

Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

**Metrificação e análise de carreiras de
jogadores de futebol por meio de um
sistema computacional**

Lorenzo Bertin Salvador

Monografia Final

MAC 499 — Trabalho de
Formatura Supervisionado

Supervisora: Prof^a. Dr^a. Kelly Rosa Braghetto

São Paulo
2023

O conteúdo deste trabalho é publicado sob a licença CC BY 4.0
(Creative Commons Attribution 4.0 International License)

Agradecimentos

Aos meus pais pelo apoio incondicional durante toda a minha trajetória.

Aos meus irmãos, em especial ao Emanuel, que é o idealizador por trás deste projeto e quem me motivou a criar este sistema.

A todos os meus queridos amigos, da época da Carlitos, Bandeirantes e também da USP.

A minha orientadora Kelly, pelo acompanhamento e ajuda ao longo do ano.

A todos os professores que passaram pela minha vida, em especial a minha mãe.

A Philipp Weiß e toda a equipe do Transfermarkt pela permissão de uso dos dados do site para este trabalho.

Resumo

Lorenzo Bertin Salvador. **Metrificação e análise de carreiras de jogadores de futebol por meio de um sistema computacional**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2023.

Este trabalho descreve a implementação de um sistema que calcula pontuações para as carreiras de jogadores de futebol, baseando-se em uma fórmula matemática preexistente. O fluxo do sistema consiste em extrair automaticamente as informações dos atletas, times e competições a partir de fontes públicas na internet, armazená-las em um banco de dados orientado a documentos, e então aplicar a fórmula. Além disso, o software também usufrui de um banco de dados orientado a grafos, que permite computar informações envolvendo as relações inter-jogadores, que vão para além das pontuações individualmente calculadas, provenientes da aplicação da fórmula. Para tal, são utilizadas métricas próprias da estrutura dos grafos, como busca pelo caminho mínimo, centralidade de nós, detecção de comunidades, entre outras. Então, com o banco de dados populado, inclusive com os *scores* computados, o último desafio é a visualização dessas informações. Para atingi-lo, a ferramenta Apache Superset é empregada, já que ela fornece um ambiente de criação e visualização de gráficos e tabelas, que acompanha a evolução dos dados e que lida com as questões estéticas dos elementos gráficos. Ademais, a última particularidade do sistema é que ele pode ser utilizado por pessoas que não têm grande conhecimento em computação, já que todas as etapas da interação usuário-software podem ser realizadas por meio de interfaces gráficas, sejam elas preexistentes ou criadas especificamente para este trabalho. Por fim, após descrever todas as etapas envolvidas na criação do software, esta monografia também exhibe os resultados da aplicação da fórmula em mais de 700 jogadores, 1000 times e 500 campeonatos, e mostra possíveis análises a serem realizadas levando em conta não só a semântica computacional da implementação do sistema, como também o próprio ponto de vista futebolístico dos resultados alcançados.

Palavras-chave: Metrificação de futebolistas. Bancos de dados. Orientação a documentos. Orientação a grafos. Visualização de dados.

Abstract

Lorenzo Bertin Salvador. **Metrification and analysis of football players' careers using a computational system.** Capstone Project Report (Bachelor).
Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2023.

This work describes the implementation of a system that calculates scores for football players' careers based on a preexisting mathematical formula. The system flow consists of automatically extracting information about athletes, teams, and championships from public sources on the internet, storing them in a document-oriented database, and then applying the formula. Furthermore, the software also benefits from a graph-oriented database, which allows computing information involving inter-player relationships, which goes beyond the individually calculated scores resulting from the formula application. To this end, specific metrics from this data structure are used, such as the shortest path finding, node centrality, community detection, etc. Then, with the populated database, including the scores for each player, the last challenge is the visualization of all this data. To accomplish this, the Apache Superset tool is used, as it provides an environment for creating and visualizing graphs and tables, that keeps up with the database evolution and which deals with the aesthetic issue of graphical elements. Next, the last system particularity is that it can be used by users with low computational knowledge since all the steps in the human-software interactions can be carried out through graphical interfaces, either preexisting or created specifically for this work. Last, after describing all the stages needed in implementing the system, this monography also displays the results of applying the formula to more than 700 players, 1000 teams, and 500 tournaments, and shows possible analyses that could be carried out, taking into account not only the computational semantics of this software implementation but also the football point of view of the achieved results.

Keywords: Footballers' metrification. Databases. Document-oriented. Graph-oriented. Data visualization.

Lista de Abreviaturas

API	Application Programming Interface
BD	Banco de dados
CLI	Command-line Interface
CSV	Comma-separated Values
GUI	Guided User Interface
HTML	HyperText Marking Language
JSON	JavaScript Object Notation
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

Sumário

1	Introdução	1
2	Conceitos	3
2.1	Futebol	3
2.1.1	Estatísticas individuais dos jogadores de futebol	3
2.1.2	Divisão dos campeonatos pelo mundo	4
2.1.3	A temporada	4
2.2	Computação	5
2.2.1	Bancos de dados e sistemas gerenciadores	5
2.2.2	Grafos	7
3	Banco de Dados Operacional	10
3.1	MongoDB – orientação a documentos	10
3.2	Esquema do banco de dados	11
4	Extração de Dados	15
4.1	Fonte A: <i>Transfermarkt</i>	16
4.1.1	Observações acerca da qualidade e correteude da extração	18
4.1.2	Interação entre o BD e o extrator	18
4.2	Fonte B: extração manual	19
5	O Cálculo da Pontuação	21
5.1	Coeficientes de ajuste	21
5.2	Componente individual	22
5.3	Componente de conquistas	24
5.4	Componente de impacto	24
5.4.1	Impacto agudo	24
5.4.2	Impacto de longo prazo	26
5.4.3	Impacto final	27

5.5	Pontuação final	27
6	Banco de Dados de Grafo	28
6.1	A escolha do SGBD Neo4J	28
6.2	Conversão MongoDB → Neo4J	29
6.3	Métricas próprias da estrutura do grafo	31
6.3.1	Maior menor caminho (sem custo)	31
6.3.2	Componentes conexas	32
6.3.3	Quantidade de triângulos	33
6.3.4	K-1 cores	34
6.3.5	Grau de centralidade	34
6.3.6	Máximo de influência	35
6.4	Relação pontuação - grafo	36
6.4.1	O algoritmo	36
6.4.2	Os resultados	37
7	Interface de Usuário	41
7.1	Inserção de dados estáticos com o MongoDB Compass	41
7.2	Visualização e interação com o grafo no Neo4J	43
7.3	Visualização e manipulação dos dados de jogadores com Apache Superset	44
7.3.1	A escolha da ferramenta	45
7.3.2	Superset e MongoDB	46
7.3.3	Gráficos e tabelas	47
7.3.4	Customização de consultas	50
7.3.5	Parametrização de consultas	52
7.4	Inserção de dados dinâmicos de jogadores, times e campeonatos via interface customizada	54
7.4.1	Objetivo detalhado	54
7.4.2	Flask e Apache Superset	55
7.4.3	Interação com o sistema	56
7.4.4	Resultado	58
8	Discussão	62
8.1	Demografia do banco de dados	62
8.1.1	Campeonatos	62
8.1.2	Times	64
8.1.3	Jogadores	64
8.2	Pontuações	68

8.3	Análise específica por jogador	71
8.4	Análise específica por temporada	73
9	Conclusão	75
	Referências	77

Capítulo 1

Introdução

O futebol é o esporte mais popular do mundo e está presente na vida de milhões de pessoas, dentro e fora do Brasil. Os jogadores quase sempre são os protagonistas e os valores envolvidos no mercado de transferências, durante e após todas as temporadas, são estratosféricos. A importância dada para determinado futebolista, que justifica seu valor de mercado, pode ser baseada em diversos elementos, sendo os principais as participações em gols (gols e assistências) e os títulos conquistados. Além disso, as discussões entre apaixonados pelo esporte são sempre em volta de comparações para decidir os melhores jogadores de cada clube, seleção, temporada ou até mesmo da história.

Desse contexto, surge a ideia de avaliar a qualidade individual dos jogadores. Para isso, existem algumas maneiras já consolidadas dentro do mundo do futebol. As principais são os dois prêmios anuais chamados de “Bolas de Ouro”, um da FIFA (Federação Internacional de Futebol Associado) [FIFA, 2023] e o outro da revista francesa *France Football* [Football, 2023]. O problema dessas premiações é que seus critérios não são sistemáticos e podem variar ao longo dos anos, gerando incoerências e falta de embasamento para as decisões tomadas.

Nesse aspecto, existem diversos trabalhos científicos que buscam contrapor essa subjetividade à medida que utilizam as próprias estatísticas dos atletas para chegar em conclusões mais precisas, inclusive em esportes diferentes do futebol, como o trabalho de *Koenigsberg et al. (2020)*, que trata do golf, e também o de *Xia et al. (2018)*, que trata do basquete e futebol americano. Para o desenvolvimento deste projeto, a fonte mais relevante é o artigo de *E. P. Salvador et al. (2022)*, que propõe uma fórmula matemática para calcular a qualidade da carreira completa de um jogador de futebol, abrangendo tanto jogadores da atualidade como também os que já penduraram as chuteiras.

A ideia deste projeto foi implementar um sistema [L. B. Salvador, 2023] que tem a aplicação dessa fórmula como objeto central, ou seja, dado um jogador como entrada, o resultado produzido pelo sistema é sua pontuação. Para isso, foram planejados quatro componentes centrais e um complementar para compor

o sistema. Os quatro centrais são os seguintes: extrator de dados, banco de dados operacional, calculadora da fórmula e uma interface de gerenciamento e de visualização dos dados. O componente auxiliar é um banco de dados de grafos, que basicamente enriquece a análise das informações fornecendo uma perspectiva diferente da fórmula para a avaliação das carreiras dos jogadores. Esses componentes serão descritos com detalhes nos próximos capítulos do texto. Além disso, a partir de agora, todas as referências ao “sistema” ou ao “software”, fazem alusão ao código-fonte contido em [L. B. Salvador, 2023](#), que foi apelidado de **ModoK**.

Sendo assim, é possível definir três grandes objetivos para este trabalho: i) criação de um sistema computacional capaz de aplicar a fórmula proposta por [E. P. Salvador et al. \(2022\)](#); ii) produção de uma análise comparativa entre os resultados obtidos a partir da fórmula e alguma outra maneira de se mensurar a performance de um jogador em comparação com outros futebolistas; iii) possibilidade de que usuários que não têm conhecimento em computação utilizem o sistema, de modo que profissionais de outras áreas, como a Educação Física ou o Jornalismo, por exemplo, consigam gerenciar a aplicação e interagir com os dados.

As principais técnicas utilizadas durante o desenvolvimento deste sistema foram as seguintes: *web scraping*¹ (ou raspagem de dados), durante a fase de extração das informações; análise de rede e outras metrificações utilizando a estrutura dos grafos; *queries* nos bancos de dados por meio de diversas linguagens de consultas e, por último, multiprocessamento, aplicado durante implementação do servidor web. Já com relação às ferramentas, as mais relevantes foram: MongoDB, para o banco de dados orientado a documentos; Neo4J, para o banco de dados orientado a grafos; Apache Superset, para a visualização dos grafos e Flask, para a implementação de um servidor web. Tanto as técnicas quanto as ferramentas serão descritas em detalhes no restante desta monografia.

A divisão dos conteúdos dos próximos capítulos será feita da seguinte forma: o Capítulo 2 expõe o conhecimento prévio necessário para o entendimento do trabalho; o Capítulo 3 descreve o banco de dados operacional; o Capítulo 4 mostra como os dados foram extraídos da web; no Capítulo 5, a fórmula para calcular a pontuação da carreira de um jogador é descrita em detalhes; então, o Capítulo 6 discorre sobre o banco de dados de grafos e as métricas provenientes dessa estrutura; em seguida, o Capítulo 7 descreve e expõe todas as interfaces envolvidas no sistema e, por fim, os Capítulos 8 e 9 são responsáveis por discutir os resultados do trabalho e concluir a monografia, respectivamente.

¹ *Web scraping* é a técnica utilizada para coletar automaticamente dados presentes em páginas web.

Capítulo 2

Conceitos

Apesar de pouquíssimo conhecimento sobre futebol ser realmente mandatório para entender o funcionamento e objetivo do sistema, algumas peculiaridades podem requerer certo contexto com relação à organização desse esporte pelo planeta e também a algumas nomenclaturas mais específicas. Por outro lado, do ponto de vista computacional, também há alguns elementos-chaves que são importantes de serem introduzidos. Por esse motivo, este capítulo se propõe a descrever e exemplificar alguns conceitos menos conhecidos do futebol e os mais primordiais da computação, que serão úteis para compreensão total do trabalho por parte do leitor.

2.1 Futebol

Primeiramente, serão descritas cada uma das métricas individuais que determinam a performance de um jogador ao longo de um jogo ou campeonato. Depois, será abordada a divisão dos campeonatos ao longo do mundo. Por fim, as especificidades de uma temporada do futebol serão discutidas.

2.1.1 Estatísticas individuais dos jogadores de futebol

- **Gols:** O maior momento de uma partida de futebol, é a pontuação de um jogo. Em geral, jogos não tem mais de 5 gols, mas há exceções. Disputas de pênaltis (após o fim da partida) não são consideradas.
- **Assistências:** O passe para o companheiro fazer o gol. É importante notar que nem todo gol tem uma assistência, já que existem os gols de pênalti, faltas e também os gols em que o último passe é muito anterior ao gol e, portanto, irrelevante para sua marcação.
- **Hat-tricks:** O *hat-trick*, ou triplete, é o evento no qual um jogador faz três ou mais gols em um jogo, sem considerar gols durante disputas de pênaltis (após o fim da partida).

- Pontos por jogo: Uma vitória equivale a três pontos, empate um ponto e derrota nenhum ponto. Após um campeonato, soma-se a quantidade de pontos e divide-se pelos pontos disputados (três vezes o número de partidas).
- *Clean Sheets*: Contabiliza a quantidade de jogos que o time disputou sem tomar gols. Disputas de pênalti não são levadas em conta.
- Pênaltis defendidos: Quantidade de pênaltis durante os jogos que um goleiro defende (inclui prorrogações, mas não disputas de pênaltis).

2.1.2 Divisão dos campeonatos pelo mundo

Os campeonatos de futebol profissionais são repartidos em dois grandes grupos: os de clubes e os de seleções. Com relação aos torneios de seleções, em geral, são divididos em qualificatórios (ex.: eliminatórias para a Copa do Mundo), continentais (ex.: Copa América) e mundiais (ex.: Copa do Mundo). Já os campeonatos de clubes são divididos em mundiais (ex.: Mundial de Clubes da FIFA), continentais (ex.: Copa Libertadores), nacionais (ex.: Campeonato Brasileiro), locais (ex.: Campeonato Paulista) e alguns (poucos) qualificatórios (ex.: pré-Libertadores).

Em geral, essas divisões são muito semelhantes em todos os países e continentes, com variações nos números de clubes em cada campeonato e na importância atribuída a cada um deles. Outra característica que pode mudar é o formato do campeonato. Existem dois formatos base para os torneios de futebol profissional: mata-mata e pontos corridos. Pontos corridos são campeonatos em que cada partida vale o mesmo que as outras do ponto de vista de pontuação, ou seja, não há uma final e o time campeão é o que acumulou mais pontos no decorrer do torneio. Um exemplo de pontos corridos é o Campeonato Brasileiro, que ocorre ao longo de boa parte do ano. Os mata-matas, também chamados de Copas, por outro lado, são campeonatos em que a cada disputa um dos times é eliminado, até que sobram dois times para disputarem a final. A Copa do Brasil, por exemplo, é um torneio deste estilo. É importante notar que há torneios que misturam os dois formatos, como é o caso da Copa do Mundo, que é iniciada com grupos de quatro seleções onde os dois times que melhor pontuarem contra os outros de mesmo grupo passam para a próxima fase, que é disputada no formato mata-mata, ou seja, eliminatório.

2.1.3 A temporada

A temporada é o nome atribuído ao ano do futebol, ou seja, é o período do ano em que os campeonatos acontecem. Por razões históricas e também geográficas, há dois tipos de temporadas no mundo futebolístico: i) No Brasil e na maioria dos países do hemisfério sul, os campeonatos adotam a temporada que segue o ano do calendário, ou seja, um jogo que acontece dia 28/02 e outro que ocorre dia 01/12 fazem parte da mesma temporada; ii) Na Europa e em outros países do hemisfério norte, a temporada inicia na metade do ano e vai até

a metade do ano seguinte, logo, um jogo dia 23/02 e outro dia 03/08 do mesmo ano fazem parte de temporadas diferentes. Deste modo, os campeonatos ao redor do mundo podem seguir qualquer uma das temporadas, mas em geral, há uniformidade intra-continental. Torneios mundiais, como o Mundial de Clubes e a Copa do Mundo podem apresentar diferenças nas datas de acordo com o país sede, mas, em geral, a Copa segue o modelo i), já que a Copa do ano X ocorre integralmente no ano X, enquanto o Mundial segue o modelo ii), visto que a disputa dos campeões continentais de certo ano ocorre no início do ano seguinte.

2.2 Computação

Com relação aos conhecimentos computacionais necessários para acompanhar esta monografia, destacam-se dois: os bancos de dados (além de seus respectivos sistemas gerenciadores) e os grafos.

2.2.1 Bancos de dados e sistemas gerenciadores

Nos dias atuais, é possível dizer que a imensa maioria da população mundial interage frequentemente com um ou mais bancos de dados, ainda que sem saber. Nesta seção, os conceitos básicos acerca do assunto serão apresentados e os diferentes tipos de BDs (bancos de dados) exemplificados, bem como suas implicações neste trabalho.

