comparação direta entre dois jogadores. Esta comparação deverá envolver as suas estatísticas numa dada época.
------------------	---

Tabela 3.15: Requisito funcional 5

4. Plano da solução

Tendo em vista os objetivos propostos pelo SCB, assim como os requisitos levantados para o projeto, prosseguimos então para o desenho da abordagem que iríamos tomar para a sua

concretização. Este capítulo apresentará a arquitetura da solução, assim como uma exposição da divisão e ciclo de trabalho no grupo.

1 Arquitetura

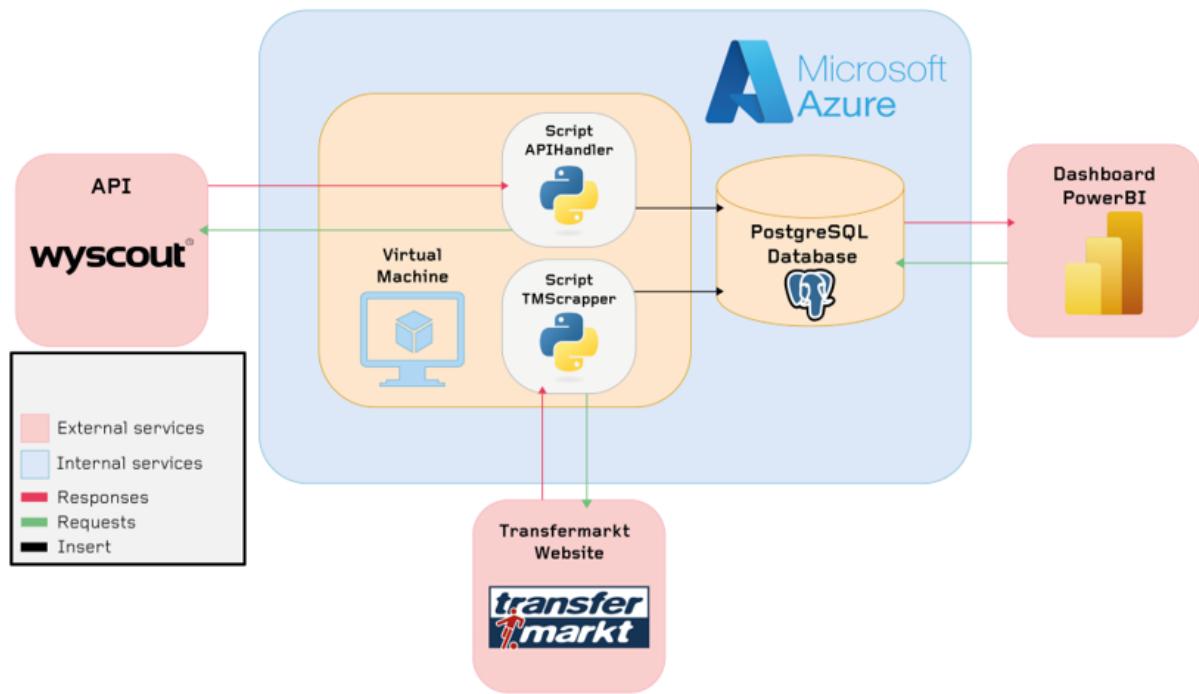


Figura 4.1: Arquitetura da solução

Podemos observar na figura 4.1 a arquitetura geral da solução. Utilizando este diagrama, podemos notar as 3 componentes principais da solução:

- **Aquisição de dados** : De forma a ser possível a criação das *views* pretendidas, é necessário uma primeira fase de obtenção, tratamento e armazenamento de dados a partir da API disponibilizada. Tal é realizado a partir de *scripts* Python que depois de os obter e tratar, enviam para armazenamento numa base de dados.
- **Projeto Power BI** : Como referido nos requisitos, a ferramenta utilizada para a criação das *views* pretendidas é o Power BI. Este cria as respetivas *views* utilizando os dados armazenados na base de dados.
- **Infraestrutura** : Com o âmbito de facilitar a rotina das *scripts*, assim como a disponibilização e sincronização dos dados entre os vários utilizadores do projeto, foi criada uma infraestrutura externa usando a plataforma *Azure*. Esta será composta por um máquinas virtuais que irá executar os *scripts* frequentemente e um servidor de base dados.

Cada uma destas componentes será discutida com mais detalhe nos capítulos seguintes.

2 Ciclo de trabalho

O processo de trabalho pode ser dividido entre a parte da comunicação com a empresa para averiguação dos objetivos e requisitos do projeto, tal como atualizações do seu estado; e o processo de desenvolvimento propriamente dito.

De forma a aumentar as chances de sucesso do projeto, assim como a satisfação do utilizador final, o ciclo de trabalho passou por um constante contacto com os contactos da parte do SC Braga, de modo a estes estarem sempre a par do estado do projeto e simultaneamente poderem dar *feedback* sobre aspectos desenvolvidos e outros que gostavam de ver realizados.

O desenvolvimento deste foi repartido em diferentes partes/áreas mais especializadas e distribuída pelos diferentes membros do grupo, de modo a criar uma divisão mais concreta das tarefas a realizar e ao mesmo tempo permitir a criação de equipas menores mas mais especializadas para tratarem das diferentes questões, relativamente isoladas, do projeto.

Esta divisão foi então:

- Componente de **ETL e Base de Dados**, responsável pela extração, limpeza e armazenamento de dados.
- **Cloud e DevOps**, abrangendo a questão da infraestrutura necessária, como o servidor da base de dados e a máquina virtual onde a atualização dos dados periódica seria realizada, e a automação de processos do desenvolvimento de forma a tornar mais simples e eficiente o fluxo de trabalho.
- **Projeto PowerBI**, que tem o papel de desenvolver a parte útil para o cliente.

3 Solução final

A solução final foi organizada na diretoria final da seguinte forma:

```
solução
├── cloud
│   ├── credentials_scripts
│   └── terraform
├── db
│   └── scripts
├── docs
│   └── segunda componente
├── etl
│   ├── exemplos
│   └── scripts
└── powerbi
    └── dashboards
```

Cada pasta de 1º nível é correspondente a uma das componentes, à exceção de *db* e *etl* que logicamente fazem mais sentido serem separadas.

5. Infraestrutura

1 Especificação

1.1 Base de dados

Características técnicas

- PostgreSQL flexible server
- 1vCore
- 2Gib RAM
- 32Gib storage

Por um período de 12 meses, para as características acima mencionadas não será cobrado qualquer valor. Após esse periodo o recurso passará a incorrer no custo establecido.