Um banco de dados é uma coleção de dados, e um dado é qualquer elemento que possa ser gravado e que tenha algum sentido implícito (Elmasri e Navathe, 2010). Essa definição é ampla, mas é justamente por isso que existem diversos tipos de modelos de dados usados na implementação de bancos de dados, cada um com um objetivo principal e um caso de uso ideal. Os principais exemplos de modelos de dados são os seguintes: relacional, orientado a objetos, chave-valor, orientado a documentos, orientado a colunas e orientado a grafos. Outro conceito que está atrelado ao de banco de dados é o do SGBD (Sistema Gerenciador de Bancos de Dados), que é o pedaço de software capaz de concretizar um banco de dados, ou seja, que permite um usuário interagir com os dados, alterando, inserindo, removendo ou consultando um ou mais registros armazenados. Como exemplos de SGBDs é possível citar o *PostgreSQL*, *MongoDB*, *Neo4j*, *MySQL*, etc.

Para que a interação entre usuário e SGBD ocorra, é necessário haver alguma linguagem capaz de realizar as operações requisitadas pelo usuário, sem que haja ambiguidade e que vise a maior performance possível. Cada SGBD, portanto, pode implementar sua própria linguagem, chamadas de linguagens de consulta. A linguagem mais conhecida, que é baseada em cálculo e álgebra relacional (Elmasri e Navathe, 2010) e é implementada por diversos SGBDs, é a *SQL (Structured Query Language)*, mas existem outras, como *CYPHER* e *MQL (MongoDB Query Language)*.

O último conceito básico de bancos de dados que vale ser mencionado é a diferença entre os BDs relacionais e os não relacionais. Os relacionais são mais tradicionais, e baseiam-se em uma maneira rígida de representar os dados: tabelas bidimensionais chamadas de relações (Garcia-Molina *et al.*, 2008). Em uma dimensão estão as colunas da tabela, que representam os diferentes atributos de uma relação, e na outra dimensão estão as linhas, que representam instâncias dessa relação. BDs relacionais usufruem da linguagem SQL, já descrita anteriormente, e que foi idealizada e otimizada especificamente para esse tipo de estrutura de dados.

Para exemplificar, considere uma relação chamada de **PESSOA**. Os atributos dessa relação podem ser nome, idade e CPF. Um banco de dados que contém apenas essa relação é ilustrado na Figura 2.1.

	Nome	Idade	CPF
1.	João	33	11111111
2.	Maria	20	909090909
3.	Lionel	36	66666666

Tabela 2.1: Ilustração da relação **PESSOA**. Contém 3 atributos e 3 registros.

Em contra-partida, os bancos de dados não relacionais, conhecidos também como *NoSQL*, não possuem essa estrutura rígida. Por outro lado, diferentemente dos relacionais, ainda não há nenhuma linguagem de consulta padronizada para os SGBDs NoSQL. Os exemplos mais relevantes de BDs não relacionais para este trabalho são os BDs orientados a documentos e os orientados a grafos.

Um banco de dados orientado a documentos é dividido em coleções, e cada coleção contém documentos semanticamente similares. Esses documentos contém atributos (possivelmente aninhados), e podem ser representados em formatos de dados semiestruturados como o *XML* (Extensible Markup Language) ou o *JSON* (JavaScript Object Notation). É possível fazer um paralelo entre as relações de um BD relacional e as coleções de um orientado a documentos, mas a principal diferença é que os documentos não precisam seguir uma estrutura pré-definida, diferentemente dos registros de uma relação, que devem respeitar o esquema do BD (Sadalage e Fowler, 2012). Dessa maneira, ao longo do tempo, novos documentos inseridos no banco podem apresentar uma estrutura diferente dos anteriores, sem que isso prejudique o armazenamento dos documentos antigos (Elmasri e Navathe, 2010). Na Figura 2.1, é exemplificada uma coleção chamada de **PESSOA**, que contém dois documentos.

```

{"PESSOA":
  [
    {
      "nome": "Joao",
      "idade": 33,
      "cpf": "11111111",
      "passaporte": "4932049024"
    }
  ]
}

```

```

    },
    {
      "nome": "Maria",
      "idade": 20,
      "cpf": "909090909"
    }
  ]
}

```

Programa 2.1: Exemplo de coleção contendo dois documentos

Os bancos de dados orientados a grafos serão explicados a seguir.

2.2.2 Grafos

Um grafo é uma estrutura matemática que contém uma coleção de vértices e arestas, ou nós e relacionamentos, respectivamente. Uma aresta liga dois vértices, e um passeio é uma maneira de sair de um nó e chegar em outro, por meio dos relacionamentos presentes no grafo.

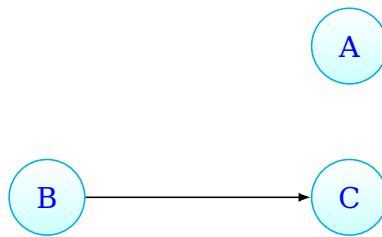


Figura 2.1: Exemplo de um grafo contendo 3 vértices e 1 aresta

Um grafo direcionado é aquele em que cada aresta só pode ser percorrida em um sentido. Na Figura 2.2, a aresta que liga A com B só pode ser percorrida nesse sentido ($A \rightarrow B$), já que o grafo é direcionado. Um grafo não direcionado não distingue o sentido das arestas, então toda aresta pode ser percorrida nos dois sentidos, como é possível notar na Figura 2.3

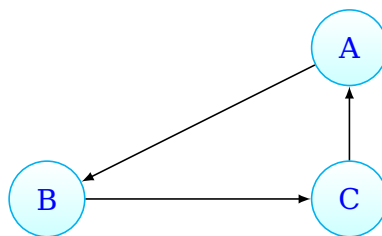


Figura 2.2: Exemplo de um grafo direcionado

Um grafo conexo é aquele em que quaisquer dois vértices são conectados por algum caminho, enquanto um grafo desconexo é aquele que contém algum par de nós que não pode ser conectado por nenhum caminho. As figuras 2.4 e 2.5 exemplificam esse conceito.

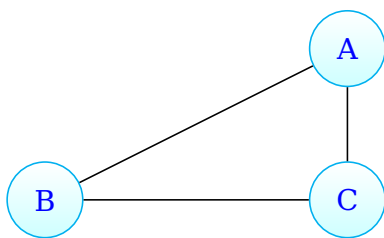


Figura 2.3: Exemplo de um grafo não-direcionado

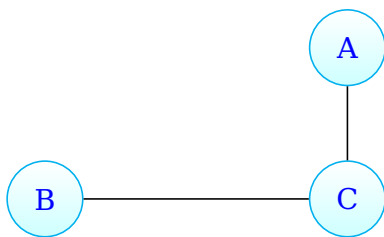


Figura 2.4: Exemplo de um grafo conexo

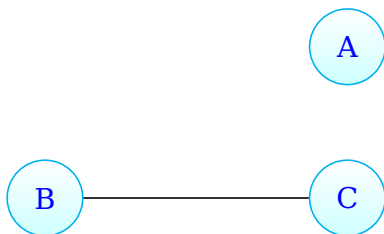


Figura 2.5: Exemplo de um grafo não conexo

Os grafos possuem grande poder representativo e, por esse motivo, são utilizados para modelar diversos cenários encontrados pelo mundo, desde as ruas de uma cidade até as redes de relacionamentos entre pessoas. Além disso, uma vasta quantidade de algoritmos foi criada para extrair métricas e percorrer grafos de diferentes maneiras. Um exemplo são os algoritmos do caminho mínimo, que visam descobrir o menor caminho entre dois nós do grafo. Entre eles, destacam-se o BFS (*Breadth-First Search*), o DFS (*Depth First Search*) (Tarjan, 1971) e o Algoritmo de Dijkstra (Dijkstra, 1959). Além disso, descobrir se um grafo é conexo ou não também pode ser útil para entender sua estrutura, por exemplo.

Nesse contexto, os bancos de dados orientados a grafos foram criados para expor essa estrutura a consultas, alterações, inserções e remoções, da mesma forma que outros tipos de BD. Em geral, os SGBDs orientados a grafos também permitem que os nós e os relacionamentos possam conter propriedades, aumentando ainda mais o potencial semântico da estrutura. O SGBD orientado a grafos mais popular na atualidade é o Neo4J¹, que utiliza a CYPHER como linguagem de consulta. A Figura 2.6 ilustra um grafo criado no Neo4J.

¹ <https://neo4j.com/>

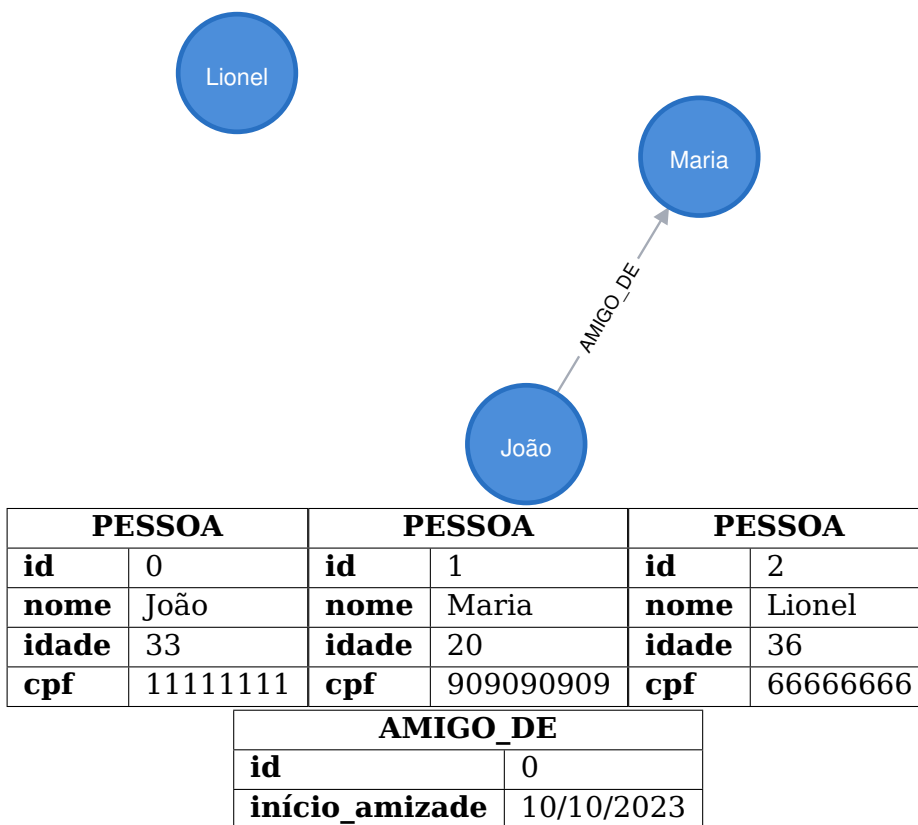


Figura 2.6: Ilustração do grafo criado no Neo4J juntamente com a representação de seus atributos

Capítulo 3

Banco de Dados Operacional

Para permitir o cálculo correto da pontuação de cada jogador, diversas informações externas ao jogador precisam ser levadas em conta durante a aplicação da fórmula, como dados sobre os times, campeonatos, países e continentes em que ele atuou. Portanto, fez-se necessária a implementação de um banco de dados que pudesse armazenar todas as informações relevantes, tanto para a análise dos resultados, quanto para o cálculo da pontuação propriamente dita. É importante esclarecer que boa parte dessas informações (jogadores e times) são advindas da internet, mais especificamente da plataforma *Transfermarkt*¹. Por isso, alguns elementos descritos nesta seção fazem referências ao site.

Nesse contexto, uma das primeiras decisões de projeto tomada foi a escolha do tipo de BD mais apropriado para o teor dos dados a serem armazenados.

3.1 MongoDB – orientação a documentos

O SGBD escolhido para armazenar os dados dos jogadores, times e campeonatos foi o MongoDB, que é um sistema NoSQL que implementa o modelo de dados orientado a documentos. A principal motivação para essa escolha foi a característica da estrutura dos dados advindos do *web scraping*, descrita futuramente no Capítulo 4. Esses dados compõem a maior parte dos dados armazenados no BD. Em suma, a estrutura possui muitos objetos aninhados, que contêm diversas sequências hierárquicas de valores, como é o caso das performances individuais dos jogadores, que são divididas por temporada, por campeonato e por clube, ou seja, três aninhamentos por jogador. Por esse motivo, uma maneira conveniente de representar esses dados é utilizando a estrutura dos documentos do MongoDB.

Além disso, esse SGBD possui ótima integração com a linguagem *Python* por meio da biblioteca *pymongo*, que permite realizar todas as operações necessárias para interação com o BD, além de se aproveitar da própria estrutura de

¹ <https://www.transfermarkt.com/>

dicionários fornecida pela linguagem para representar os documentos.

3.2 Esquema do banco de dados

O BD para esse sistema é separado em seis coleções: **PLAYER** (Jogador), **TEAM** (Time), **CHAMPIONSHIP** (Campeonato), **COUNTRY** (País), **CONTINENT** (Continente) e **SCORE** (Pontuação). As cinco primeiras são utilizadas na fórmula e a sexta é onde os seus resultados são armazenados. A seguir, as figuras 3.1, 3.2, 3.3 e 3.4 expõem os diagramas de cada coleção do BD.

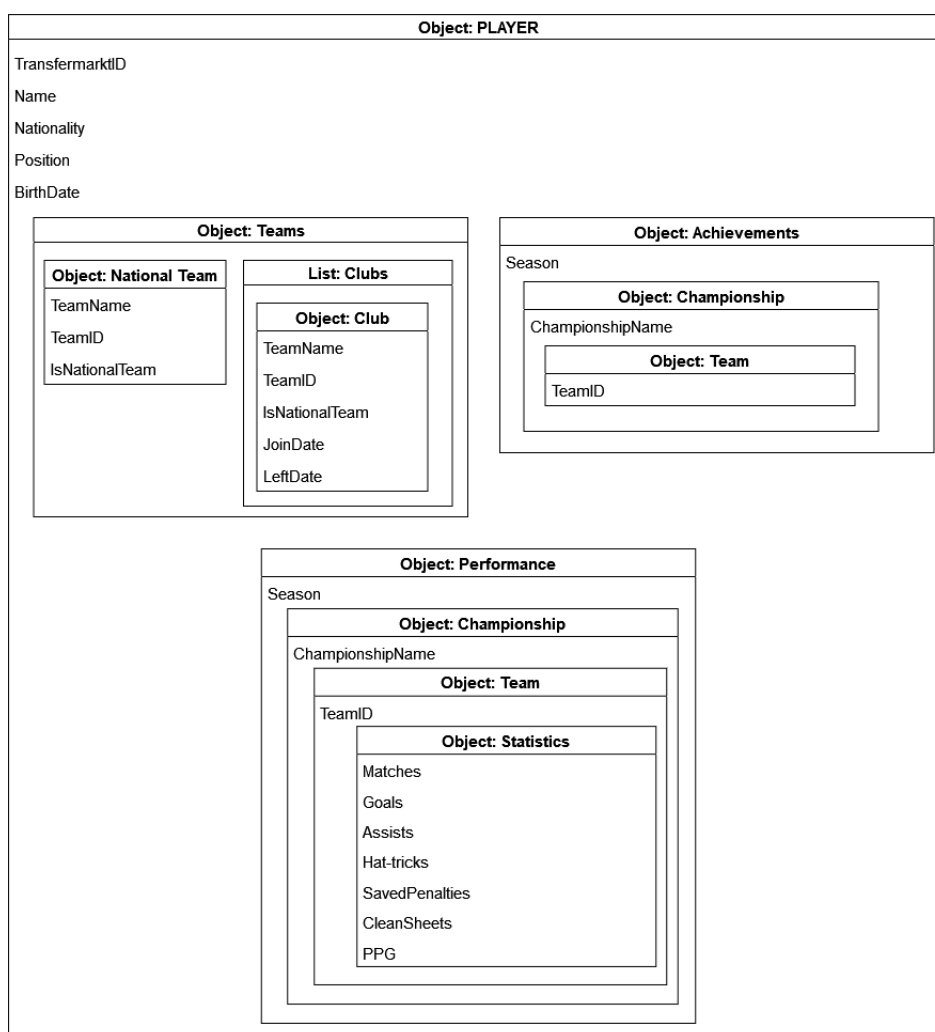


Figura 3.1: Diagrama da coleção **PLAYER**

3.2 | ESQUEMA DO BANCO DE DADOS

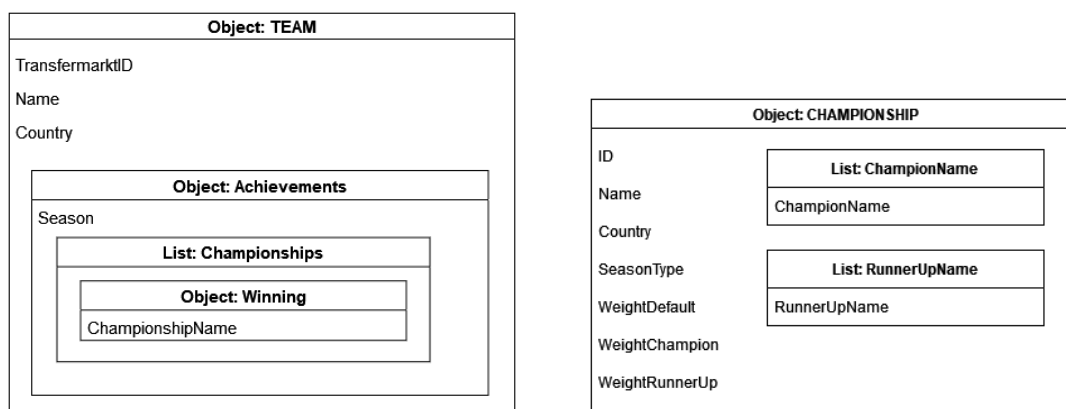


Figura 3.2: Diagrama das coleções **TEAM** e **CHAMPIONSHIP**

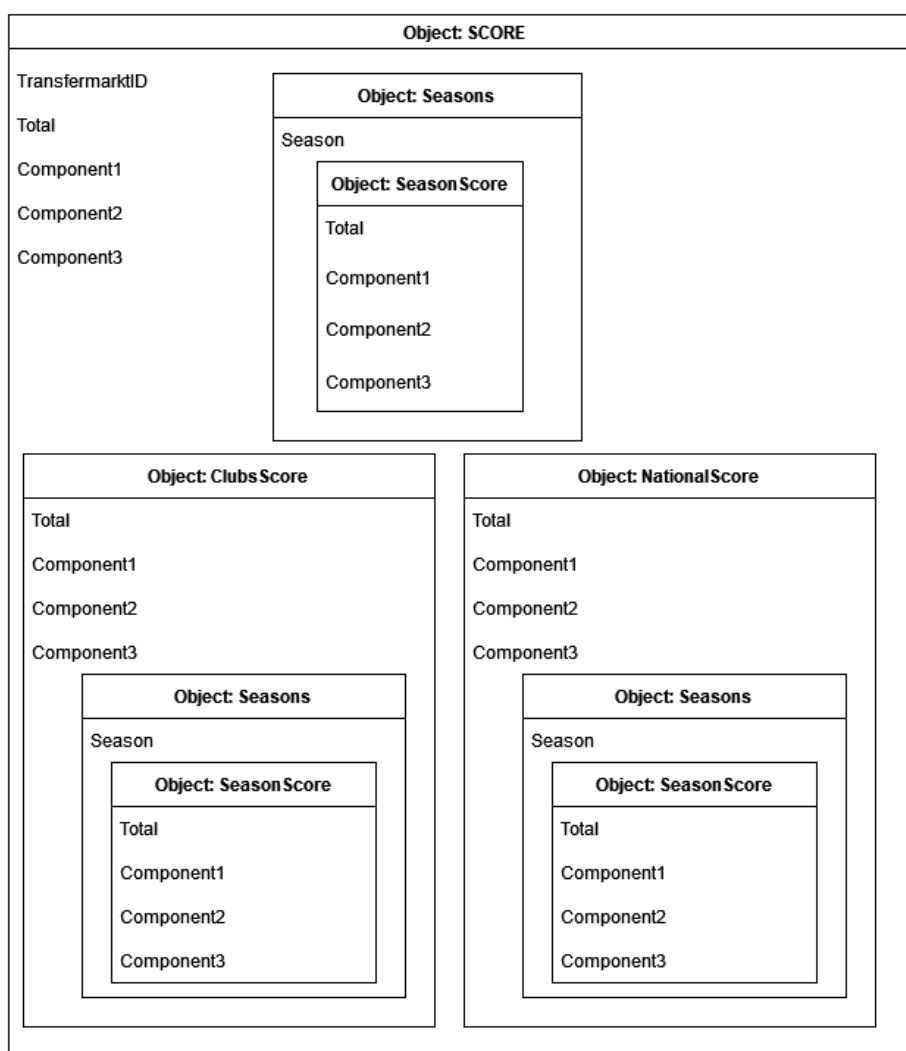


Figura 3.3: Diagrama da coleção **SCORE**

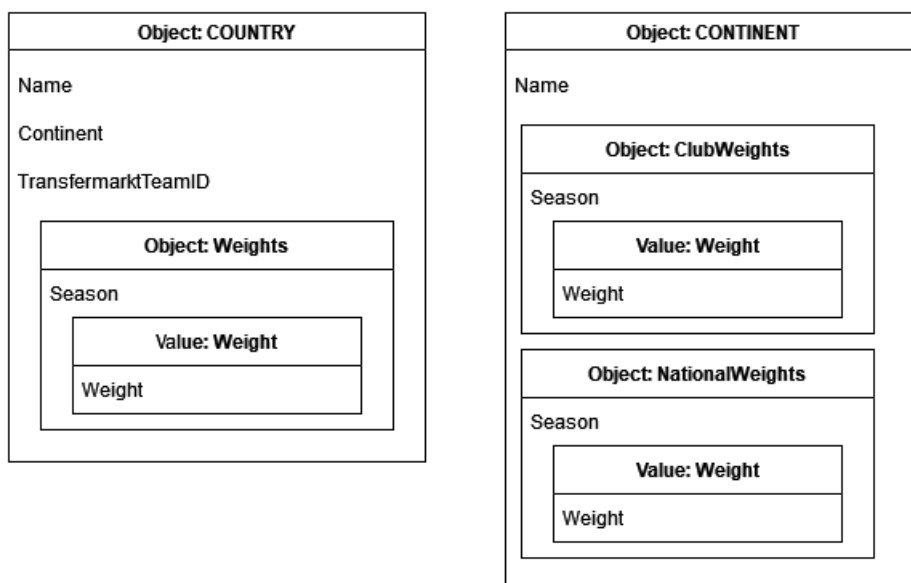


Figura 3.4: Diagrama das coleções **COUNTRY** e **CONTINENT**

Na coleção **PLAYER**, o identificador principal é o atributo *TransfermarktID*, enquanto os outros atributos apresentam as informações básicas de cada jogador. O objeto *Teams* armazena as informações dos times e seleção pelos quais o atleta passou, incluindo as datas de início e fim de cada passagem. Enquanto isso, o objeto *Achievements* descreve os títulos conquistados pelo futebolista e o objeto *Performance* trata de todos os campeonatos disputados por ele, incluindo a quantidade de partidas, gols, assistências, etc. atingida em cada torneio.

Para a coleção **TEAM**, o identificador principal também é chamado de *TransfermarktID*. Além disso, o atributo *Country* indica o país do clube e o objeto *Achievements* armazena todos os títulos do time. Já na coleção **CHAMPIONSHIP**, o identificador principal é o atributo *ID*, *Country* indica o país do campeonato e *WeightDefault*, *WeightChampion* e *WeightRunnerUp* indicam os diferentes pesos para cada competição.

Ademais, a coleção **SCORE** é responsável por armazenar as pontuações de cada jogador, logo seu identificador principal também é o atributo *TransfermarktID*. Além do valor total do *score*, essa coleção também guarda a pontuação parcial de cada componente (*Component1*, *Component2* e *Component3*) utilizado na fórmula (descrita em detalhes no Capítulo 5). As pontuações parciais também são armazenadas para cada temporada, por meio do objeto *Seasons*; para cada clube que o jogador passou, pelo objeto *ClubsScore* e também para a seleção nacional, com o objeto *NationalScore*. Sendo assim, é possível fragmentar o cálculo e destrinchar as pontuações por diferentes pontos de vista, possibilitando algumas das análises apresentadas no Capítulo 8.

Por último, as coleções **COUNTRY** e **CONTINENT** armazenam informações

de cada país e continente, respectivamente. Para **COUNTRY**, são armazenados o atributo identificador (*Name*), o atributo *Continent*, com o continente de cada país e o objeto *Weights*, que contém o coeficiente de peso do país para cada temporada. Na coleção **CONTINENT**, *Name* é o identificador do continente e, além disso, são armazenados os pesos de cada temporada para clubes, por meio do objeto *ClubWeights* e para a seleção nacional, com o objeto *NationalWeights*.

Uma necessidade que surge nas consultas, derivada dessa modelagem dos dados, é a do cruzamento entre coleções. Os atributos mais importantes durante esses cruzamentos são: IDs do Transfermarkt para jogadores, times e campeonatos; para os campeonatos, também é necessário encontrar os nomes dos títulos (*ChampionName*), dos vice-campeonatos (*RunnerUpName*) e o valor dos pesos (*WeightDefault*, *WeightChampion* e *WeightRunnerUp*); além disso, como citado anteriormente, cada campeonato também referencia um país e cada país referencia o seu continente.

Desse modo, a grande quantidade de consultas inter-coleções poderia ser considerada como um “ponto fraco” da modelagem, à medida que dentro da coleção do jogador, por exemplo, não estão armazenados nenhum dos coeficientes utilizados pela fórmula. Contudo, como a quantidade de países, continentes e campeonatos não é grande (da ordem de 10^2) e índices podem ser criados para otimizar as consultas em determinados atributos, o desempenho do sistema não é afetado.

Como introduzido no parágrafo anterior, a criação de índices em alguns atributos é necessária para aumentar a performance das consultas. Por esse motivo, as coleções **PLAYER**, **TEAM** e **SCORE** contêm um índice no atributo *TransfermarktID*. Já **CONTINENT** e **COUNTRY** contêm um índice no atributo *Name*, e **CHAMPIONSHIP** contêm três índices nos atributos *ID*, *ChampionNameSanitized* e em *RunnerupNameSanitized*².

Em geral, o fluxo da informação começa no jogador, que participou ou ganhou algum campeonato com determinado peso. Esse campeonato, por sua vez, é ligado a um país e o país é ligado a um continente. Dessa maneira, a informação para uma performance ou título se torna completa. Para os times, o fluxo é semelhante, mas só funciona para títulos, que é a única informação utilizada pela fórmula e, portanto, armazenada pelo BD.

² *ChampionNameSanitized* e *RunnerupNameSanitized* são derivados de *ChampionName* e *RunnerupName*, respectivamente, e tratam de apenas remover caracteres especiais, espaços e hífen dos nomes dos campeonatos.

Capítulo 4

Extração de Dados

Para que seja possível obter informações precisas, atualizadas e completas com relação aos jogadores e times de todo o planeta, é praticamente obrigatório utilizar ferramentas e plataformas de terceiros, que reúnem uma imensa quantidade de dados de maneira sistemática e de fácil visualização. Esses serviços podem ter diferentes alvos e objetivos, desde auxiliar olheiros e dirigentes de clubes a selecionarem a próxima jovem promessa ou apenas facilitar a vida do torcedor que deseja acompanhar o desempenho do seu time do coração. Alguns exemplos desses serviços são os seguintes: *Transfermarkt*, *Ogol*, *Futebol80*, *Fussballtransfers*, *Goal*¹. Este trabalho, portanto, utiliza uma dessas plataformas a fim de permitir a extração das informações necessárias para a correta aplicação da fórmula.

Os critérios levados em conta para a realização da escolha da plataforma foram os seguintes: i) quantidade e qualidade da informação com relação aos principais torneios de futebol, ii) quantidade e qualidade da informação com relação aos torneios em países menos relevantes para o futebol mundial, iii) quantidade e qualidade da informação com relação às épocas mais antigas, iv) facilidade na obtenção dos dados do ponto de vista técnico e, por fim, v) autorização para obtenção e utilização dos dados.

Com base nesses critérios, o site escolhido foi o *Transfermarkt*, que para i), iv) e v) é definitivamente o melhor entre as opções, já que é muito completo para as ligas europeias e de seleções a partir do século XXI, é gratuito, não necessita de autenticação para visualizar todas as informações e, o mais importante, autorizou o uso dos dados para este trabalho. Apesar disso, para as ligas de países menores, longe do eurocentrismo, principalmente tratando-se do século XX, o site apresenta lacunas na completude e precisão das informações. Por esse motivo, a extração das informações para este sistema é proveniente de duas fontes: A) o site *Transfermarkt* e B) planilhas que contêm dados específicos que não estão presentes na plataforma.

¹ <https://www.transfermarkt.com/>, <https://www.ogol.com.br/>, <https://www.futebol80.com.br/>, <https://www.fussballtransfers.com/> e <https://www.goal.com/br>

Outro fato importante de ser mencionado, é que as inserções automatizadas são realizadas para jogadores e times, apenas. Campeonatos, países e continentes precisam ser inseridos manualmente, já que as informações que eles armazenam não são passíveis de web scraping, por se tratarem de coeficientes completamente customizados para a aplicação da fórmula. Mesmo assim, na prática, o repositório [L. B. Salvador, 2023] disponibiliza um script de inicialização que já insere no MongoDB os campeonatos, países e continentes automaticamente, contendo todos os pesos utilizados pela fórmula.

4.1 Fonte A: *Transfermarkt*

Apesar de exibir todas as informações sem custos e necessidade de *login*, o site *Transfermarkt* não oferece nenhum tipo de API (*Application Programming Interface*) para obter seus dados. Por esse motivo, para realizar a extração, foi necessário realizar um *web scraping* diretamente nas páginas HTML (*HyperText Markup Language*) da plataforma. O *script* que realiza essa tarefa foi implementado em *Python* e utilizou duas bibliotecas principais: *Requests*² e *BeautifulSoup*³. A primeira biblioteca é responsável por realizar as requisições HTTP para todas as URLs do site que contêm informações úteis para a fórmula. A segunda é caracterizada como um *parser* HTML ou XML (neste caso apenas HTML) que recebe uma *string* contendo todo o código fonte da página e devolve uma estrutura na qual é possível procurar elementos por suas *tags*, identificadores, classes, atributos, etc., de modo a possibilitar o *web scraping* propriamente dito. A motivação para a escolha dessas bibliotecas, em detrimento de outras com o mesmo propósito, foi a familiaridade e experiência do autor com esse material, além do fato de ambas serem muito bem documentadas.

A seguir, são expostos dois diagramas contendo as URLs necessárias para a obtenção de cada um dos dados úteis para a aplicação correta da fórmula. O da Figura 4.1 se refere às informações dos times e o da Figura 4.2 se refere às informações dos jogadores.

Como é possível ver nos diagramas, os jogadores são responsáveis pela maior parte das URLs, o que faz sentido já que a pontuação é justamente focada neles. De qualquer forma, para times, o importante é extrair informações básicas (nome e país de origem) e os títulos conquistados. Para jogadores, além do básico (nome, nacionalidade e data de nascimento), é importante obter a sua posição, já que a fórmula para goleiros e defensores é diferente da fórmula para o resto dos jogadores. Além disso, é necessário coletar uma lista de times em que ele atuou, incluindo a seleção nacional (sem incluir a seleção olímpica ou categorias de base). Por fim, o foco da extração é obter informações sobre os títulos conquistados e sobre a performance em cada campeonato disputado, que é caracterizada pela quantidade de gols, assistências, *hat-tricks*, pênaltis

² <https://requests.readthedocs.io/en/latest/>

³ <https://www.crummy.com/software/BeautifulSoup/>

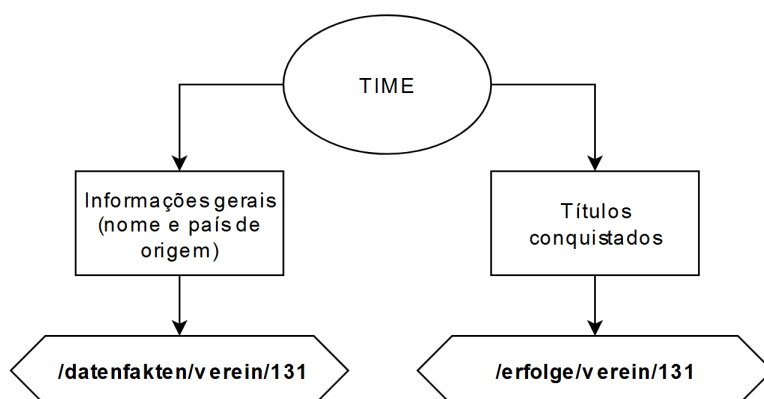


Figura 4.1: Diagrama das URLs necessárias para a extração dos times

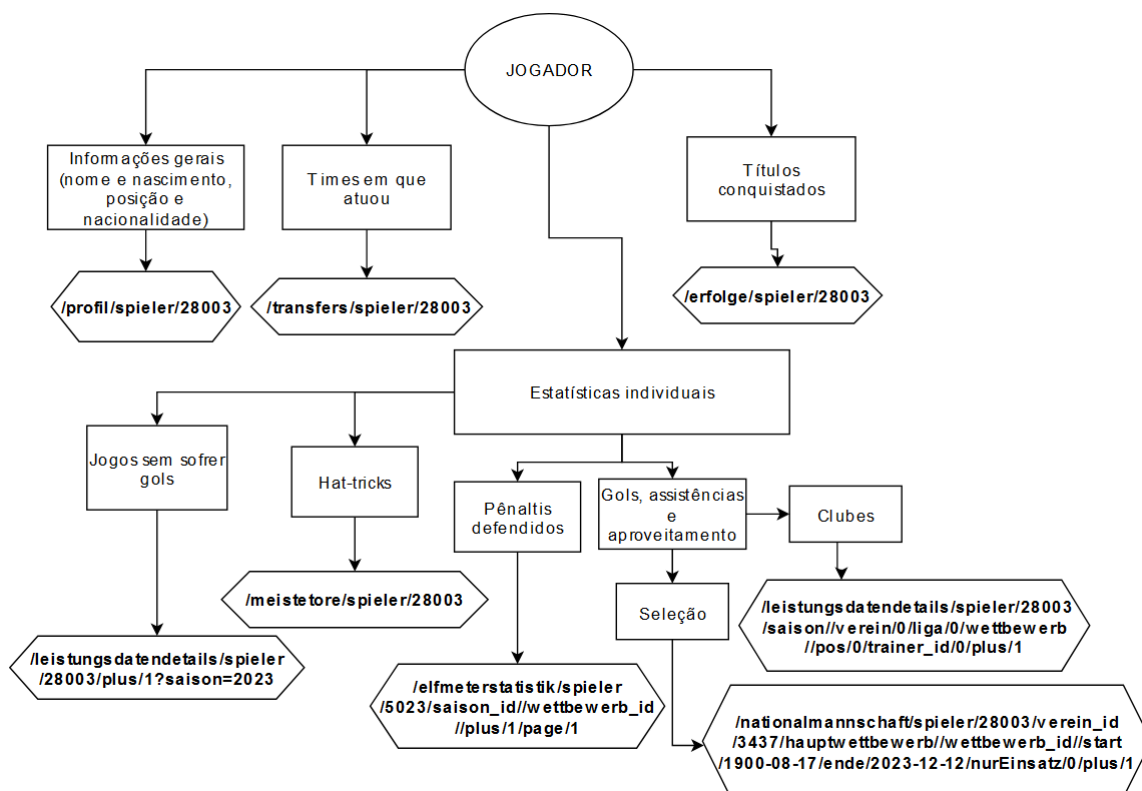


Figura 4.2: Diagrama das URLs necessárias para a extração dos jogadores

defendidos (apenas goleiros), jogos sem sofrer gols (goleiros e defensores) e aproveitamento na competição.