Custo 14,65€ /mês

1.2 Máquina virtual

Características técnicas

- Ubuntu 20.04
- Standard B1s
- 1vCore
- 1Gib RAM
- 30Gib storage

Custo 7,98€ /mês

2 Critérios de escolha

O PowerBI permite para bases de dados dois modos de carregamento de dados: importação e **direct query**. O modo de importação implica uma fase inicial de carregamento de todos os dados para cache na máquina do utilizador. Dependendo da quantidade de dados, esta operação pode ser extremamente demorada, tendo os nossos contactos do SCB reportado casos de esperas de 30+ minutos. Por outro lado, com os dados carregados, todas as operações do projeto são consequentemente mais rápidas. No caso de **direct query**, os dados são ao invés carregados à medida que são necessários nas *views*, permitindo uma inicialização do projeto mais rápida. Embora tal pareça assemelhar-se à realização de pedidos diretos a uma base de dados, é limitado no facto em que estes pedidos não podem ser filtrados, sendo portanto sempre pedidos de tabelas completas. No final optámos por **direct query** pois para os casos de uso que nos foram apresentados, como por exemplo a partilha do projeto a equipas além do compartimento de dados trazia para pessoas fora da área um carregamento inicial cria uma barreira de utilização;

ou em situações para utilização urgente, a necessidade de um potencial carregamento de dados demorado é indesejável.

Na escolha da base de dados, baseamos a nossa decisão em dois critérios: custo, suporte à funcionalidade de **direct query** no powerBI para essa base de dados e capacidade de disponibilização dos dados. Esta abordagem visa garantir uma infraestrutura eficiente, equilibrando o custo operacional com a capacidade de atender aos requisitos específicos de cada componente.

A escolha do PostgreSQL flexible server para a base de dados reflete não apenas o custo acessível, mas também a capacidade de suportar as necessidades **direct query** no PowerBI e a competência para provisionar os dados com a cadênciia necessária para este projeto.

Na escolha da máquina virtual baseamos a nossa decisão nos dois seguintes critérios: custo e desempenho. Optamos por uma solução que atende aos requisitos mínimos do projeto, garantindo uma solução ótima face à utilização prevista.

3 Terraform (IaC)

O Terraform é ferramenta de Infraestrutura como Código (IaC) que disponibiliza uma linguagem declarativa para provisionar infraestrutura nos diversos fornecedores de soluções em nuvem como Azure, AWS, entre outros. No contexto deste projeto, foram desenvolvidos módulos específicos para o aprovisionamento da infraestrutura mencionada anteriormente. Esses módulos desempenham um papel crucial ao facilitar a portabilidade e replicação da infraestrutura, promovendo assim a reutilização eficiente de código e a consistência na implementação.

4 Extra

Para tornar o desenvolvimento mais ágil, foram implementadas workflows para atualização automática dos ficheiros relevantes do projeto na máquina virtual, i.e. scripts de ETL.

Além disso, de forma a controlar os custos dos recursos, foram introduzidos alertas de orçamento no grupo de recursos com o intuito de não exceder um limite mensal de 30 euros.

6. ETL e Base de Dados

Neste capítulo, será discutido o trabalho realizado sobre a componente de extração, tratamento e armazenamento de dados. Começaremos por uma explicação do trabalho realizado sobre a base de dados, seguido por um estudo do processo de ETL.

1 Base de dados

Para a criação do esquema da base de dados é importante realçar o ambiente com que trabalhamos. Como referido no capítulo 5, o servidor de base de dados escolhido é dedicada a bases de dados **PostgreSQL**. PostgreSQL é um gestor de bases de dados relacional baseado em SQL o que permitiu uma familiarização rápida neste ambiente.

O desenvolvimento do esquema da base de dados teve ainda em conta 3 pontos: os objetivos e requisitos propostos pelo cliente; os dados disponibilizados pela API *WyScout*; e as necessidades apresentadas pela equipa do projeto Power BI à medida que este era desenvolvido. Este último

corresponde principalmente a questões de otimização ou reestruturação da base de dados para melhorar o desempenho do projeto.

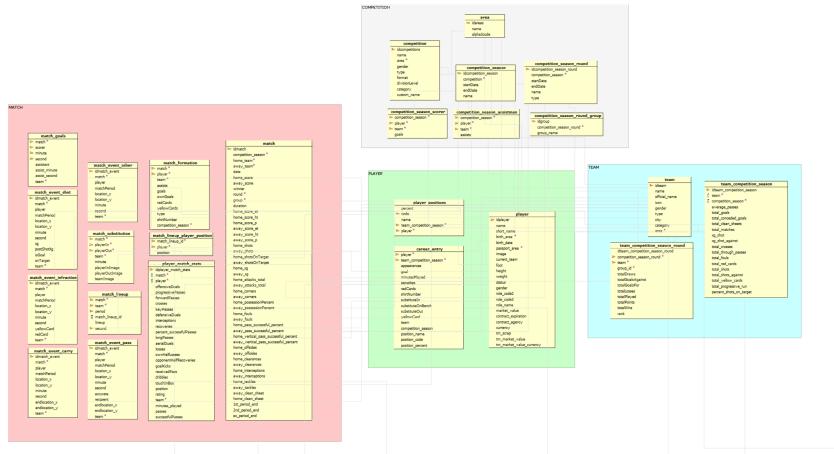


Figura 6.1: Diagrama da base de dados

De forma sucinta, o esquema da base de dados presente na imagem 6.1 pode ser dividido em 4 partes principais:

- **Estrutura das competições**: correspondente a tabelas com dados que permitem a estruturação hierárquica das competições. Ex.: áreas, competição, época, grupo, época.
 - **Equipas** : dados das equipas para as diferentes épocas e competições em que participaram
 - **Jogadores** : correspondente aos dados dos jogadores pessoalmente e da sua carreira
 - **Partidas** : partidas das diferentes épocas, os seus resultados e dados complementares

Além das tabelas e estrutura base desenhada, a criação de *Views* na base de dados também foi um elemento essencial do projeto visto que através de realizações de testes de desempenho foi notório que a transferência da carga computacional para a base de dados sempre que possível diminuía consideravelmente os tempos de carregamento no Power BI. Isto acopla-se ao facto de para a obtenção dos resultados de algumas destas *Views*, várias tabelas eram necessárias e, como foi explicado no capítulo 5, o Power BI iria ter de pedir todas estas tabelas e aguardar a sua receção antes de realizar a computação de junção e filtragem pretendida. Em alguns casos esta transferência da computação para a base de dados diminui o tempo de carregamento no Power BI em vários minutos.

De forma a manter o desempenho da base de dados aceitável ao longo do tempo, com a introdução de novas épocas para as várias competições, optou-se pela divisão desta em duas bases de dados. Estas duas bases de dados têm o mesmo esquema mas propósitos diferentes, sendo a principal encarregada de armazenar as e épocas mais recentes de cada competição e a outra base de dados, **BD de arquivo**, guarda os dados das épocas mais antigas. Assim, permitimos que dados que são apenas pontualmente utilizados não impactem o desempenho da base de dados principal mas sendo ao mesmo tempo acessíveis.