Por conta da estrutura da plataforma, as informações sobre a performance na seleção estão em URLs diferentes da performance dos clubes. Além disso, os dados aparecem de diferentes maneiras ao longo das URLs, como em tabelas com resumos por campeonato, no caso da performance de clubes, ou também listas jogo a jogo, no caso da performance de seleções. De qualquer forma, com as dez URLs mostradas nos diagramas é possível obter tudo que é necessário para a aplicação total da fórmula.

4.1.1 Observações acerca da qualidade e corretude da extração

Uma das dificuldades quando se trata de realizar *web scraping*, em comparação com o uso de uma API existente, é a impossibilidade de se ter certeza de que a estrutura da página não será alterada, pois caso isso aconteça, é possível que o comportamento do extrator seja total ou parcialmente prejudicado. Por esse motivo, para a realização do *web scraping* em questão, durante cada extração, alguns testes em tempo de execução são realizados para checar se a estrutura da página é a esperada. Mesmo assim, essa checagem é limitada já que, por exemplo, não é possível ter certeza de que uma resposta do extrator que indica que certo jogador não possui gols estaria errada, visto que é possível que ele realmente não possua nenhum gol ao longo de sua carreira. Deste modo, além das checagens de estrutura, o *scraper* pode ser habilitado para fornecer avisos sempre que algum conteúdo seja peculiar (como é o caso do jogador que não possui títulos ou *hat-tricks*), e então cabe ao usuário se certificar de que o comportamento reportado faz sentido ou então de que realmente há alguma anomalia com relação à informação.

Dessa maneira, a implementação desse *web scraper* requer manutenção sempre que a estrutura da página é alterada, porém, neste caso, o ideal seria que um site do porte do *Transfermarkt* oferecesse um serviço de API ou então que não necessitasse de constantes alterações em sua estrutura.

4.1.2 Interação entre o BD e o extrator

Uma importante observação a ser feita com relação a interação do BD com o módulo de extração do sistema é a seguinte: em geral, o extrator obtém todas as informações de cada jogador e time sem precisar consultar o banco de dados, de modo que, mesmo que um campeonato não tenha sido registrado no BD mas esteja sendo referenciado na fonte do jogador, ele possa ser inserido no documento deste jogador. Isso impede que seja necessário refazer as extrações para todos os jogadores e times quando um novo campeonato é inserido no banco de dados e também garante maior independência entre os módulos do sistema, facilitando a depuração e diminuindo a chance de erros.

Apesar disso, um obstáculo encontrado para que essa prática seja totalmente

aplicada é causado pela discrepância, natural do mundo futebolístico, entre a definição do período de uma temporada em cada um dos países do mundo (já detalhada no Capítulo 2). Assim, suponha dois jogos ocorrendo na mesma data em campeonatos distintos. Para garantir a precisão da fórmula, é preciso saber em qual temporada cada jogo deve estar contido, mas ainda que as duas partidas ocorram na mesma data, a temporada pode variar dependendo do campeonato atrelado a cada um dos jogos. Uma exemplificação desse fenômeno é um jogo de *Champions League* (Europa) no dia 19/05/2023 e outro de Libertadores (América do Sul) no mesmo dia. Enquanto para a Europa a partida pertence à temporada 2022, para a América do Sul o jogo faz parte da temporada 2023.

Deste modo, para a informação ser completamente fidedigna, é necessário saber, para cada campeonato, qual modelo de temporada ele adota. Sendo assim, sempre que um novo torneio é inserido, é preciso indicar em qual dos dois tipos de temporada (Norte ou Sul) ele é baseado. Para isso, o atributo *SeasonType* da coleção **CHAMPIONSHIP** foi idealizado. Portanto, durante a extração, algumas informações só são inseridas corretamente após a consulta desse atributo, fato esse que impede a total independência entre o banco de dados e o extrator.

4.2 Fonte B: extração manual

As planilhas têm como principal objetivo corrigir e complementar informações advindas da fonte A. Por meio desses registros, os dados de times e jogadores podem ser corrigidos, sendo que para times é possível adicionar e/ou remover títulos e para jogadores é possível adicionar e/ou remover performances em campeonatos e títulos.

Com relação às performances do jogador, há duas maneiras de modificá-las: a) jogo a jogo, isto é, uma linha por jogo, onde em cada linha há a indicação da quantidade de gols, assistências, pênaltis defendidos e resultado final da partida (de onde é possível extrair os *clean sheets* e pontos por jogo), ou b) campeonato a campeonato, onde cada linha apresenta o resumo das estatísticas para cada campeonato, da mesma forma que o próprio *Transfermarkt* já faz. A Tabela 4.1 ilustra uma planilha no formato a) e a Tabela 4.2 ilustra uma planilha no formato b).

date	champId	clubId	scored goals	conceded goals	goals	assists	saved penalties
20/05/1990	BRA1	199	3	1	1	0	0
12/09/1995	RIOSP	199	4	4	3	1	0

Tabela 4.1: Exemplo de uma planilha contendo a performance jogo-a-jogo de um jogador

A motivação para essas especificidades é o fato da fonte desses dados não ser homogênea e variar muito, então é possível que, para certo jogador, seja necessário obter as informações faltantes diretamente dos jornais da época, por exemplo no caso do Pelé, sendo, portanto, mais natural que se utilize o modo a).

champId	season	clubId	matches	goals	assists	hat-tricks	clean sheets	saved penalties	PPG
CL	2022	131	2	0	1	0	0	0	3
WM22	2022	3437	7	7	3	0	0	0	2.75

Tabela 4.2: Exemplo de uma planilha contendo a performance por campeonato de um jogador

Por outro lado, se a informação é advinda de outra plataforma já sistematizada, muitas vezes ela já está no modo b), por isso é necessário que se tenha suporte para mais de um modo de inserção.

Assim, o fluxo a cada extração de dados de jogadores e times é o seguinte: cria-se a estrutura de extração advinda do *Transfermarkt*, checa-se se há planilhas de inserção para esse jogador ou time e, em caso afirmativo, são feitas as atualizações necessárias. Deste modo, para o resultado final, não há diferenciação entre o que foi provido pelo *Transfermarkt* e o que foi alterado manualmente.

Por fim, é importante notar que dados como nome, data de nascimento, país de origem, posição (jogador) e clubes em que atuou (jogador) não são informações alteráveis pelas planilhas. Além disso, se um jogador não estiver presente na base de dados do *Transfermarkt* não é possível inseri-lo manualmente apenas por meio das planilhas.

Capítulo 5

O Cálculo da Pontuação

Como já apresentado anteriormente, a pontuação é baseada em uma fórmula (E. P. Salvador *et al.*, 2022), que permite metrificar a qualidade da carreira de um jogador no que diz respeito à performance individual, títulos conquistados e impacto causado nos times em que atuou. É importante notar que o cálculo não permite avaliar o potencial futuro de um jogador, nem leva em conta valores de mercado envolvidos nas transferências ao longo de sua carreira, muito menos a sua capacidade de influência fora das quatro linhas. A ideia é lidar apenas com os atributos inerentes ao jogo de futebol.

Sendo assim, a estrutura da fórmula é composta por três grandes componentes, que fornecem pontuações independentes, que posteriormente são somadas para gerar o valor final da pontuação. Os componentes são: individual, conquistas e impacto. Ao longo de cada etapa, além dos dados advindos dos próprios jogadores e times, é necessário levar em conta diversos coeficientes que atrelam o campeonato, país, continente e alguns outros fatores ao evento tratado, seja ele a performance individual, conquistas ou impacto do jogador. A próxima seção é dedicada a explicar o funcionamento desses coeficientes e, em seguida, cada seção buscará descrever um dos componentes.

5.1 Coeficientes de ajuste

Cada coeficiente varia no intervalo $[0, 1]$ e seu objetivo é ajustar cada elemento a ser quantificado na pontuação, considerando a dificuldade ou importância proveniente dos contextos dos campeonatos, países ou continentes. Por exemplo, um torneio como a Copa do Mundo deve ser mais importante na fórmula do que o campeonato brasileiro. Não apenas por ser mais difícil do ponto de vista técnico, mas também pela importância e frequência que o torneio ocorre. Assim, o peso de cada um deles deve ser diferente. Outro exemplo é que o título do campeonato húngaro, por exemplo, deve ser menos recompensado do que a conquista do campeonato espanhol. Ou então que ganhar o Campeonato Paulista de 2023 deve ser diferente de ganhar o mesmo torneio de 1966.

Em resumo, são diversos fatores envolvidos no futebol passíveis de serem ajustados de acordo com o contexto. Para países, continentes e alguns campeonatos excepcionalmente, os pesos são únicos por temporada, enquanto para a maioria dos campeonatos o peso é fixo em toda a história, apenas diferenciando os pesos entre participante, campeão e vice-campeão. Para entender como cada peso foi estipulado, seria necessário outra monografia, então o importante para este trabalho é saber que eles existem e que são facilmente alteráveis, caso qualquer um deseje modificá-los.

As tabelas 5.1, 5.2 e 5.3 mostram alguns exemplos de diferentes coeficientes.

Campeonato	Peso	Peso título	Peso vice
Campeonato Brasileiro	2,0	20,0	N/A
Copa do Mundo	19,02	190,2	47,5
Copa Libertadores	4,75	47,5	N/A
Copa do Rei	1,5	15,0	N/A

Tabela 5.1: Exemplo da estrutura dos coeficientes dos campeonatos

País	Temporada	Peso
Brasil	1929	1,0
Brasil	1986	0,72
Brasil	2021	1,0
Inglaterra	1998	0,53
Inglaterra	2006	0,89
Inglaterra	2014	0,8

Tabela 5.2: Exemplo da estrutura dos coeficientes dos países

Continente	Temporada	Peso Clube	Peso Seleção
Europa	1973	1,0	1,0
Am. do Sul	2003	0,82	1,0
Am. do Sul	2012	0,67	1,0
Oceania	2016	0,39	0,4

Tabela 5.3: Exemplo da estrutura dos coeficientes dos continentes

5.2 Componente individual

Esse componente tem como principal objetivo avaliar as estatísticas individuais do jogador. Para isso, os seguintes atributos são levados em conta: partidas, gols, assistências, *hat-tricks*, aproveitamento, jogos sem sofrer gols (defensores e goleiros, apenas) e também pênaltis defendidos (goleiros, apenas). Os valores para cada atributo são os seguintes:

- Partida = 1
- Gols = 2
- Assistências = 1,5
- *Hat-tricks* = 3
- Jogos sem sofrer gols = 1,3 (goleiro) e 0,65 (defensor)
- Pênaltis defendidos = 2,5
- Aproveitamento = $(Vitorias * 3 + Empates) / Partidas$

Ao iterar pelo objeto contendo a performance de um jogador, para cada instância de um campeonato essa pontuação é calculada e, após todos os ajustes (pesos do campeonato, país e continente), o valor final é encontrado e então somado a pontuação total desse componente.

Exemplo de aplicação desse componente: Primeiramente, é necessário obter no registro do jogador a performance em determinado campeonato. No caso do registro do Programa 5.1, “ES1” representa o Campeonato Espanhol, 131 representa o clube Barcelona e o jogador em questão é Lionel Messi.

```

{"2012": {
  "ES1": {
    "131": {
      "Partidas": 32,
      "Gols": 46,
      "Assistencias": 13,
      "Hat-tricks": 2,
      "Aproveitamento": 0.85,
      "Jogos sem sofrer gols": 0,
      "Penaltis defendidos": 0
    }
  }
}}

```

Programa 5.1: Exemplo de fragmento da performance de um jogador

$$Resultado = (1 * 32 + 2 * 46 + 1,5 * 13 + 3 * 2) = 149,5$$

Então, esse resultado deve ser multiplicado pelo aproveitamento de pontos e pelos coeficientes de ajuste correspondentes ao campeonato, país e continente:

$$C_{Individual} = 149,5 * 0,85 * Peso_{ES1} * Peso_{Espanha(2012)} * Peso_{Europa(Clubes-2012)}$$

$$C_{Individual} = 127,07 * 2,0 * 1,0 * 1,0 = 254,15$$

5.3 Componente de conquistas

Esse componente tem como principal objetivo avaliar os títulos conquistados por cada jogador. Percorrendo cada título presente no registro do atleta, a pontuação é encontrada multiplicando o peso do campeonato conquistado pelo restante dos ajustes. Além disso, para diferenciar jogadores que jogaram 100% do campeonato de jogadores que atuaram apenas 5%, por exemplo, é feita outra multiplicação considerando o número de partidas jogadas contra o número de partidas esperadas para aquele torneio.

Exemplo de aplicação desse componente: Primeiramente, é necessário obter do registro do jogador os títulos conquistados (Programa 5.2).

```

{"2012": {
  "Campeao do Campeonato Espanhol":{
    "Time":131
  }
}}
```

Programa 5.2: Exemplo de fragmento dos títulos de um jogador

Então, deve-se multiplicar os ajustes correspondentes ao campeonato, país, continente e jogos disputados, que nesse caso vale 1,0 pois o jogador esteve presente em 84% (32/38) das partidas do torneio.

$$C_{\text{Títulos}} = \text{Peso}_{\text{TítuloES1}} * \text{Peso}_{\text{Espanha(2012)}} * \text{Peso}_{\text{Europa(Clubes-2012)}} * \text{AjusteJogos}$$

$$C_{\text{Títulos}} = 20 * 1,0 * 1,0 * 1,0 = 20$$

5.4 Componente de impacto

Esse componente tem como principal objetivo avaliar o impacto causado pelo jogador em cada clube/seleção que atuou, sendo o componente mais complexo de ser calculado, já que precisa levar em conta não só o que o jogador conquistou no período em que atuou mas também o que o time conquistou em toda a história.

A pontuação final do componentes leva em conta dois sub-componentes: impacto agudo e impacto de longo prazo. Cada porção do cálculo é feita por temporada e por time, e o resultado é somado no final para construir o impacto geral. As próximas explicações supõem uma temporada e time fixos.

5.4.1 Impacto agudo

O impacto agudo é a diferença entre a pontuação de títulos do jogador na temporada em questão (*ScoreJogador*) com a pontuação de títulos do time na

temporada correspondente (*ScoreTime*, definida a seguir). Essa subtração é então multiplicada por 10, como na fórmula:

$$Impacto_{agudo} = (ScoreJogador - ScoreTime) * 10$$

A temporada correspondente é encontrada da seguinte maneira: levando em conta a passagem completa de um jogador por um time, com começo C e fim F, primeiro é contabilizada a duração D da passagem ($D = F - C + 1$). Então ordena-se de forma decrescente as pontuações de títulos do jogador pelo time em todas as temporadas (serão D temporadas: C, C + 1, C + 2, ..., F - 2, F - 1, F). Com isso, encontra-se o índice I ($0 \leq I < D$) da temporada em questão, em comparação com a passagem inteira do jogador pelo time, ou seja, $I = 0$ corresponde à temporada com mais pontuação, $I = 1$ a segunda, até $I = D - 1$ que é a de pior pontuação. Depois, considerando a mesma duração D, só que começando na temporada C-D e terminando em C - 1, essas temporadas são ordenadas pela pontuação de títulos do time em cada uma. Então, a temporada de índice I é escolhida e considerada a temporada correspondente, cuja pontuação corresponde a *ScoreTime* na fórmula anterior.

Exemplo para facilitar a compreensão do cálculo, utilizando jogador e clube fictícios: Primeiramente, é necessário obter os dados de conquistas durante a passagem do jogador pelo clube, nesse caso nas temporadas 2006 até 2010 ($C = 2006$, $F = 2010$ e $D = 5$) e calcular as pontuações correspondentes a cada temporada (como mostra a Tabela 5.4).

Jogador	Temporada	Pontuação títulos
J1	2006	60
J1	2007	32
J1	2008	56
J1	2009	0
J1	2010	10

Tabela 5.4: Pontuação de títulos do jogador J1 pelo clube C1 de 2006 até 2010

Então, é feito o mesmo levantamento para o clube, porém durante as temporadas (C-1, C-2, ..., C-D), nesse caso de 2001 até 2005 (Tabela 5.5).

Clube	Temporada	Pontuação títulos
C1	2005	100
C1	2004	23
C1	2003	0
C1	2002	45
C1	2001	0

Tabela 5.5: Pontuação de títulos do clube C1 de 2001 até 2005

Seja 2008 a temporada de interesse para o cálculo do impacto agudo. Essa

temporada corresponde à segunda maior pontuação durante a passagem do jogador pelo clube ($I = 1$). De 2001 até 2005, a segunda melhor temporada do clube foi 2002, logo ela é a temporada correspondente. Sendo assim, a aplicação da fórmula é:

$$Impacto_{agudo} = (56 - 45) * 10 = 11 * 10 = 110$$

5.4.2 Impacto de longo prazo

Um pouco mais simples que o anterior, esse impacto busca ilustrar a importância de cada temporada de um jogador em certo time com relação a todos os títulos do time na história (até a saída do jogador). A fórmula que representa este impacto é a seguinte:

$$Impacto_{longo} = (ScoreJogador / ScoreTime) * 1000$$

A ideia é calcular a pontuação de títulos do jogador por determinado time para a temporada em questão ($ScoreJogador$) e dividi-la pela pontuação de títulos do time desde o início de sua história até a última temporada do jogador atuando pelo time ($ScoreTime$). Depois, o resultado é multiplicado por 1000. Caso $ScoreTime$ seja igual a 0, $ScoreJogador$ também deveria ser 0, e esse cálculo é zerado.

Utilizando as mesmas informações expostas durante a explicação do impacto agudo (tabelas 5.4 e 5.5), considerando a temporada 2008 como alvo e supondo que o clube C1 nasceu em 2001, falta apenas obter os dados de pontuação do clube de 2006 até 2010 (que em geral será igual a pontuação do jogador nessas temporadas).

Clube	Temporada	Pontuação títulos
C1	2006	60
C1	2007	32
C1	2008	56
C1	2009	0
C1	2010	10

Tabela 5.6: Pontuação de títulos do clube C1 de 2006 até 2010

Dessa forma, o score total do time é:

$$ScoreTime = \sum_{i=2001}^{2010} ScoreTime_i$$

$$ScoreTime = 0 + 45 + 0 + 23 + 100 + 60 + 32 + 56 + 0 + 10 = 326$$

Aplicando a fórmula:

$$Impacto_{longo} = (56/326) * 1000 = 171,77$$

5.4.3 Impacto final

O impacto final de cada temporada é calculado da seguinte maneira:

$$Impacto_{final} = Aprov * (Impacto_{agudo} + Impacto_{longo})/2$$

Aprov corresponde ao aproveitamento de pontos do jogador em toda a temporada, considerando todos os times que ele atuou e competições que disputou, *Impacto_{agudo}* corresponde a soma de todos os impactos agudos calculados para os times que o jogador atuou naquela temporada e *Impacto_{longo}* corresponde a soma de todos os impactos de longo prazo calculados para os times em que o jogador atuou naquela temporada.

É importante notar que, por conta da diferença existente no impacto agudo, o valor do impacto final pode ser negativo. Desse modo, a fim de evitar valores negativos, caso o impacto final seja menor que 0, ele é zerado. A razão para esta escolha é que o jogador seria “punido” (já que sua pontuação decresceria) por atuar em um time que já foi muito vitorioso mas que atualmente não consegue conquistar os mesmos títulos.

Considerando os exemplos anteriores de impacto agudo e de longo prazo, supondo que o aproveitamento do jogador durante a temporada foi de 80% e que ele atuou apenas no clube C1, o impacto final para a temporada 2008 seria:

$$Impacto_{final} = 0,8 * (110 + 171,77)/2 = 112,70$$

5.5 Pontuação final

Como já introduzido no início da seção, a pontuação final é calculada da seguinte maneira:

$$P_f = C_{individual} + C_{titulos} + C_{impacto}$$

Além disso, como já introduzido no Capítulo 3, no banco de dados há uma coleção específica (**SCORE**) para armazenar, além de seu valor final, todas as porções da fórmula, divididas entre componentes, temporadas e diferenciando clubes das seleções. Dessa maneira, é possível realizar mais análises levando em conta não só o valor final da pontuação, mas diversos fragmentos da fórmula também, permitindo comparações que envolvem isolar um dos componentes, comparar pontuações proporcionalmente a temporadas disputadas e diversas outras observações.

Capítulo 6

Banco de Dados de Grafo

A partir do banco de dados operacional, que contém o registro completo de todos os jogadores e suas respectivas pontuações, é possível criar um grafo que promove uma representação distinta da estrutura de documentos desse BD. A ideia é que cada nó do grafo represente um jogador do registro, e cada aresta (não direcional) represente as atuações conjuntas entre dois jogadores, ou seja, se dois atletas estão conectados quer dizer que em alguma temporada eles atuaram juntos pelo mesmo time.

O grafo possibilita a representação não só das informações individualmente, isto é, cada jogador com sua pontuação, como também da interação entre cada elemento do BD. Dessa maneira, pode-se tratar o conjunto de jogadores coletivamente, de acordo com as relações formadas entre companheiros de equipe ao longo de suas carreiras. Do ponto de vista futebolístico, mais análises podem ser feitas, já que é possível, por exemplo, saber qual jogador coleciona mais ou menos companheiros de equipe diferentes. Além disso, do ponto de vista computacional, a estrutura de um grafo possui métricas próprias que também podem enriquecer o estudo, como qual a maior distância entre dois jogadores, ou qual a taxa de crescimento do grafo conforme mais jogadores são adicionados no BD.

Assim, para melhor compreensão das etapas envolvidas neste processo, as próximas seções buscam elucidar as motivações para escolha do SGBD e o processo de conversão entre os bancos de dados, além de mostrar na prática alguns resultados fornecidos por esse estudo.

6.1 A escolha do SGBD Neo4J

Atualmente, existem diversos SGBDS baseados em grafos disponíveis no mercado, cada um com suas particularidades e objetivos. Dentre eles, destacam-

se: *Amazon Neptune*¹, que faz parte do Amazon Web Services (AWS), *OrientDB*² (que pode ser orientado não só a grafos como também a documentos, chave-valor e objetos) e o *Neo4J*³, que foi o sistema escolhido para este projeto. As principais vantagens do Neo4J são: gratuidade da ferramenta; facilidade de manipulação dos dados, já que utiliza a linguagem de consulta CYPHER; não necessidade de instalação local, já que pode funcionar com uma máquina virtual; interface web intuitiva e poderosa; completa integração com a linguagem Python, que permite que a conversão entre MongoDB e Neo4J ocorra facilmente; implementação nativa de diversos algoritmos que extraem métricas do grafo, como busca por caminhos, detecção de comunidades e detecção de centralidade, e, por fim, a última vantagem do Neo4J é a familiaridade do autor com a ferramenta.

Abaixo segue uma tabela com a comparação entre algumas ferramentas de gerenciamento de dados orientadas a grafos:

	Neo4J	Amazon Neptune	OrientDB
Empresa responsável	Neo4J	Amazon	OrientDB
Licença de software	GPLv3 para a Community Edition	Proprietária	Apache 2 (open source)
Preços	Gratuito para 1 instância de BD	Muito variáveis, dependendo do local de hospedagem. De 0.1 USD até 70 USD por hora.	Gratuito
Objetivo	Fornecer performance, flexibilidade e agilidade, além de implementar transações ACID.	Fornecer escalabilidade e facilidade ao usuário utilizando a estrutura de <i>cloud computing</i> fornecida pelo AWS.	Implementar um SGDB de código aberto que possa ser facilmente manipulado inclusive por quem conhece apenas SQL
Linguagem de consulta	CYPHER (código aberto adaptada do openCypher)	Apache TinkerPop Gremlin, SPARQL e openCypher	SQL (estendido para dar suporte a consultas próprias de grafos)
Integração com linguagens	Python, Javascript, Java, PHP, Ruby, Go, Perl, etc.	As linguagens presentes no AWS (Python, PHP, Java, Ruby, etc.)	Python, C, C++, Java, Javascript, Ruby, etc.
Outras características	Possui outras versões pagas e com licenças comerciais para empresas.	Todo o gerenciamento de hardware e armazenamento é feito automaticamente pelo AWS.	Possui outras versões pagas e com licenças comerciais para empresas.

Tabela 6.1: Comparação entre três dos principais bancos de dados orientados a grafos

6.2 Conversão MongoDB → Neo4J

A conversão entre MongoDB (BD Operacional) e o Neo4J (BD Grafo) ocorre ao mesmo tempo em que a inserção ou atualização de um jogador é feita. Primeiro, os registros são armazenados no SGBD orientado a documentos, para então serem adicionados na estrutura do grafo. Sendo assim, para formar os nós de um jogador, basta obter seu ID, nome e pontuação total. Para as arestas de cada jogador, o algoritmo percorre o registro de todos os outros jogadores armazenados no BD Operacional em busca de atletas que atuaram juntamente do jogador

¹ <https://aws.amazon.com/pt/neptune/>

² <http://orientdb.org/>

³ <https://neo4j.com/>

a ser atualizado/inserido. A ideia é que, a partir de cada tupla (campeonato, temporada, time) contida na performance de um jogador, seja possível encontrar todos os outros atletas que atuaram com ele. É importante destacar que, com esse procedimento, jogadores que jogaram em uma mesma temporada por um mesmo time mas não jogaram nenhum campeonato simultaneamente não são considerados pares, tornando a estrutura mais fidedigna com a realidade, já que nesse caso é impossível que tenham jogado pelo menos uma partida juntos.

Outra observação importante é que, para simplificar a visualização e armazenamento do grafo, algumas decisões foram tomadas a fim de remover arestas redundantes ou em excesso. Primeiramente, cada par de jogadores só possui uma aresta entre si, independente da quantidade de temporadas ou times em que ambos atuaram conjuntamente. Além disso, a seleção nacional não é válida para considerar que dois jogadores jogaram juntos e, por último, campeonatos que não estão registrados no banco de dados operacional (categorias de base, por exemplo) também não são válidos para relacionar dois jogadores.

Abaixo seguem dois exemplos de requisições CYPHER enviadas ao Neo4J, uma para criar o nó de um jogador e a outra para relacionar dois jogadores.

Primeiro a criação de um nó:

```
CREATE (p:Player {name:"Lionel Messi",
                  transfermarkt_id:28003,
                  score:8434}
        )
```

Depois, a criação da relação entre dois jogadores:

```
MATCH
  (a:Player),
  (b:Player)
WHERE a.transfermarkt_id = 28003 AND b.transfermarkt_id = 44352
CREATE (a)-[rab:PLAYED_WITH {
                  seasons: [2014, 2015, 2016, 2017, 2018, 2019],
                  teams:[131],
                  cost:-6,
                }
        ]->(b)
RETURN type(rab)
```

Neste último comando, é possível perceber que o relacionamento (aresta) que liga dois jogadores contém quatro atributos diferentes. O primeiro e o segundo, “seasons” e “teams”, correspondem a lista de temporadas e times, respectivamente, em que os jogadores jogaram juntos. O terceiro e último atributo, “cost”, é calculado da seguinte forma:

$$cost = -1 * len(seasons) * len(teamsIds)$$

“len(seasons)” e “len(teams)” representam a quantidade de temporadas e times presentes em “seasons” e “teams”, respectivamente.

A ideia é que quanto mais temporadas e clubes presentes ligando dois jogadores, menor é a distância entre os nós no grafo. Dessa maneira, é possível realizar análises considerando arestas com ou sem peso, que serão apresentadas na Seção 6.3.

6.3 Métricas próprias da estrutura do grafo

Esta seção descreverá cada uma das métricas e fará a exposição dos resultados obtidos ao executá-las em 768 jogadores inseridos no banco de dados. Não é possível garantir que o registro de cada atleta apresenta suas informações completas, mas o importante é analisar o que é possível alcançar com cada uma das métricas.

Para todas essas métricas foram utilizados os algoritmos nativos da biblioteca *Graph Data Science* (GDS) do Neo4J, então a implementação desses algoritmos fica fora do escopo deste projeto. Além disso, o grafo do Neo4J, que por natureza precisa ser direcionado, é convertido em um não direcionado para que realmente faça sentido os resultados, já que o relacionamento “X joga juntamente com Y” não possui direção.

Outro fato importante é que essas métricas não utilizam a pontuação dos jogadores para obter seus resultados, elas apenas se aproveitam da estrutura do grafo, principalmente as arestas, para fornecer informações que não são possíveis de serem obtidas apenas com a fórmula explicada anteriormente.

6.3.1 Maior menor caminho (sem custo)

Como introduzido no Capítulo 2, um caminho entre dois nós de um grafo é a sequência de nós (ligados por pelo menos uma aresta) que possibilitam sair de um nó A e atingir outro nó B. O fato de ser o menor caminho significa que a quantidade de nós entre o nó de origem e o de destino é mínima. No caso do caminho sem custo, cada aresta tem a mesma distância, que é representada por 1 unidade. Sendo assim, a ideia da métrica é analisar qual o maior caminho dentre todos os menores caminhos entre cada par de jogadores.

A consulta:

```
CALL gds.allShortestPaths.stream('undirectedGraph')
YIELD sourceNodeId, targetNodeId, distance
WITH sourceNodeId, targetNodeId, distance
WHERE gds.util.isFinite(distance) = true
WITH gds.util.asNode(sourceNodeId) AS source,
gds.util.asNode(targetNodeId) AS target, distance
WHERE source <> target
```



```
RETURN source.transfermarkt_id AS source,
target.transfermarkt_id AS target, distance
ORDER BY distance DESC, source ASC, target ASC
```

A consulta devolve uma lista ordenada do maior caminho para o menor. Cada item da lista contém o nó de origem, o nó de destino e a distância entre eles. O algoritmo utilizado por trás dessa consulta é o All Pairs Shortest Path (APSP), que é mais eficiente do que encontrar o caminho mínimo (utilizando Dijkstra [Dijkstra, 1959], por exemplo) para cada par de nós do grafo.

O resultado

O maior menor caminho encontrado contém 11 nós, e na realidade são 13 caminhos com essa distância. É importante notar que, dos jogadores inseridos, muitos são de épocas completamente diferentes, desde os anos 50 até 2023, então é natural que existam caminhos com diversos nós. Além disso, é natural que quanto mais antigos forem os jogadores, principalmente os que atuavam em solo sul americano, menos informações estão presentes em seus respectivos registros.

Um dos 13 caminhos é o seguinte: Waldemar Victorino (Uruguai, anos 70-90), Hugo de León (Uruguai, anos 70-90), Álvaro Gutiérrez (Uruguai, anos 90), Aljosa Asanovic (Croácia, anos 80-90), Roberto Ayala (Argentina, anos 90-2000), Santiago Cañizares (Espanha, anos 90-2000), José Antonio Camacho (Espanha, anos 70-90), Amancio (Espanha, anos 60-70), Ferenc Puskás (Hungria, anos 40-60), Raymond Kopa (França, anos 50-66) e, por fim, Just Fontaine (França, anos 50-60).

6.3.2 Componentes conexas

As duas métricas que tratam da quantidade de componentes do grafo são as seguintes: componentes fortemente conexas e componentes fracamente conexas. Para o caso do grafo deste trabalho, ambas são equivalentes, já que a diferença está em considerar ou não cada aresta direcionada.

A ideia do algoritmo é percorrer o grafo e contar quantas componentes conexas existem. Uma componente é conexa se todos os seus nós formam um subgrafo conexo. Caso todos os 768 nós do grafo fossem conectados a todos os outros, o algoritmo devolveria apenas uma componente, que seria composta por todos os vértices do grafo.

A consulta

```
CALL gds.alpha.scc.stream('undirected_graph')
YIELD nodeId, componentId
RETURN gds.util.asNode(nodeId).name AS Name, componentId AS Component
ORDER BY Component DESC
```

No caso do algoritmo das componentes fracamente conexas, basta substituir “scc” (*Strongly Connected Components* - Componentes Fortemente Conexos) na consulta acima por “wcc” (*Weakly Connected Components* - Componentes Fracamente Conexos) . O algoritmo devolve cada jogador com um número que indica a sua componente, então jogadores com o mesmo número fazem parte da mesma componente.

O resultado

O algoritmo indica que são 43 componentes conexas no grafo. A maior componente conexa contém 714 nós (92%) do grafo, e as componentes restantes são compostas por um ou dois jogadores, em geral. Provavelmente, quanto mais nós forem inseridos no grafo, maior a chance de aumentar esse percentual e diminuir a quantidade de componentes, já que aumentaria a possibilidade de encontrar algum elo que ligasse duas componentes que atualmente são desconexas.

6.3.3 Quantidade de triângulos

A métrica da quantidade de triângulos conta quantos triângulos cada nó forma. Um triângulo é um conjunto de três que estão ligados entre si. Sendo assim, um nó que pertença a muitos triângulos representa um jogador com muitas ligações diferentes, e o oposto significa que o jogador está mais isolado no grafo.

A consulta

```
CALL gds.triangleCount.stream(['undirected_graph'])
YIELD nodeId, triangleCount
RETURN gds.util.asNode(nodeId).name AS name, triangleCount
ORDER BY triangleCount DESC
```

O algoritmo devolve para cada jogador um valor correspondente a quantidade de triângulos em que ele está contido.

O resultado

Os dois jogadores com mais triângulos são Iker Casillas (Espanha, anos 2000-2010) e Xavi (Espanha, anos 2000-2010), ambos com 532 triângulos. Há também 93 jogadores que não possuem nenhum triângulo sendo que a maioria deles faz parte do conjunto de jogadores que atuaram em meados do século XX. Também é possível notar que dos 10 jogadores com mais triângulos, 9 atuaram no Real Madrid ou no Barcelona, clubes espanhóis, nos anos 2010, além de 6 serem da seleção da Espanha campeã da Copa do Mundo de 2010.

Os 10 primeiros são os seguintes: Iker Casillas (Espanha e Real Madrid) e Xavi (Espanha e Barcelona), Cristiano Ronaldo (Real Madrid), Carles Puyol (Espanha e Barcelona), Victor Valdés (Espanha e Barcelona), Zlatan Ibrahimovic

(Barcelona), Hernán Crespo, Sergio Ramos (Espanha e Real Madrid), Andrés Iniesta (Espanha e Barcelona) e Clarence Seedorf (Real Madrid).