2 ETL

A área de **Extraction, Transformation e Loading** de dados, representa todo o processo de obtenção dos dados desejados, a sua transformação ou manipulação para limpeza ou geração de novos dados e por fim o armazenamento do resultado, que neste caso será numa base de dados externa.

A nossa fonte de dados deste projeto é a API WyScout. A partir deste conseguimos informação relevante sobre as competições, as épocas destas, as equipas e jogadores que competiram e os detalhes das partidas. Além disso, alguns dados extra sobre os jogadores são possíveis de obter, como a sua carreira.

Para a concretização deste projeto criamos um conjunto de scripts. A linguagem escolhida foi *Python* (versão 3.10) devido à nossa familiaridade com esta linguagem e à sua versatilidade. Além disso, o *bottleneck* neste processo são os pedidos realizados à API, sendo que a natureza mais lenta desta linguagem relativamente a C por exemplo não é tão relevante.

As scripts de criadas foram:

- **scouting_data** : Script principal do projeto. A partir de um ficheiro JSON que dispõe dos competições e épocas desejadas, percorre cada competição e para esta cada época, fazendo pedidos à API para obter os dados desejados. Durante o ciclo são realizadas também certas verificações usando a base de dados de modo a mitigar inconsistências na API; ex.: por vezes em partidas de divisões mais baixas, a API devolve *id* de jogadores que não têm entrada na própria API, o que resultaria em erros de chaves estrangeiras na BD. Ele tem dois modos de utilização : povoação completa da base de dados, que segue o fluxo explicado a partir do ficheiro JSON; modo de update, em que atualiza apenas as épocas mais recentes das competições da base de dados num dado período de tempo - função disponibilizada pela API -, o que diminui o tempo de execução.
- **tm_script** : Uma característica relevante no processo de *scouting* de jogadores é o seu valor de mercado. O WyScout é no entanto, neste aspeto, bastante limitado, não apresentado na maioria dos casos valores atualizados, ou valor qualquer. De modo a aumentar a utilidade do projeto, resolvemos então utilizar uma outra fonte de dados para resolver esta questão, nomeadamente o site *TransferMarkt*, que é uma das proeminentes fontes de dados de futebol. Este script realiza então, para todos os jogadores armazenados na base de dados, uma procura pelo seu valor de mercado neste site. Tal é realizado através do conhecimento do seu clube atual de acordo com os dados armazenados e *scraping* utilizando similaridade de nomes, uma vez que não possuímos os *ids* de referência do TransferMarkt e os nomes neste site nem sempre são 1:1 com o WyScout.
- **data_requests_updater** : Este script trata de dois processos: atualizar os ficheiros com as competições pretendidas de forma a listar as épocas mais recentes; verificar na base de dados se existem épocas desatualizadas (antes da antepenúltima época) e, em caso positivo, os dados dessas épocas são transferidos para a base de dados de arquivo e eliminados da base de dados principal.
- **seasons** : Para facilitar a adição manual de novas competições e épocas, visto não ser esperado do usuário final proficiência no uso da consola de comandos e do formato JSON, implementámos um script para facilitar esta tarefa. O script tem dois modos principais: por comandos e por GUI.

A forma de utilização destes scripts está documentada no guia correspondente à segunda componente.

Todos estes scripts, à exceção do *seasons*, foram desenhados com o propósito de serem corridos de forma rotineira.

Destes, o principal é o **scouting_data** e a frequência intenedida é diária usando o modo *update*. No entanto, tendo em conta que existem épocas do ano em que as equipas dos SCB necessitam de total acesso à API, entendemos que seria útil esta script não correr durante esses períodos. Nestes casos, quando a script é ativada novamente, ela irá fazer um update total das épocas. Para cada época, a recolha total dos dados demora cerca de 7 minutos, podendo variar dependendo da sua dimensão. Para cada época, são realizados 5 pedidos gerais, 1 pedido por cada equipa,

4 pedidos por cada partida e 4 pedidos por cada jogador, assumindo condições ideias, i.e. não sendo detetadas inconsistências na API que resultem em pedidos extra. Assumindo uma época dentro do comum com: 18 equipas, 500 jogadores e 300 partidas; seriam realizados 4023 pedidos (+- 5.5 minutos casos se utilize a capacidade máxima de 12 pedidos por segundo da API, considerando apenas a realização de pedidos). Realizar esta tarefa de forma sequencial tornaria este script inutilizável, demorando mais de uma hora a correr para cada época. Deste modo, tal como o grupo do projeto do ano passado, tivemos de optar por uma rota diferente. No projeto do ano passado, a rota escolhida foi a utilização de programação assíncrona. No nosso caso, nós optamos por computação paralela utilizando *threads* por três motivos:

- Nos testes realizados, o uso de *threads* foi ligeiramente mais rápido do que o modo assíncrono;
- O uso de *threads* permite controlar mais facilmente a quantidade de pedidos realizados durante as horas de trabalho;
- Uso de thread independente para a conexão com a base de dados, não bloqueando todo o programa quando uma thread de extração de dados faz um pedido à base de dados;

Nos resultados obtidos, a extração de dados de uma época demora +- 7 minutos, havendo portanto um aumento significativo no desempenho da script. Tendo em conta que no total esperamos guardar no mínimo 60 épocas (7 horas), nota-se também a importância do modo de atualização apenas das épocas atuais. Realça-se também que um segundo *bottleneck* é criado pelo grande volume de inserções que se realiza na base de dados no decorrer da execução. Resolvemos este problema utilizando um *thread worker* para uma conexão da base de dados, que vai realizando as tarefas numa *stack*. Além disso, de forma a diminuir o uso de memória RAM da máquina virtual (o que permite a escolha de uma opção menos custosa), os valores extraídos pelas *threads* são guardados temporariamente em ficheiros enquanto que não são tratados pelo *worker* que os irá inserir na base de dados.

O **tm_script** irá correr semanalmente porém, pode ser executada apenas à medida que for necessário visto não realizar pedidos à API.

O **data_requests_updater** correrá apenas mensalmente, sendo apenas útil para a deteção do término e começo das épocas.

3 Extra

É importante realçar que a escolha dos dados a extrair, i.e. as competições de interesse, podem ser facilmente modificadas através de ficheiros JSON. Exemplos destes estão presentes na pasta competições dentro da diretoria ETL. Estão já definidos JSON para um conjunto de competições definidas em A.1.

As frequências de execução das diferentes scripts foi definida utilizando a ferramenta *contrab* num sistema Unix.

Guias para a modificação destes dois estão presentes na segunda componente deste projeto.

7. Microsoft Power BI

Foi-nos pedido que elaborássemos *dashboards* recorrendo à ferramenta Microsoft Power BI¹. São baseadas em *drill-through*, que é uma funcionalidade que permite aos utilizadores analisar dados em diferentes níveis de detalhe, que permite navegação entre páginas com diferentes tipos de informação e dados. Para isso é necessário haver relações apropriadas entre tabelas, de modo a que sejam apresentados os dados que o cliente pretende. É importante referir que cada ligação entre diferentes tabela afeta o contexto em que os dados são expostos nas *dashboards*. Serão mais à frente explicados as decisões relativas a esta afirmação.