6.3.4 K-1 cores

O algoritmo das K-1 cores tenta pintar o grafo com a menor quantidade de cores possível, sendo que a coloração de cada nó precisa respeitar a seguinte regra: nenhum par de nós conexos pode ser pintado com a mesma cor. Esse problema é NP-Completo, então o resultado não necessariamente é o ótimo, e a qualidade de sua resposta pode depender do número de iterações do algoritmo. De qualquer modo, é um resultado que provê mais informações acerca da conectividade do grafo dos jogadores.

A consulta

```
CALL gds.beta.k1coloring.stream(['undirected_graph'])
YIELD nodeId, color
RETURN gds.util.asNode(nodeId).name AS name, color
ORDER BY name
```

O algoritmo devolve a quantidade total de cores necessárias e, para cada jogador, devolve um número representando a sua cor.

O resultado

O número mínimo de cores encontrado para pintar esses 768 jogadores é 20, ou seja, cada cor pinta, em média, 39 nós. Se cada nó fosse vizinho de todos os outros, seriam necessárias 768 cores e, se cada nó fosse desconexo em relação aos outros, seria necessário 1 única cor para pintar todos os nós. Mesmo com a possibilidade desse resultado não ser o ótimo, ele é interessante pois em um grafo com 768 nós, onde cada nó contém diversos vizinhos, esperaria-se que muitas cores fossem necessárias para pintá-lo inteiro, porém na prática não é o que acontece.

6.3.5 Grau de centralidade

Diferente das últimas métricas, esta aqui busca indicar a importância de cada nó dentro do grafo, já que ele devolve quais são os nós mais populares, isto é, que contêm o maior número de vizinhos. Para o contexto dos jogadores de futebol, um jogador com muitos vizinhos é aquele que teve muitos companheiros diferentes ao longo da carreira, e, em geral, isso é resultado do jogador ter passado por muitos times em sua trajetória. O inverso também pode ser válido, ou seja, jogadores que têm poucos vizinhos são aqueles que estiveram em um único clube ao longo da vida, como é o caso de Francesco Totti, da Roma, Carles Puyol, do Barcelona ou o Paolo Maldini, do Milan. Na prática, outros casos de jogadores com poucos vizinhos são aqueles da metade do século XX, que não estão com todas as informações completas no seu registro e que a maioria dos companheiros de clubes também não estão inseridos no banco de dados.

A consulta

```
CALL gds.degree.stream(['undirected_graph'])
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score AS followers
ORDER BY followers DESC, name DESC
```

Para cada jogador, o algoritmo devolve a quantidade de vizinhos que ele contém.

O resultado

O jogador com mais vizinhos é Zlatan Ibrahimovic (Suécia, anos 2000-2010), com 69 vizinhos, seguido de Cristiano Ronaldo (Portugal, anos 2000-2020) com 64 e Hernán Crespo (Argentina, anos 90-2000) com 60. Cada um deles passou por 10, 5 e 7 clubes, respectivamente. Ibrahimovic já se aposentou mas teve uma carreira de mais de 20 anos, Cristiano Ronaldo ainda joga e sua carreira também já passou dos 20 anos, e Hernán Crespo também já se aposentou com uma carreira de mais de 15 anos.

Enquanto isso, Francesco Totti (Itália, anos 2000-2010) tem 20 vizinhos e está em 189°, Carles Puyol (Espanha, anos 2000-2010) tem 49 vizinhos e está em 25° e Paolo Maldini (Itália, anos 90-2000) tem 38 vizinhos e está em 76°.

É claro que a centralidade de um jogador, nesse caso, está diretamente ligada aos outros jogadores inseridos no banco, então seria possível aumentar esse valor para um jogador específico inserindo intencionalmente seus companheiros de equipe ao longo da vida. Como a ideia aqui é mostrar as possíveis análises que cada métrica permite no grafo, não faz diferença quem é o jogador com maior grau e o que isso implica, o importante é exemplificar a capacidade que a estrutura do grafo tem de fornecer informações para além da fórmula.

6.3.6 Máximo de influência

O algoritmo CELF (*Cost Effective Lazy Forward*), recebe um inteiro K e devolve K nós que maximizam o espalhamento no grafo. Basicamente, a ideia é que se um nó tem coeficiente de espalhamento alto, significa que ele é essencial para espalhar uma informação por toda a estrutura. Mais do que se ter muitos vizinhos, significa que os vizinhos desse nó contém muitos vizinhos, e assim por diante. Sendo assim, pode-se dizer que, de forma geral, esses K vértices são os K mais importantes do grafo.

A consulta

```
CALL gds.beta.influenceMaximization.celf.stream(
  ['undirected_graph'],
  {seedSetSize: 10}
)
YIELD nodeId, spread
```

```
RETURN gds.util.asNode(nodeId).name AS name, spread
ORDER BY spread DESC, name ASC
```

Para este caso, o algoritmo foi rodado com $K = 10$, logo, o resultado contém os 10 jogadores com os maiores coeficientes de espalhamento do grafo.

O resultado

Os 10 jogadores são os seguintes: Zlatan Ibrahimovic (Suécia, anos 2000-2010), Cristiano Ronaldo (Portugal, anos 2000-2020), Lionel Messi (Argentina, anos 2000-2020), Cesare Maldini (Itália, anos 50-60), Karl-Heinz Rummenigge (Alemanha, anos 70-90), Roberto Matosas (Uruguai, anos 60-70), Luis Suárez (Uruguai, anos 2000-2020), Wim Suurbier (Holanda, anos 70-80), Manuel Bento (Portugal, anos 70-90) e, por último, Beom-sok Oh (Coreia do Sul, anos 2000-2020).

A diferença entre o valor do coeficiente de espalhamento do primeiro lugar para os outros é gritante: Ibrahimovic contém 307.0 de coeficiente e Cristiano Ronaldo 3.82. O jogador sueco foi o primeiro lugar em outra métrica, a do grau de centralidade (Seção 6.3.5), e os dois resultados corroboram entre si na conclusão de que a importância desse jogador para o grafo é alta. Além disso, é possível perceber que, com esses 10 jogadores, a maioria das temporadas desde 1950 até 2023 é contemplada, tendo representantes para cada uma das décadas entre esse período. Esse fato também faz sentido tendo em vista o objetivo do algoritmo, pois para espalhar uma informação no grafo seria necessário conter representantes de diversos componentes distintos da estrutura, a fim de atingir o maior número de nós possíveis.

6.4 Relação pontuação - grafo

Na Seção 6.3, foram analisadas formas de se obter informações valiosas utilizando apenas a estrutura do grafo e deixando a fórmula e suas pontuações de lado. Agora, será apresentada uma outra métrica que utiliza tanto os vértices e arestas do grafo quanto os resultados da aplicação da fórmula em cada jogador. Note que, apesar dos resultados da aplicação da fórmula serem inteiramente apresentados somente no Capítulo 8, nesta seção algumas pontuações serão exibidas para auxiliar na exposição desta métrica.

6.4.1 O algoritmo

A ideia deste algoritmo é fornecer respostas para a seguinte indagação: um jogador com uma pontuação X está rodeado de jogadores com uma pontuação semelhante? Ou seja, existe alguma relação entre a pontuação de um jogador e a pontuação de seus vizinhos quando consideramos a carreira inteira dos atletas?

Para atingir esse objetivo, o procedimento é simples. Primeiramente, a lista de jogadores presentes no BD é percorrida e, para cada jogador, encontra-

se todos os seus vizinhos. Então, calcula-se a média da pontuação total e da componente individual dos vizinhos. Sendo assim, para cada jogador existem 5 informações: seu *score* total (soma dos três componentes), seu *score* individual (apenas o componente descrito na Seção 5.2), a média dos *scores* totais dos vizinhos, a média dos *scores* individuais dos vizinhos e, por fim, a quantidade de vizinhos. Como exemplo, a Tabela 6.2 mostra esses valores para o jogador Lionel Messi.

Lionel Messi	
Pontuação total:	8491
Pontuação individual:	5115
Pontuação (média) total vizinhos:	2475
Pontuação (média) individual vizinhos:	847
Total de vizinhos:	46

Tabela 6.2: Exemplo de saída do algoritmo para o jogador Lionel Messi

O motivo pelo qual a pontuação da componente individual também é contemplada nessa métrica é o fato de que, como ela não considera nenhuma conquista coletiva, não importa diretamente quais foram os times pelos quais o jogador passou em sua carreira. Isso permite que a análise seja mais justa ao comparar a pontuação de um jogador com a de todos os seus vizinhos. É claro que, ao mesmo tempo, jogadores que não tenham características ofensivas podem ser prejudicados por esse cálculo, e, por esse motivo, é importante usar tanto a pontuação total quanto a individual para se chegar em qualquer conclusão a partir dessa métrica.

6.4.2 Os resultados

Os resultados apenas consideram jogadores com 5 ou mais vizinhos, para que a comparação seja mais justa, então no total 594 jogadores participaram dessa métrica. Além disso, para cada jogador, há duas maneiras de comparar sua pontuação com a de seus vizinhos: pela diferença *jogador – vizinhos* ou pela razão *jogador/vizinhos*.

Na Tabela 6.3 são expostos os resultados médios dos jogadores, em comparação com seus vizinhos. Pode-se perceber que, em geral, os jogadores possuem sua pontuação menor que a dos seus vizinhos, tanto a total quanto a individual, mas que a discrepância é maior quando consideramos a pontuação total.

A Tabela 6.4 mostra os 5 jogadores com maior diferença em comparação com os vizinhos, enquanto a Tabela 6.5) mostra os 3 jogadores com a pior pontuação em comparação com seus vizinhos, considerando a pontuação total. O mesmo é feito com a razão entre as pontuações (também considerando a pontuação total) nas tabelas 6.6 e 6.7.

	Diferença média	Desvio padrão das diferenças	Razão média	Desvio padrão das razões
Pontuação total	-163.90	972.78	0.89	0.74
Pontuação individual	-63.76	504.44	1.01	2.44

Tabela 6.3: Tabela contendo a diferença e a razão entre a pontuação total e individual dos jogadores comparadas com a de seus vizinhos

	Jogador	Diferença (pontuação total)	N° de vizinhos
1°	Lionel Messi	6016	46
2°	Cristiano Ronaldo	5986	64
3°	Andrés Iniesta	3460	47
4°	Thierry Henry	3136	53
5°	Ronaldo	3020	50

Tabela 6.4: Top 5 jogadores no quesito diferença entre suas pontuações totais e a de seus vizinhos

	Jogador	Diferença (pontuação total)	N° de vizinhos
594°	Pablo Garcia	-2433	17
593°	Pedri	-2393	15
592°	Ansu Fati	-2366	18

Tabela 6.5: Piores 3 jogadores no quesito diferença entre suas pontuações totais e a de seus vizinhos

	Jogador	Razão (pontuação total)	N° de vizinhos
1°	Eusébio	3.97	5
2°	Cristiano Ronaldo	3.94	64
3°	Lionel Messi	3.43	46
4°	Zico	3.07	5
5°	Claudio Taffarel	2.97	9

Tabela 6.6: Top 5 jogadores no quesito razão entre suas pontuações totais e a de seus vizinhos

	Jogador	Razão (pontuação total)	N° de vizinhos
594°	Beom-seok Oh	0.01	5
593°	Pablo Garcia	0.02	17
592°	Pedri	0.04	15

Tabela 6.7: Piores 3 jogadores no quesito razão entre suas pontuações totais e a de seus vizinhos

Na sequência, são exibidas 4 tabelas realizando o praticamente o mesmo cálculo, só que considerando as pontuações individuais dos jogadores em vez das totais. São elas as tabelas 6.8, 6.9, 6.10 e 6.11.

	Jogador	Diferença (pontuação individual)	N° de vizinhos
1°	Cristiano Ronaldo	4420	64
2°	Lionel Messi	4268	46
3°	Robert Lewandowski	2146	29
4°	Luis Suárez	1966	39
5°	Zico	1741	5

Tabela 6.8: Top 5 jogadores no quesito diferença entre suas pontuações individuais e a de seus vizinhos

	Jogador	Diferença (pontuação individual)	N° de vizinhos
594°	Ansu Fati	-1048	18
593°	Pedri	-1045	15
592°	Fernando Muslera	-931	6

Tabela 6.9: Piores 3 jogadores no quesito diferença entre suas pontuações individuais e a de seus vizinhos

O que mais chama atenção nesses resultados são as presenças repetidas de Lionel Messi e Cristiano Ronaldo e a grande diferença de pontuação entre eles e seus respectivos vizinhos, considerando qualquer um dos 4 modos de comparação. Isso mostra que, apesar de estarem em grandes clubes ao longo de suas carreiras, apenas isso não foi suficiente para atingirem pontuações tão altas, já que se esse fosse o caso seus vizinhos teriam pontuações igualmente grandes. A explicação para a pontuação final desses jogadores ser tão alta é, portanto, a total dominância deles nas três componentes em relação a seus companheiros de time ao longo de suas carreiras.

Do outro lado, nos piores jogadores, considerando essa métrica, apareceram diversas vezes Ansu Fati (Espanha, anos 2020) e Pedri (Espanha, anos 2020), que

	Jogador	Razão (pontuação individual)	N° de vizinhos
1°	Zico	9.17	5
2°	Cristiano Ronaldo	7.16	64
3°	Lionel Messi	6.03	46
4°	Eusébio	5.36	5
5°	Gunar Nordahl	4.57	5

Tabela 6.10: Top 5 jogadores no quesito razão entre suas pontuações individuais e a de seus vizinhos

	Jogador	Razão (pontuação individual)	N° de vizinhos
594°	Fabian Basuadlo	0.01	5
593°	Beom-seok Oh	0.02	17
592°	José Basualdo	0.05	15

Tabela 6.11: Piores 3 jogadores no quesito razão entre suas pontuações individuais e a de seus vizinhos

são dois jovens (21 e 20 anos, respectivamente) considerados muito promissores tanto no Barcelona quanto na seleção espanhola. A presença deles neste cálculo é justificada por dois motivos: o fato de serem jovens e o fato de jogarem no Barcelona. As duas características contribuem para essas aparições, já que, além de não terem tido tempo de serem campeões e marcarem muitos gols por terem pouca idade, estão em um time muito vitorioso, com diversos companheiros acostumados a serem campeões e atuarem no altíssimo escalão do futebol, fazendo com que a média das pontuações de seus vizinhos seja bem mais alta que a deles próprios.

Capítulo 7

Interface de Usuário

A interface de usuário desenvolvida para o sistema objetivou atender cinco grandes requisitos funcionais: i) permitir a inserção dos dados estáticos (países, continentes e campeonatos), ii) permitir a visualização e interação com o grafo, iii) permitir a visualização e manipulação dos dados/pontuações dos jogadores, iv) permitir a inserção dos dados dinâmicos (jogadores e times), e, por fim, v) permitir o gerenciamento de novas inserções de times e campeonatos.

É importante notar que a maioria desses requisitos funcionais poderia ser atendida por uma interface de linha de comando. Porém, como a intenção é facilitar o uso do sistema por pessoas com pouco ou nenhum conhecimento na área de computação, essa não seria a opção ideal, já que é bem menos intuitiva e demanda familiaridade com programas desta categoria. Sendo assim, as próximas subseções descreverão como cada um dos requisitos foi cumprido, seja implementando concretamente uma solução, utilizando alguma ferramenta já existente ou mesclando ambas opções.

7.1 Inserção de dados estáticos com o MongoDB Compass

O MongoDB Compass ¹ é a GUI (*Guided User Interface*) oficial para o SGBD MongoDB, ou seja, a princípio é possível realizar qualquer tipo de interação com o banco de dados utilizando essa ferramenta. Para o caso do sistema aqui apresentado, como existem inserções automatizadas e advindas de fontes externas, o Compass não é suficiente para todas as interações com o BD operacional. Mesmo assim, para dados contidos nas coleções estáticas (países, continentes e campeonatos), a utilização dessa ferramenta é a maneira mais fácil de atualizar, remover ou inserir novos documentos. O ponto negativo de usar essa solução é que, eventualmente, se o BD é apagado, seja intencionalmente ou por alguma falha, esses dados não são persistidos, já que os *scripts* de inicialização não

¹ <https://www.mongodb.com/products/tools/compass>

acompanham a evolução do BD. Porém, é razoável pensar que essa é uma característica inerente à qualquer SGBD, e é por esse motivo que existem diversas maneiras de realizar *backups* automáticos de todos os documentos presentes no sistema.

Na Figura 7.1, é possível ver como um documento da coleção **CHAMPIONSHIP** é exibido ao usuário. Além disso, o botão *ADD DATA*, à esquerda da imagem, permite a inserção de novos registros.

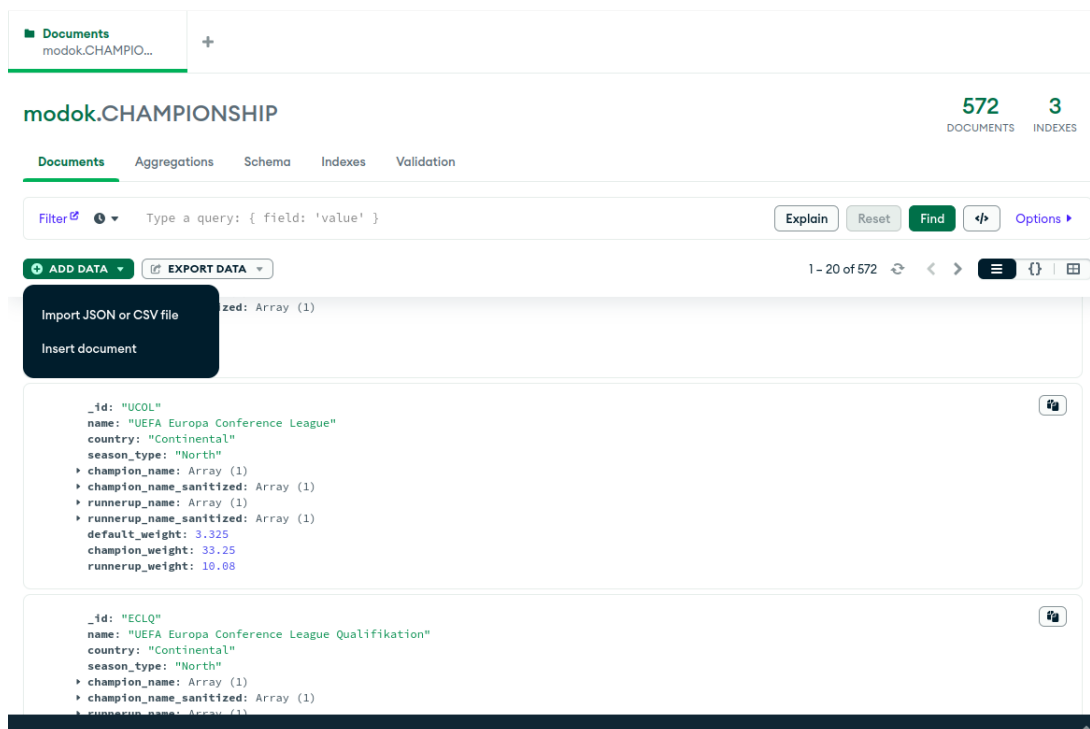


Figura 7.1: Screenshot exibindo a interface do MongoDB Compass

Além de possibilitar a inserção de dados, o Mongo Compass também permite a realização de consultas utilizando MQL (MongoDB Query Language), que é a linguagem de consulta do MongoDB, além da criação de índices e *scripts* para a validação dos dados. Na Figura 7.2, a coleção **COUNTRY** é filtrada para exibir apenas os países com a Ásia como continente.

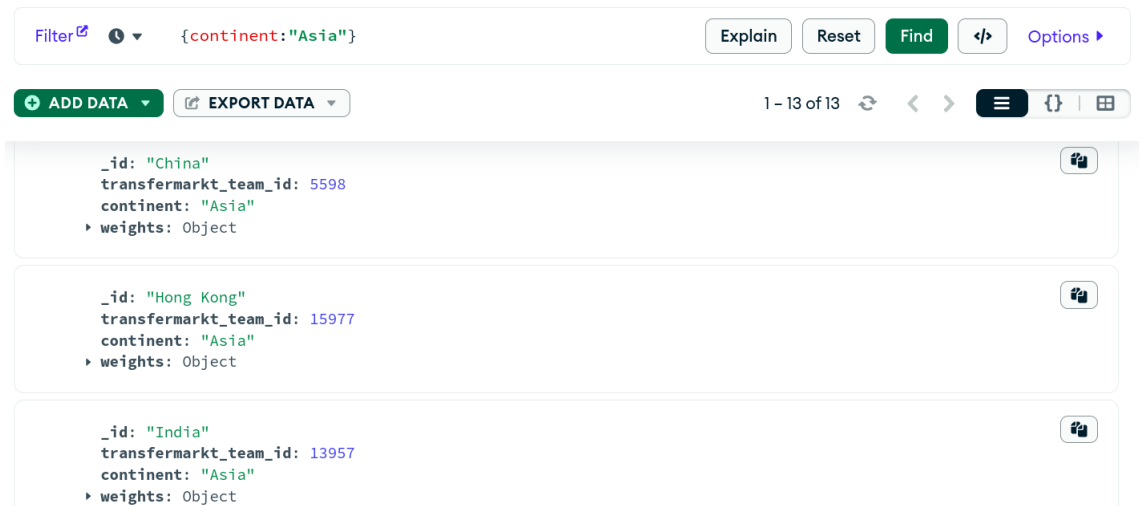


Figura 7.2: Consulta de exemplo realizada no MongoDB Compass

7.2 Visualização e interação com o grafo no Neo4J

O Neo4J, seja ele instalado localmente ou hospedado na própria nuvem da organização, fornece ao usuário dois elementos importantes para a interação com o banco de dados: o próprio sistema gerenciador, que é capaz de realizar as principais operações no grafo (inserção, remoção, atualização e consulta), e uma interface web que também permite modificar os dados, mas que tem como principal benefício a visualização dos nós e arestas. Assim, além de permitir a obtenção das métricas do grafo de maneira automatizada, utilizando consultas pré definidas em *scripts* (já descritas nas seções 6.3 e 6.4), se o usuário desejar, também é possível interagir com o grafo manualmente e visualizá-lo pela interface web.

O ponto negativo é que, dependendo da quantidade de nós e arestas do grafo, a visualização completa se torna difícil, e então é necessário filtrar de algum modo a consulta para que os elementos da saída possam ser carregados e visualizados. De qualquer forma, isso é um problema inerente à tentativa de se visualizar um banco com muitos dados, e não depende do gerenciador ou da interface implantada.

Nas figuras 7.3 e 7.4, é possível perceber a diferença de visualização entre duas partes do grafo, uma com poucos nós e outra com muitos, ambas realizadas pela interface web do Neo4J.

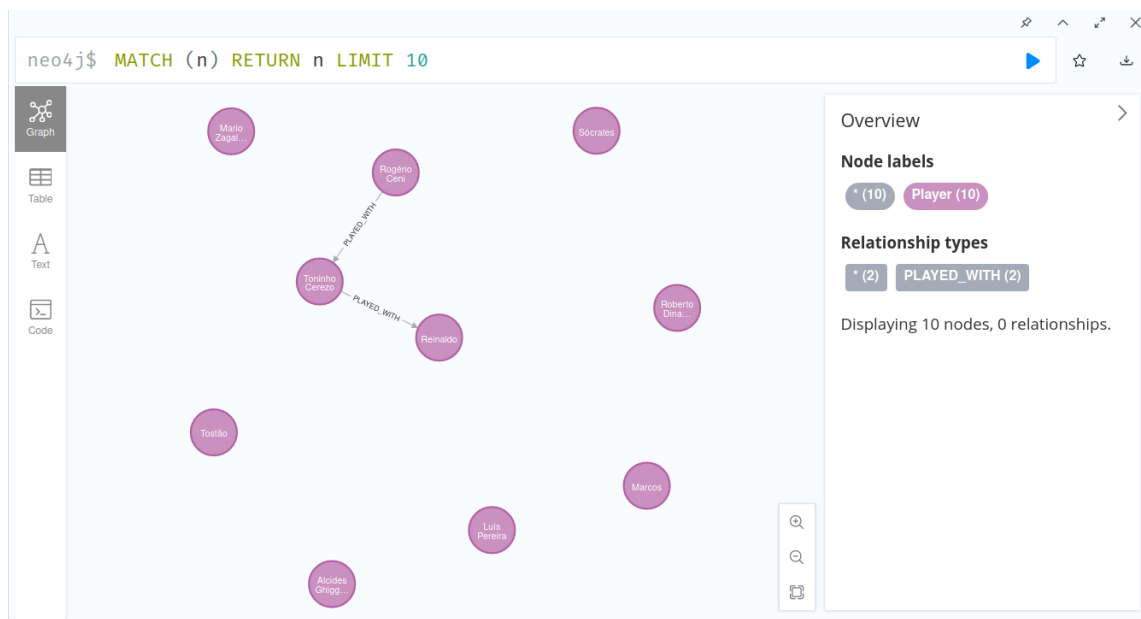


Figura 7.3: Visualização do grafo com poucos nós

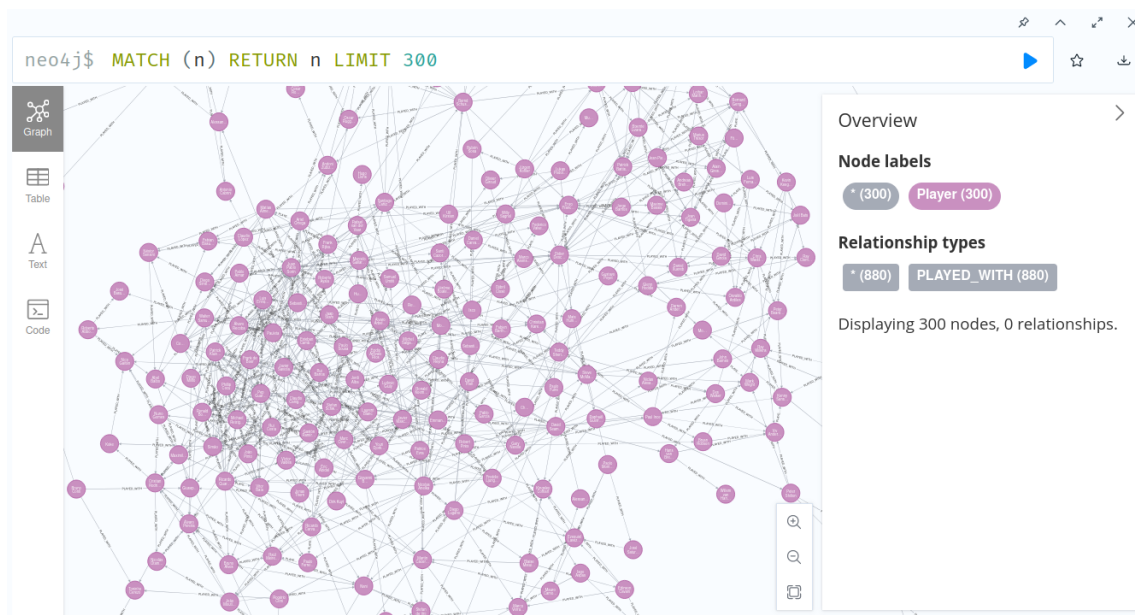


Figura 7.4: Visualização do grafo com muitos nós

7.3 Visualização e manipulação dos dados de jogadores com Apache Superset

Nas seções 7.1 e 7.2, foram apresentadas duas ferramentas que permitem a interação com o usuário mas que já estão totalmente implementadas, isto é, não foram adaptadas ou criadas especificamente para este projeto. Por outro lado, nesta e na Seção 7.4, serão descritas interfaces parcialmente ou totalmente produzidas particularmente para o sistema deste trabalho.

A primeira delas é o Apache Superset ², que é uma poderosa solução de código aberto para visualização de dados, que pode se conectar a diversos SGBDs e é capaz de ser implementada como um servidor web, podendo receber requisições de diversos utilizadores ao mesmo tempo. Assim, aumentando sua escalabilidade e não dependendo do usuário ter acesso local ao banco de dados, a ferramenta permite a interação com todos os dados por meio da rede, inclusive com autenticação, já implementada por padrão no Superset.

A principal funcionalidade dessa ferramenta é a criação de diversos tipos de tabelas e gráficos com pouco ou nenhum esforço, baseando-se nos próprios atributos das relações do BD conectado. A vantagem é que esses elementos de visualização evoluem junto com o banco, já que o Superset mantém uma conexão persistente com o SGBD e, por isso, é capaz de atualizar os gráficos e tabelas concomitantemente com o banco de dados, sem necessidade de reinicialização ou atualização periódica.

7.3.1 A escolha da ferramenta

A visualização de dados é uma necessidade vital no desenvolvimento de qualquer tipo de sistema que interaja com algum BD. Entender os usuários de um produto, monitorar atividades e prevenir erros são alguns exemplos de motivações para implantar uma ferramenta que permita a visualização de dados automaticamente. Além disso, alguns tipos de informação, como dados geográficos, praticamente não fazem sentido se não forem visualizados em um mapa, por exemplo. Dito isso, é esperado que haja uma grande quantidade de ferramentas com esta finalidade disponíveis no mercado, porém, a realidade é que a maioria delas são pagas, ou possuem uma versão paga, ou então não são capazes de se conectar com o MongoDB, usado neste projeto. A Tabela 7.1 compara algumas ferramentas de visualização de dados.

Dessa forma, as principais vantagens do Apache Superset em relação às outras tecnologias são: o suporte total ao MongoDB, mesmo sendo necessário um gerenciador intermediário, e também a facilidade para a criação de novos gráficos a qualquer momento, com todos os modelos de gráficos comuns (barra, setores, linhas, etc.) disponíveis. Outra vantagem é que, como ele é escrito em Python, a integração com a interface customizada também não é muito complexa, como será explicado na Seção 7.4.

² <https://superset.apache.org/>

	Apache Superset	Grafana ^a	D3.js ^b
Empresa	Apache (sem fins lucrativos)	Grafana Labs	Desenvolvido pela própria comunidade
Licença	Apache License	AGPL-3.0 license	ISC license
Preço	Grátis	Grátis (possui versões pagas)	Grátis
Objetivo	Visualização de dados por usuários de diferentes níveis técnicos	Visualização e unificação de dados que podem vir de diversas fontes	Biblioteca para o javascript; Gráficos 100% customizados, criados do zero
Suporte ao MongoDB	Não nativamente. É necessário um intermediário (Apache Drill)	Não nativamente. É necessário um <i>plugin</i> disponível apenas na versão paga	Sim, basta um <i>driver</i> do SGBD para javascript
Outras características	Permite gráficos interativos; Consultas SQL mesmo usando o MongoDB; Criado em Python.	Integração com <i>frameworks</i> de <i>front-end</i> ; Permite configuração de alertas automáticos de acordo com dados.	A ideia dessa solução é criar os gráficos do zero então não é muito fácil de produzir novas visualizações

Tabela 7.1: Gráfico comparativo entre ferramentas de visualização notáveis

^a <https://grafana.com/grafana/>

^b <https://d3js.org/>

7.3.2 Superset e MongoDB

Como introduzido anteriormente, de maneira nativa o Superset é compatível apenas com SGBDs que utilizam SQL e isso poderia ser um empecilho para o desenvolvimento do sistema deste trabalho, já que o MongoDB implementa apenas o MQL (MongoDB Query Language) para suas consultas. A solução para essa incompatibilidade é advinda de outra ferramenta Apache, o Apache Drill³, que é um motor de consultas que possui um *plugin* para se conectar com o MongoDB (ou outros SGBDs não relacionais) e permitir consultá-lo utilizando SQL. Assim como o Superset, o Drill também mantém uma conexão ativa com o SGBD e, portanto, acompanha a evolução dos dados sem necessidade de ser reiniciado. Sendo assim, o fluxo da interação parte de uma atualização no MongoDB (inserção de um jogador novo, por exemplo), passa pelo Apache Drill, que por fim alimenta o Apache Superset com a informação a ser exibida para o usuário.

```
SELECT PLAYER.name, SCORE.Total
FROM mongo.modok.PLAYER
JOIN mongo.modok.SCORE
ON SCORE.transfermarkt_id = PLAYER.transfermarkt_id;
```

Programa 7.1: Consulta no Apache Drill que devolve o nome e a pontuação de cada jogador.

```
db["PLAYER"].aggregate[
```

³ <https://drill.apache.org/>

```

{
  '$lookup':{
    "from":"SCORE",
    "localField":"transfermarkt_id",
    "foreignField":"transfermarkt_id",
    "pipeline": [
      {"$project": {"Total": 1, "_id":0 }}
    ],
    'as':"SCORE"
  },
},
{
  '$project':{
    "_id":0,
    "name":1,
    "SCORE":1,
  }
}
]
)

```

Programa 7.2: Consulta no MongoDB que devolve o nome e o score de cada jogador

O ponto negativo da necessidade de um intermediário é que a consulta se torna mais custosa computacionalmente, e como o Superset e o Apache Drill precisam estar rodando de maneira simultânea, o hardware hospedeiro pode apresentar lentidão dependendo da quantidade de memória disponível. Por esse motivo, é ideal que esses dois serviços estejam em uma máquina separada daquela na qual o usuário visualiza os elementos gráficos.

Em 7.1 e 7.2, é possível notar a diferença de sintaxe entre as consultas no Apache Drill (SQL) e no MongoDB (MQL), ambas visando devolver o nome e a pontuação total de cada jogador.

7.3.3 Gráficos e tabelas

Esta e as próximas seções são dedicadas a descrever as possibilidades fornecidas pelo Apache Superset, tanto na questão gráfica como em modos de interação com o usuário. Primeiramente, o mecanismo de criação de gráficos e tabelas (chamados de *charts* pela ferramenta) será exibido.

Ao criar um novo *chart*, o usuário deve decidir qual tipo de elemento gráfico ele deseja usar como base. As opções são diversas: tabelas, gráfico de barras, linhas, setores, pontos, gráficos de caixa (*boxplots*), geográficos, 3D, e muitos outros. Ele então escolhe seu *dataset* (que pode ser alguma relação física existente no BD ou uma relação virtual criada por alguma consulta), e começa a editar o objeto. A Figura 7.5 ilustra essa etapa inicial.