De modo a facilitar a visualização dos dados na plataforma, dividimos as temáticas dos dados em 7 páginas:

- **Competições:** Visualização das diferentes competições de futebol, permitindo a seleção do país e do logótipo da mesma;
- **Competição:** Visualização da tabela classificativa de uma liga de futebol, lista de jogos efetuados até ao momento (i.e. até à última atualização da base de dados) e lista de *rankings* de melhores marcadores², jogadores com mais assistências³, *clean-sheets*⁴ e *rating*⁵ médio na liga. É permitida a seleção de uma equipa ou de um jogo, redirecionando o utilizador para o elemento pretendido. Existe também a possibilidade de escolher uma de três épocas (atual e as duas anteriores), pelo que a seleção de uma outra época levará à visualização de dados dessa referida época desportiva - por exemplo, a tabela classificativa irá conter outras equipas;
- **Jogo:** Visualização dos 11 iniciais de cada equipa e substitutos utilizados na partida, com elementos visuais representativos do número de golos, número de cartões amarelos recebidos, número de cartões vermelhos recebidos, substituições e *rating* obtido no jogo. Além disso, podemos observar dados estatísticos do jogo, tal como o número de remates e a posse de bola de cada equipa; dados estatísticos de cada jogador participante no jogo, como o número de passes e o número de duelos ofensivos; cronograma da partida, onde é possível observar os eventos mais importantes na partida com a anotação do minuto em que aconteceu;
- **Equipa:** Visualização do plantel de uma equipa, jogos efetuados na sua liga doméstica; estatísticas da equipa nessa competição, tal como *xG*⁶ e o número de golos marcados e a visualização de *rankings* de melhores marcadores, jogadores com mais assistências e melhor *rating* médio. Existe também a possibilidade de escolher uma de três épocas (atual e as duas anteriores), pelo que a seleção de uma outra época levará à visualização de dados dessa referida época desportiva - por exemplo, o plantel irá mudar;

¹Disponível em <https://www.microsoft.com/pt-pt/power-platform/products/power-bi?market=pt>

²Considera-se um marcador como um jogador que marca um golo numa partida de futebol

³Uma assistência consiste no passe de um jogador A para um jogador B, onde o segundo marca um golo.

⁴Consideraremos uma *clean-sheet* o número de jogos em que um guarda-redes (titular ou que substitui alguém numa partida) sem sofrer qualquer golo.

⁵O *rating* de um jogador consiste numa pontuação entre 0 e 10 que reflete o seu desempenho na partida.

⁶Considera-se a métrica *xG* como o número de golos expectáveis, de acordo com alguns dados estatísticos.

- **Jogador:** Visualização de dados biográficos de um jogador, como a idade, melhor pé e valor de mercado (segundo o website Transfermarkt⁷); dados estatísticos na liga comparando com os valores médios da competição divididos em 3 secções: passe, defesa e ataque; histórico de partidas efetuadas com visualização do *rating* obtido em cada uma; gráfico ilustrativo das posições ocupadas na época e respetivas percentagens. Existe também a possibilidade de escolher uma de três épocas (atual e as duas anteriores), pelo que a seleção de uma outra época levará à visualização de dados dessa referida época desportiva - iremos observar os dados estatísticos desse jogador nessa época;
- **Comparação de jogadores:** Comparação entre dois jogadores, ambos selecionáveis através de filtros definidos. São disponibilizados gráficos como o *radar chart* que permitem a comparação de métricas futebolísticas de ambos os jogadores;
- **Filtragem de jogadores:** Pesquisa avançada de jogadores através da aplicação de filtros como o valor de mercado, país da liga em que joga, melhor pé, idade e posição, permitindo o redirecionamento para a página de um jogador selecionado pelo utilizador.

Nas próximas secções, iremos explicar detalhadamente as decisões e pormenores técnicos de cada uma das páginas elaboradas em Microsoft Power BI. Depois de elaborarmos todas as referidas páginas, conseguimos publicar este documento PowerBI na versão *web* de forma a ser acessível a muita gente, desde que estas tenham permissões. No entanto, as *scripts* visuais Python não funcionam corretamente, uma vez que não conseguimos que a versão *web* consiga usar o interpretador com as devidas bibliotecas.

1 Competições

Nesta página (ver figura A.2), é permitida ao utilizador a procura de uma liga de um determinado país. Para tal, foi criado um *slicer* - elemento nativo do Power BI que permite a seleção de dados através da aplicação de um filtro - utilizando as colunas calculadas *CountryNames* e *CountryFlags* da tabela *scouting competition*. O PowerBi permite criar colunas calculadas, guardando elementos de uma outra tabela que são referenciados por uma chave (um número inteiro identificador) na tabela original, desde que essa relação entre as duas tabelas esteja ativada na aplicação. De forma a podermos observar as ligas de um determinado país, foi criada uma tabela onde são representados o logótipo da mesma e o seu nome - estes dados podem ser obtidos da tabelas da base de dados *scouting competition* e *scouting competition_season*. Escolhemos representar estes dados em formato tabular em vez de, por exemplo, usar um *slicer* pois foi a forma que o grupo encontrou (e replicada em várias páginas da aplicação) para o utilizador poder ser redirecionado corretamente da página de obtenção de uma competição para a página de uma competição ao incluir os identificadores das duas tabelas utilizadas nesta página (*scouting competition* e *scouting competition_season*), mas omitindo o seu conteúdo ao utilizador.

É importante referir que a API utilizada no projeto não disponibiliza os logótipos das suas competições nem as bandeiras de países, pelo que houve a necessidade de criarmos duas medidas DAX (*Data Analysis Expressions*) que, usando o campo *custom_name* da tabela *scouting competition*, obteríamos o URL do logótipo da competição e com o recurso ao campo *name* da tabela da base de dados *scouting area*, será possível obter o URL de uma bandeira de um dado país - note que, devido a não ser possível obter o URL tendo em conta o nome de alguns países (por exemplo Costa do Marfim ou República Democrática do Congo), alguns URLs tiveram de ser obtidos de forma *hard-coded*.

⁷Disponível em <https://www.transfermarkt.pt/>

2 Competição

Depois de selecionada uma competição na página anterior, o utilizador conseguirá visualizar a página detalhada de uma determinada liga (ver figura A.3). É disponibilizada uma tabela classificativa de equipas. Além disso, são disponibilizadas duas métricas fornecidas pela API e que foram adicionadas à tabela classificativa: xG e xGA ⁸. Todos estes dados foram retirados da tabela da base de dados `scouting_team_competition_season_round`.

De forma a permitirmos o redirecionamento para a página de uma determinada equipa, teve de ser adicionada uma coluna com IDs das respetivas, escondendo o seu conteúdo para o utilizador (técnica já replicada na página **Competições**).

Nesta página também disponibilizamos uma lista de jogos do campeonato, disponibilizando a data, grupo/eliminatória da competição onde a partida está inserida, nomes e logótipos das equipas, resultado do jogo e a indicação se a partida terminou ao fim dos 90 minutos (**Tempo Regulamentar**), ao fim dos 120 minutos (**Prolongamento**) ou se foi a **Grandes Penalidades**. Para tal, foram usadas várias colunas calculadas da tabela da base de dados `scouting_match` como **ScoreBoard** - caso uma partida acabe ao fim dos 90 minutos, o resultado da partida estará no formato [número de golos da equipa visitada] - [número de golos da equipa visitante] de acordo com o número de golos durante esse intervalo de tempo; caso uma partida acabe ao fim dos 120 minutos, o resultado da partida estará no formato [número de golos da equipa visitada] - [número de golos da equipa visitante] de acordo com o número de golos durante esse intervalo de tempo; caso uma partida vá para grandes penalidades, o resultado da partida estará no formato [número de golos da equipa visitada] - [número de golos da equipa visitante] de acordo com o número de golos durante o prolongamento, mas terá um "*" do lado da equipa vencedora nas grandes penalidades.

De forma a permitirmos o redirecionamento para a página de uma determinada partida, teve de ser adicionada várias colunas com identificadores da partida (coluna `id_match` de `scouting_match`), da época de uma competição (coluna `id_competitionseason` de `scouting_competition_season`) e de uma determinada competição (coluna `idcompetitions` de `scouting_competition`), escondendo os seus conteúdos para o utilizador.

Por fim, também são disponibilizadas, sob a forma de *tabs* (ou separadores), *rankings* dos melhores marcadores, melhores assistentes, guarda-redes com mais *clean-sheets* e melhor *rating* médio. De modo a disponibilizarmos estes dados de forma rápida para o utilizador do PowerBI, foi necessário criar *views* na base de dados em vez de seleccionarmos os dados desejados nas tabelas da BD. A tabela 7.1 resume quais as *views* usadas (terá o mesmo nome que as tabelas em PowerBI) para criar um determinado *ranking* e quais os dados disponibilizados em cada um:

Ranking	View da BD	Dados disponibilizados
Melhores marcadores	<code>scouting_competition_season_scorers_view</code>	Logótipo da equipa, Nome, imagem e idade do jogador, Número de golos marcados, Número de jogos efetuados pelo jogador
Mais assistências	<code>scouting_competition_season_assistmen_view</code>	Logótipo da equipa, Nome, imagem e idade do jogador, Número de assistências efetuadas, Número de jogos efetuados pelo jogador
<i>Clean-sheets</i>	<code>scouting_competition_season_gk_stats</code>	Logótipo da equipa, Nome, imagem e idade do GR, Número de <i>clean-sheets</i> , Número de jogos efetuados pelo GR
<i>Rating</i> médio	<code>scouting_player_team_competition_season_rating</code>	Logótipo da equipa, Nome, imagem e idade do jogador, <i>Rating</i> médio, Número de jogos e minutos efetuados pelo jogador

Tabela 7.1: Tabelas utilizadas para criar os *rankings* da página **Competição**

De forma a podermos redirecionar o utilizador para a página do jogador, foi adicionada a cada

⁸Métrica correspondente ao número esperado de golos sofridos até ao momento da competição

um dos *rankings* anteriormente citados um identificador de jogador (coluna `idplayer` da tabela `scouting_player`) ocultando o seu conteúdo para o utilizador.

Também são visíveis nesta página alguns filtros de grupos/eliminatórias e de fases de campeonatos que, quando aplicados, todas as dashboards serão atualizadas e alguns filtros de datas, grupos e barras de pesquisa, permitindo a visualização dos dados mais desejados pelo utilizador.

Por fim, foi criado um *slicer* com as indicações das épocas da competição selecionada disponíveis na base de dados, onde uma seleção num destes elementos remete o utilizador para a visualização dos dados da liga na época selecionada.

3 Jogo

Na página de jogo (ver figura A.5), podemos observar vários *scripts* visuais Python que mostram os 11 iniciais de cada equipa e respetivos suplentes utilizados e indicação dos golos e assistências efetuados, ocorrência de cartões amarelos e vermelhos, minutos de substituições e *rating* obtidos. Todos estes dados foram retirados da tabela da base de dados `scouting_matchFormation_extended_rating - view` calculada para efeitos de *performance* - e utilizou-se a biblioteca Python MPLSoccer⁹ para desenhar o campo de futebol. Uma vez que uma equipa de futebol pode mudar o seu esquema tático quantas vezes as que o seu treinador desejar, apenas foi desenhado o esquema tático do minuto 1 da primeira parte.

Também serão disponibilizados três separadores referentes a estatísticas da partida, estatísticas de cada jogador na partida e um cronograma sobre todos os eventos ocorridos na partida. Os dados das estatísticas da partida são representados sobre a forma de barras empilhadas e são oriundas da tabela `scouting_match`. Os dados das estatísticas de um jogador são representados por duas tabelas do PowerBI, onde clicando numa entrada da tabela de listagem de jogadores que participam na partida, levará a que sejam apresentados as respetivas estatísticas na segunda tabela. Por último, foi elaborado um *script* visual Python que mostra ao utilizador as ocorrências de todos os eventos da partida de forma mais intuitiva para o utilizador.

4 Equipa

Na página de equipa (ver figura A.4), são apresentadas as informações relativas a uma equipa selecionada na página **Competição**. É possível alterar a época pretendida de forma a que se possa observar as alterações dos dados incluídos na página em tempo real sob a forma de um *slicer*. Estão representadas em formato tabular os detalhes mais pertinentes do plantel da equipa na época selecionada como nacionalidade ou nome. Acrescentamos também filtros de posição, melhor pé e valor de mercado, caso o utilizador pretenda que lhe sejam apresentados jogadores com as características selecionadas - todos estes dados são retirados da tabela `scouting_career_entry`. Nessa tabela de jogadores, é permitido que o utilizador selecione uma entrada na tabela para redirecionar para a página de jogador.

É também disponibilizado um histórico de jogos em que a equipa em questão participou desde o início da época até ao último jogo ou apenas até a data atual, caso esteja selecionada a época actual, cujos dados são obtidos da tabela `scouting_match`. Neste elemento gráfico, o utilizador consegue ser redirecionado para a página de um jogo ao clicar numa determinada partida. Além disto, numa outra *tab*, será possível observar estatísticas da equipa na liga. Juntamente com estes dados (todos eles retirados da tabela da base de dados `scouting_team_competition_season`, são apresentados *bullet charts* que permitem comparar algumas estatísticas com os dados médios da liga.

Por último, são disponibilizados sob a forma de *tabs* (ou separadores), *rankings* dos melhores marcadores, melhores assistentes e melhor *rating* médio dessa equipa. Uma vez que são *rankings*

⁹Documentação disponível em <https://mplsoccer.readthedocs.io/en/latest/index.html>

semelhantes aos criados na página **Competição**, estes dados foram retirados da mesma forma à efetuada na tabela 7.1.

5 Jogador

Na página de jogador (ver figura A.6), estão presentes as principais características do jogador, tais como nome, idade, peso, altura, valor de mercado, pé preferido, posição, número de jogos em que participou e respetivos minutos, a liga e divisão em que joga e respetivo país, número de golos e assistências, número de cartões amarelos e vermelhos e o *rating* médio. Está também disponível um *slicer* com as equipas a que o jogador pertenceu durante a época selecionada e um histórico dos jogos em que participou em formato tabelar. É possível visualizar *bullet charts* com estatísticas de ataque, defesa e passe, que estão divididas sob a forma de *tabs*. Cada valor nesse gráfico representa a média do jogador na liga na época selecionada e o traço de controlo a preto corresponde à média de todos os jogadores com a mesma posição na mesma liga na época selecionada. Foi necessário criar diferentes *views* na BD para representar corretamente os valores nestes gráficos. Todos os dados estatísticos usados nesta página estão contidos nas tabelas `scouting_career_entry` e `scouting_career_entry_league_stats`. Nesta página também é permitido que o utilizador selecione a época e consequentemente observar as alterações de certos dados em tempo real, por exemplo a posição e percentagens, o *slicer* que contém as equipas que frequentou nessa época e as estatísticas nos *bullet charts*.

Para o jogador, possuímos na base de dados várias tabelas distintas com diferentes tipos de conteúdo. Nesta página, são necessárias no mínimo quatro tabelas: a tabela `player` contém os dados mais básicos de um jogador, a tabela `career_entry` é relativa a informações do jogador numa determinada época e competição, a tabela `player_match_stats` corresponde às estatísticas do jogador em todas as partidas que jogou e a tabela e a tabela `player_positions` contém as posições e correspondentes percentagens numa equipa numa determinada época. Para que seja possível expor certos dados, são necessárias corretas relações entre as diferentes tabelas e por vezes foi necessário ajustar essas mesmas ligações para que obtivéssemos as informações adequadas. Isto possibilita que consigamos criar diferentes colunas calculadas para obter diferentes tipos de informação no contexto em que se insere em cada situação. Por exemplo, na tabela `career_entry` são criadas diferentes colunas para adquirir o conteúdo de outras tabelas no contexto em que se insere. Como por exemplo, a coluna calculada `época` presente nessa tabela, apenas vai mostrar as épocas das quais o jogador em questão participou. Nesta página, é possível visualizar um campo com diferentes posições, onde são indicadas as respetivas percentagens de ocupação de posição na época inserida. Para isso, utilizamos uma *script* Python de acordo com o conteúdo da tabela `player_positions`.

6 Comparação de jogadores

Na página de comparação de jogadores (ver figura A.7), aparecem vários filtros representados em *slicers* para que se possa escolher dois jogadores. É possível escolher a competição, país da competição, equipa, época, posição ou então pesquisar pelo próprio nome do jogador. Os *slicers* da ferramenta Power BI permitem apagar os filtros selecionados, sendo isso uma opção viável caso o utilizador pretenda pesquisar por diferentes jogadores após já ter selecionado algum. A página de comparação de jogadores é muito semelhante à do jogador, mas aqui podemos observar dados e estatísticas de dois jogadores distintos com a finalidade de os comparar. Para isso temos diferentes representações de informação e dados. Igual à pagina jogador, as características principais de cada jogador estão do lado direito e esquerdo da página respetivamente. Existe também um campo onde indica as posições com maior percentagem de onde o jogador joga, que como foi referido anteriormente, foi realizado com auxílio de uma *script* Python. Para caracterizar os dados de cada jogador, utilizamos diferentes tabelas na base de dados, que

posteriormente no Power BI são duplicadas de modo a ser possível expor os elementos de cada jogador, ou seja são criados dois contextos diferentes. Esta página pode ser acedida através de um botão presente em todas as páginas denominado *Comparação de jogadores*.

7 Filtragem de jogadores

Na página de procura avançada de jogadores (ver figura A.8), é possível descobrir jogadores usando uma série de filtros como o país da liga em que joga, clube, idade, valor de mercado, *rating* médio, minutos jogados, posição mais frequente da época atual e o seu melhor pé. Para tal, foram criados *slicers* para cada um dos filtros referidos e todos estes dados são oriundos da tabela `scouting current_players`, que consiste numa *view* que contém todos os jogadores guardados na base de dados da época atual de uma competição também guardada na BD. Tal como a página de comparação de jogadores, esta página pode ser acedida através de um botão presente em todas as páginas denominado *Filtragem de jogadores avançada*.

8. Conclusão e Trabalho Futuro

Consideramos o trabalho realizado uma experiência enriquecedora tanto a um nível prático, onde tivemos a oportunidade de aprofundar o uso de tecnologias comuns na indústria, como é o caso do aprovisionamento de recursos externos, tecnologias novas como foi o caso do Power BI e essencialmente, uma oportunidade de entender o tipo de projetos a nível empresarial na área do desporto. Agradecemos ao Luís e ao Miguel, nossos contactos do Braga, pela disponibilidade, paciência e ajuda no decorrer do projeto.

Na nossa perspetiva e pelo feedback que fomos recebendo ao longo do desenvolvimento, consideramos que conseguimos realizar um trabalho que satisfaz os objetivos pretendidos. No final conseguimos, utilizando recursos com capacidades limitadas, a construção de um sistema de captura e fornecimento de dados para um conjunto de Views intuitivas, interativas e úteis. Toda a funcionalidade principal do projeto está descrita no guia respetivo à segunda componente.

Naturalmente, compreendemos que há espaço para aperfeiçoar a solução, em especial no que toca às estatísticas disponibilizadas. Algumas vias para estas são: utilização de Machine Learning para cálculo de novas estatísticas; apresentações utilizando estatísticas mais pormenorizadas, ex.: mapa de passes (tal não foi possível devido ao volume de dados e limitações de filtração do Power BI).

A. Anexos



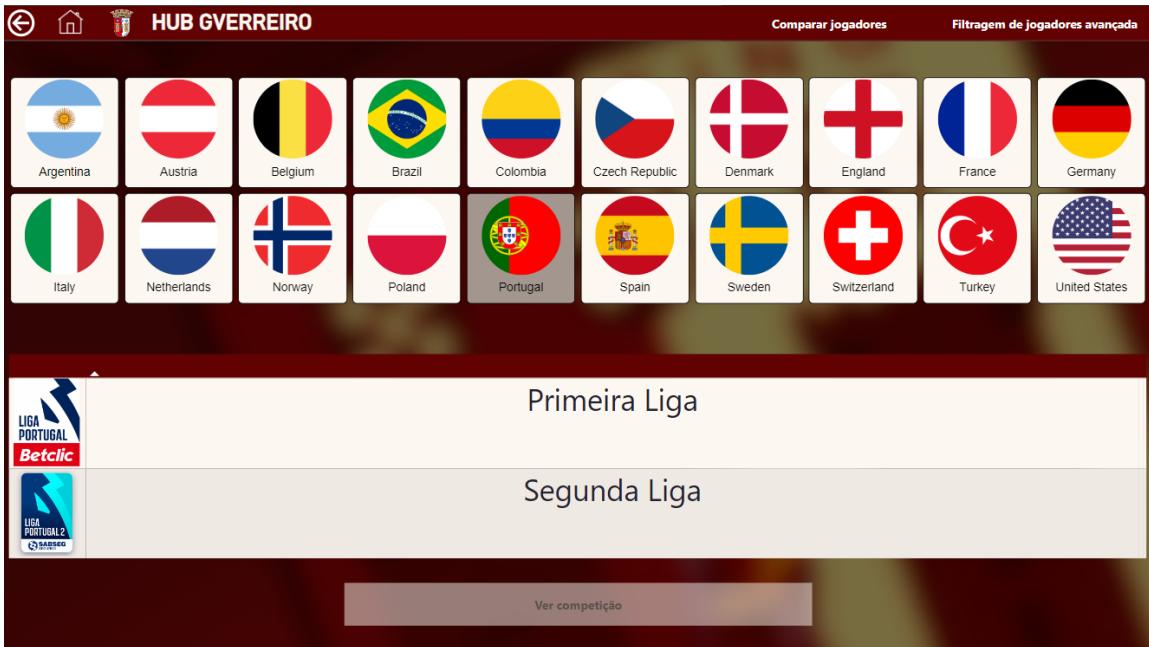


Figura A.2: Página de exemplo Competições

Filtrar classificação										Filtrar jogos							
Por fase		Por grupo/eliminatória								Por intervalo de datas		Por grupo/eliminatória		Por equipa			
Regular Season		Regular Season								Data		Grupo/Eliminatória		Visitado		Visitante	
#	Equipa	J	V	E	D	GM	GS	DG	XG	XGA	P						
1	Sporting CP	18	15	1	2	45	19	26	31,30	14,31	46						
2	Benfica	18	14	3	1	37	11	26	39,36	18,18	45						
3	Porto	18	13	2	3	30	12	18	28,51	19,00	41						
4	Sporting Braga	18	11	3	4	42	26	16	35,29	20,36	36						
5	Vitória Guimarães	18	11	3	4	31	19	12	24,02	20,55	36						
6	Moreirense	18	8	5	5	24	23	1	18,34	16,27	29						
7	Farense	18	7	3	8	26	24	2	20,88	32,56	24						
8	Famalicão	18	5	7	6	18	22	-4	18,71	23,46	22						
9	Boavista	18	5	5	8	26	33	-7	18,64	26,35	20						
10	Arouca	18	5	4	9	25	26	-1	24,06	21,42	19						
11	Gil Vicente	18	5	4	9	29	31	-2	19,78	29,50	19						
12	Casa Pia AC	18	5	4	9	19	26	-7	14,78	20,24	19						
13	Estrela Amadora	18	4	6	8	18	27	-9	18,16	27,49	18						
14	Portimonense	18	5	3	10	18	38	-20	19,23	27,64	18						
15	Estoril	18	5	2	11	32	34	-2	21,34	29,61	17						
16	Rio Ave	18	3	7	8	19	28	-9	24,42	18,63	16						
17	Vizela	18	2	7	9	18	32	-14	21,25	24,01	13						
18	Chaves	18	3	3	12	18	44	-26	20,15	28,62	12						

Figura A.3: Página de exemplo Competição



Figura A.4: Página de exemplo Equipa



Figura A.5: Página de exemplo Jogo



Figura A.6: Página de exemplo Jogador

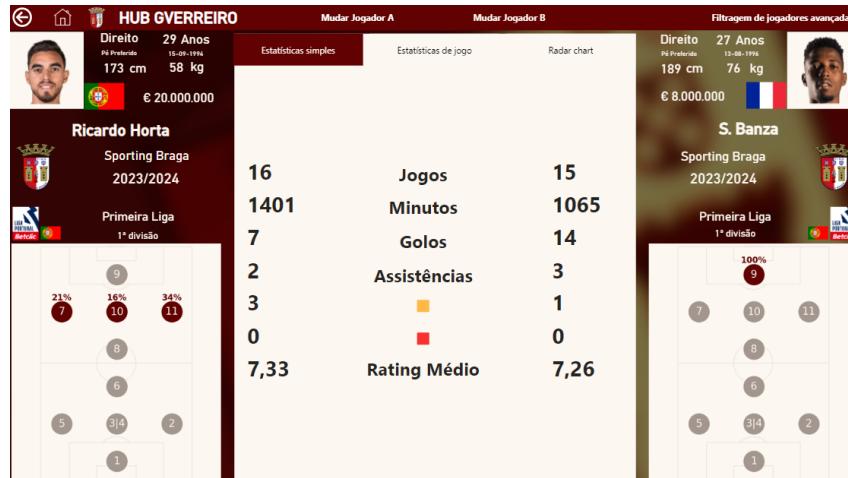
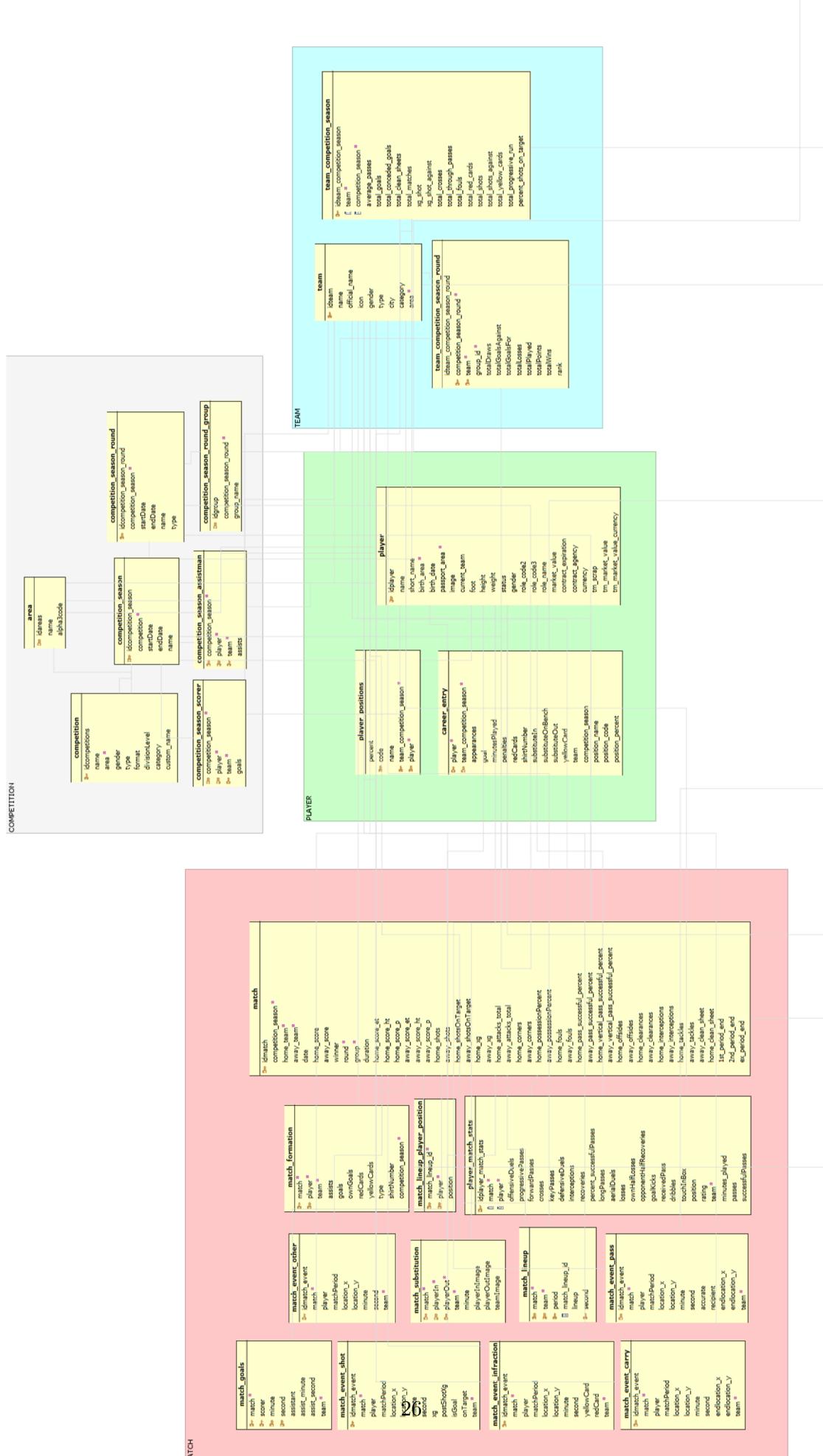


Figura A.7: Página de exemplo Comparação de jogadores

Competição		Equipa		Idade		Posição		Melhor pé		Competição	
	2. Bundesliga		Premier League	19	22	9	9	Ambos	Direito	Eredivisie	1397
						10	2			Eredivisie	1377
						11	6			Eredivisie	761
						5	7			Eredivisie	742
						8	9			Eredivisie	209
										Primeira Liga	146
										Eredivisie	839
										Eredivisie	675

Figura A.8: Página de exemplo Filtragem de jogadores



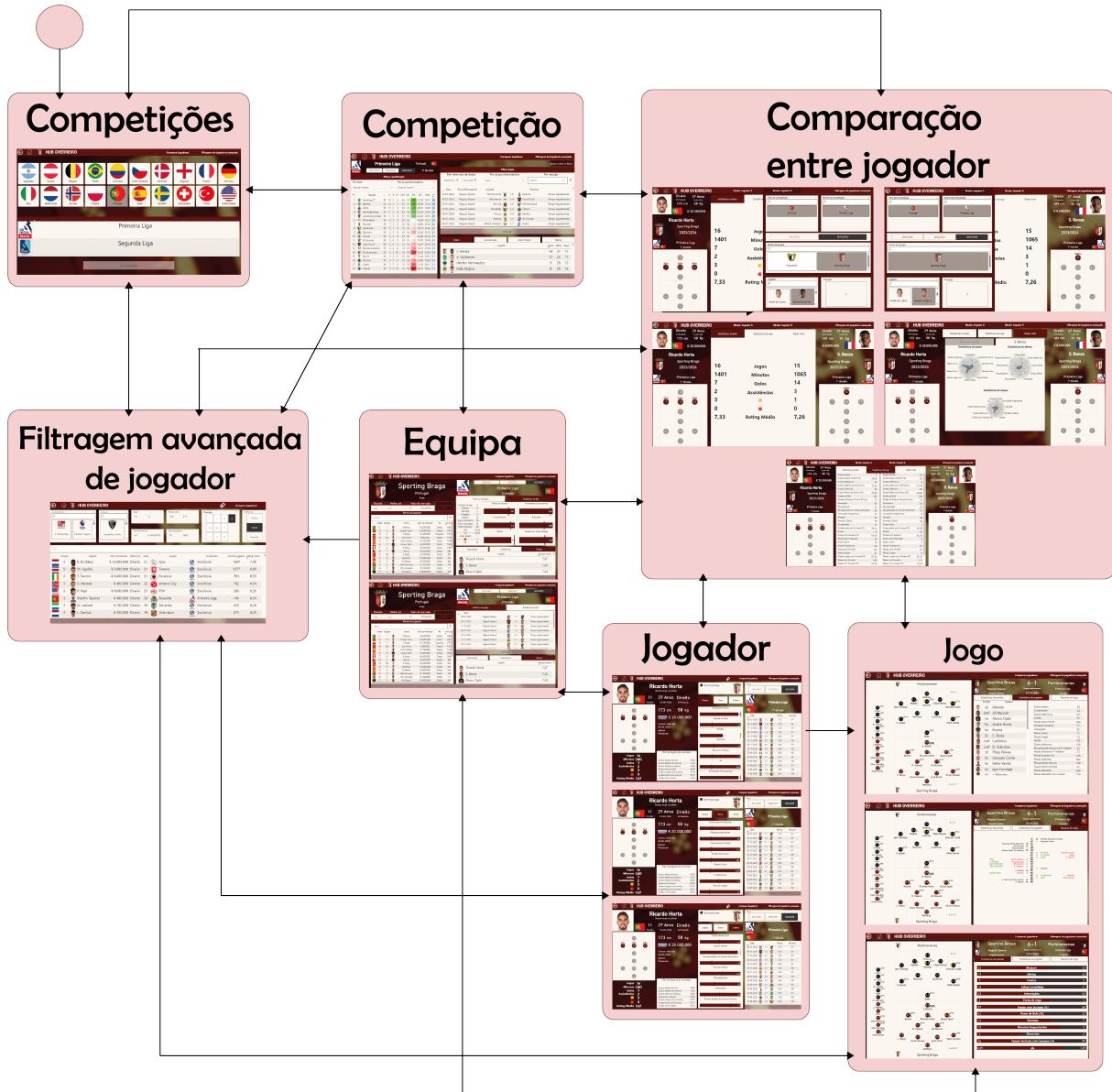


Figura A.10: Diagrama de fluxo de atividade. Note que o início da aplicação é a página Competições e todas as páginas podem ser redirecionadas para Competições através de um ícon no canto superior esquerdo