Em seguida, o usuário deve selecionar quais atributos do *dataset* serão

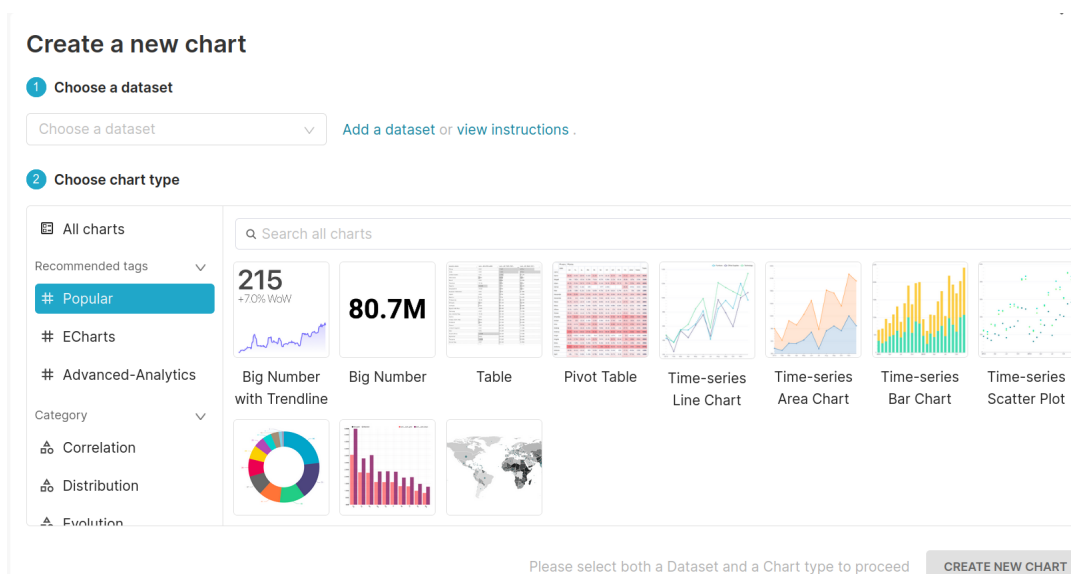


Figura 7.5: Tela para escolher o tipo de gráfico a ser criado

representados por cada eixo, bem como customizar as legendas, eixos, cores, ordenação-padrão, etc. Também é possível adicionar filtros simples, com os operadores básicos (=, <, >, !=, etc.), sem que seja necessário criar uma consulta SQL para isso. Um exemplo desse estágio é exposto na Figura 7.6.

Após criar o *chart*, o usuário deve adicioná-lo a algum *dashboard*. Cada *dashboard* pode conter diversos *charts*, e também pode ser customizado para alterar o espaço destinado a cada gráfico ou adicionar cabeçalhos, divisões, e diversos outros recursos para melhorar sua organização. Por padrão, este projeto contém quatro *dashboards*, que podem ser facilmente importados em qualquer instância do sistema.

7.3 | VISUALIZAÇÃO E MANIPULAÇÃO DOS DADOS DE JOGADORES COM APACHE SUPERSET

Add the name of the chart

Chart Source mongo.modok.PLAYER

Search Metrics & Columns

Metrics Showing 1 of 1

$f(x)$ COUNT(*)

Columns Showing 9 of 9

- ? _id
- abc transfermarkt_id
- abc name
- abc nationality
- abc position
- abc birth_date
- ? Teams
- ? Achievements
- ? Performance

DATA 4k **PIE CHART**

[View all charts](#)

Time Time

Query Query

DIMENSIONS + Drop columns here or click

METRIC + Drop a column/metric here or click

FILTERS + Drop columns/metrics here or click

ROW LIMIT 100

SORT BY METRIC

CREATE CHART

Figura 7.6: Tela de criação do gráfico

Search 768 records...

name	transfermarkt_id	nationality	position	birth_date
Bernard Genghini	117492	France	Attacking Midfield	1958-01-18
Rui Patrício	45026	Portugal	Goalkeeper	1988-02-15
Alexis Sánchez	40433	Chile	Centre-Forward	1988-12-19
Piet Keizer	142888	Netherlands	Left Winger	1943-06-14
Paulo da Silva	38800	Paraguay	Centre-Back	1980-02-01
Joseph-Antoine Bell	102600	Cameroon	Goalkeeper	1954-10-08
Jakub Blaszczykowski	29835	Poland	Right Winger	1985-12-14
Ray Wilkins	117197	England	Central Midfield	1956-09-14
Wayne Rooney	3332	England	Centre-Forward	1985-10-24
Stefan de Vrij	111196	Netherlands	Centre-Back	1992-02-05
Bixente Lizarazu	210	France	Left-Back	1969-12-09
Marco Materazzi	5778	Italy	Centre-Back	1973-08-19
João Félix	462250	Portugal	Left Winger	1999-11-10
Ludovic Giuly	5299	France	Right Winger	1976-07-10
Juan Román Riquelme	3854	Argentina	Attacking Midfield	1978-06-24
Manuel Bento	117241	Portugal	Goalkeeper	1948-06-25

Figura 7.7: Exemplo de um chart criado, neste caso, uma tabela

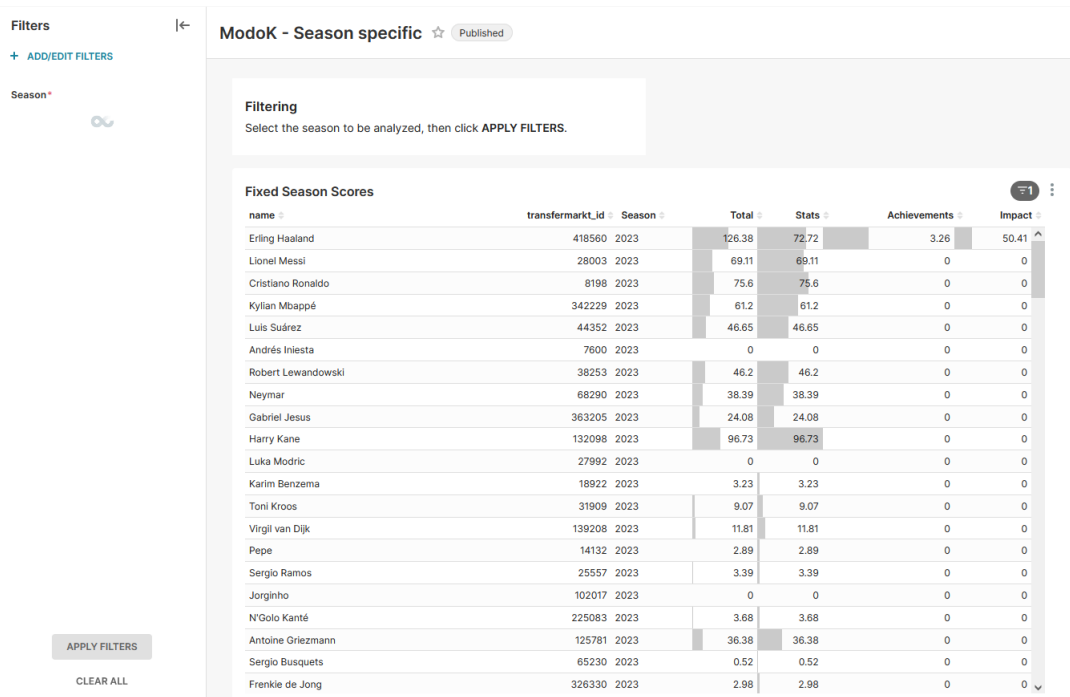


Figura 7.8: Exemplo de um dashboard criado no Superset

7.3.4 Customização de consultas

Por padrão, os *datasets* disponíveis para a criação de gráficos são apenas as próprias tabelas do BD (perceba que agora são tabelas, e não mais coleções, já que o Apache Superset trabalha com o esquema relacional), porém, por meio de uma seção do Superset chamada SQL Lab é possível realizar qualquer consulta e utilizar o seu resultado como produto de um novo *dataset* e, conseqüentemente, de um novo *chart*.

No caso do sistema deste trabalho, por exemplo, as pontuações dos jogadores estão em tabelas diferentes do registro contendo seus nomes, logo, para juntar os dois atributos em um mesmo *chart*, é necessário realizar uma consulta customizada que una os dados (como mostrado no Programa 7.3).

```
SELECT PLAYER.name, SCORE.transfermarkt_id, SCORE.Total,
       SCORE.C1, SCORE.C2, SCORE.C3
FROM mongo.modok.PLAYER
JOIN mongo.modok.SCORE
ON PLAYER.transfermarkt_id = SCORE.transfermarkt_id
```

Programa 7.3: Exemplo de consulta que utiliza duas tabelas para exibir o resultado final.

Outra observação importante é que, para o caso dos dados aninhados provenientes da estrutura de documentos do MongoDB, é necessário usufruir de duas operações suportadas pelo Apache Drill: KVGGEN e FLATTEN.

O KVGGEN retorna uma lista de pares chave → valor contidos em um atributo.

Para entender seu comportamento, suponha o seguinte dicionário:

```
{
  "Peso": {
    "2021": "1",
    "2022": "0.5"
  }
}
```

O comando

```
SELECT KVGGEN('Peso') AS Pesos
FROM coluna
```

devolve a seguinte resposta:

```
|-----|
| Pesos  |
|-----|
| [{"key": "2021", "value": "1"}, {"key": "2022", "value": "0.5"}] |
|-----|
```

Já o comando FLATTEN recebe o próprio KVGGEN e devolve uma linha para cada par chave → valor existente.

```
SELECT FLATTEN(KVGGEN('Peso')) AS Pesos
FROM coluna
```

Assim, sua aplicação no objeto inicial resultaria na seguinte resposta:

```
|-----|
| Pesos  |
|-----|
| {"key": "2021", "value": "1"} |
|-----|
| {"key": "2022", "value": "0.5"} |
|-----|
```

Para acessar cada chave ou valor, basta utilizar `.key` ou `.value`, respectivamente. Esse código SQL:

```
SELECT par.linha.key AS chave, par.linha.value AS valor
FROM (
  SELECT FLATTEN(KVGGEN('Peso')) AS linha
  FROM coluna
) as par
```

gera a seguinte resposta:

chave	valor
2021	1
2022	0.5

7.3.5 Parametrização de consultas

A última funcionalidade essencial para que o Apache Superset cumpra com o objetivo do projeto é a capacidade de realização de consultas parametrizadas. Uma consulta parametrizada é aquela que recebe um ou mais argumentos e devolve a resposta baseando-se no valor destes argumentos. A diferença de uma consulta parametrizada para o uso de um `WHERE = {valor do atributo}` é que, nesse último caso, toda vez que alguém desejasse alterar o parâmetro, a consulta precisaria ser refeita manualmente, ou seja, o usuário se direcionaria à edição de *charts* e manualmente precisaria mudar o valor contido na cláusula `WHERE`. Desse modo, seria preciso conhecimento de SQL para uma simples mudança de parâmetro, como alterar o jogador ou a temporada a ser exibida.

Para solucionar esse problema, são necessários dois componentes fornecidos pelo Apache Superset: os filtros e o *Jinja*.

Jinja

O Jinja ⁴ é um *plugin* do Superset que permite que alguma parte da consulta seja substituída por um valor arbitrário, seja ele proveniente de uma variável de ambiente, parâmetro da URL ou qualquer estrutura que possa enviar esse parâmetro ao Superset. Em geral, essa ferramenta é utilizada para arquivos HTML, XML e até CSV, mas nesse caso o próprio Apache Superset pode fornecer esse argumento para a consulta por meio de seus filtros, detalhados a seguir.

```
SELECT nome, idade
FROM USUARIO
WHERE USUARIO.id = {{ current_username() }}
```

Programa 7.4: Exemplo de parametrização fornecida pelo Jinja.

Na consulta contida no Programa 7.4, `{{ current_username() }}` é o parâmetro que devolve o usuário logado no Apache Superset. Supondo que houvesse uma tabela que representasse os usuários, seria possível consultar o nome e idade do usuário logado usando essa consulta.

Filtros

Os filtros podem ser criados para cada *dashboard* e, no caso padrão, servem apenas para filtrar valores de tabelas já carregadas. Então, se existe um gráfico

⁴ <https://jinja.palletsprojects.com/en/3.1.x/>

contendo todos os jogadores, suas posições e suas pontuações, é possível filtrar o resultado para apenas aqueles que são atacantes, por exemplo.

Figura 7.9: Exemplo de um filtro simples, que permite selecionar a posição dos jogadores exibidos

Além disso, como introduzido anteriormente, os filtros permitem que a consulta seja parametrizada, à medida que o valor sendo filtrado possa ser enviado para a consulta por meio do parâmetro `filter_values()`, do Jinja. Por exemplo, a tabela que contém a performance detalhada de um jogador em toda a sua carreira, não é possível carregá-la para todos os jogadores de uma vez (pois a resposta teria centenas de milhares de linhas). Nesse caso, o filtro permite definir um jogador alvo para a consulta e, assim, a consulta e sua exibição não ficam custosas computacionalmente.

```
WHERE PLAYER.name = (
  {{ "'" + "', '".join(filter_values('name')) + "'" }})
```

Programa 7.5: Filtragem de jogadores na cláusula `WHERE`.

No trecho de uma consulta exibido no Programa 7.5, ocorre a filtragem do jogador pelo seu nome. No *dashboard*, se um filtro no atributo “name” for adicionado, a cada alteração na seleção de um jogador para o filtro, o usuário muda a cláusula `WHERE` e, assim, realiza a consulta apenas para o jogador selecionado.

É importante notar que a característica mais relevante do Apache Superset é seu poder de customização. Assim, baseando-se em todas as funcionalidades descritas nas últimas seções, é possível perceber que o usuário pode construir seus gráficos à medida que se interessa por determinadas análises, e qualquer *chart* já existente também pode ser alterado facilmente. É claro que para realizar as consultas algum conhecimento de SQL é recomendado. Porém, para a parte gráfica, de ordenação e filtragem, quase nenhum pré-requisito técnico seria necessário. O Apache Superset, portanto, se comprova como uma ferramenta muito maleável e com grande poder de representação, tanto graficamente como em relação a sua capacidade de apoiar consultas complexas.

7.4 Inserção de dados dinâmicos de jogadores, times e campeonatos via interface customizada

O último componente a ser descrito nesta seção é a interface, criada praticamente do zero, para permitir o acompanhamento das inserções automáticas e o gerenciamento de novas adições de times e campeonatos.

7.4.1 Objetivo detalhado

As ferramentas descritas pelas seções 7.2 e 7.3 supõem que as informações estão presentes no MongoDB, tanto para visualização (Apache Superset) como para a conversão para um SGBD orientado a grafos (Neo4J). Já a ferramenta descrita na Seção 7.1 permite a inserção apenas das informações estáticas, como países, continentes e campeonatos. Falta, portanto, alguma tecnologia que interaja com os dados coletados de forma automatizada, advindos da web.

É nesse contexto que há a necessidade de criar uma interface implementada especificamente para este projeto, que se liga com o resto do sistema descrito anteriormente (extração, cálculo, inserção, etc.) e permite que o usuário atinja dois objetivos: a realização e acompanhamento das inserções automatizadas e o gerenciamento de novas adições de times e campeonatos. O primeiro objetivo envolve enviar uma lista de identificadores a serem inseridos/atualizados ou removidos, tanto de times quanto de jogadores, e então, ao final de todas as extrações requisitadas, o usuário deve ser informado de possíveis erros durante cada inserção. O segundo objetivo é consequência da evolução dos dados, já que à medida que novos jogadores e times são adicionados, é natural que apareçam referências em seus registros a outros times ou campeonatos que não estejam presentes no esquema. O usuário, então, deve poder filtrar quais desses novos registros devem ser adicionados e quais devem ser ignorados intencionalmente.

Percebe-se que ambos objetivos podem ser cumpridos por interfaces de linha de comando, muito mais fáceis de serem implementadas, mas que têm como principal desvantagem a dificuldade de interação para pessoas não familiarizadas

com esse tipo de programa. Como o objetivo do sistema é justamente facilitar ao máximo o uso por não especialistas da computação, criar uma interface de linha de comando não seria suficiente para cumprir esse intuito. A solução encontrada, portanto, foi a produção de uma interface web, descrita em detalhes a seguir.

7.4.2 Flask e Apache Superset

Flask ⁵ é um *framework* web em Python conhecido por sua simplicidade em relação a outros *frameworks* notáveis, como o Django, por exemplo. Com essa aplicação, é possível configurar as rotas de um servidor e renderizar os arquivos HTML sem que seja necessário muito conhecimento prévio sobre *back* ou *front-end*. O Apache Superset, já descrito na Seção 7.3, utiliza o Flask e, por isso, esse *framework* foi a base da interface implementada.

O fato do Superset já utilizar essa ferramenta, a princípio, poderia se tornar um empecilho para o desenvolvimento da interface customizada. A questão principal é que a implementação do Apache Superset não faz parte do repositório [L. B. Salvador, 2023], ela é apenas uma instalação do gerenciador de pacotes do Python, como qualquer outra biblioteca da linguagem, logo, para customizar sua instância talvez fosse necessário alterar os arquivos nativos da ferramenta. Outra possibilidade seria executar dois servidores web ao mesmo tempo, porém isso dificultaria a interação entre as interfaces. Para resolver o problema e permitir que ambas as interfaces coexistam, foi necessário alterar a maneira como o Flask é inicializado.

A solução encontrada foi instanciar o Superset dentro de um novo arquivo `__init__.py` (reconhecido automaticamente pelo Flask como o criador da aplicação), como uma herança de sua implementação. Dessa maneira, ao rodar o Flask dentro do repositório, toda a implementação nativa do Superset é criada, mas com a vantagem de que novos *endpoints* e configurações podem ser estendidos para a mesma instância do *framework*, sendo necessário apenas alterar esse arquivo `__init__.py` contido no repositório. A única configuração do Superset que precisa ser alterada é o endereço da página inicial, hospedada por padrão em `/superset` mas que agora deveria ser apenas em `/modok` (apelido do sistema), que por sua vez contém um botão para o usuário navegar ao Apache Superset ou aos outros módulos da interface.

```
import os
from flask import Flask
from superset.initialization import SupersetAppInitializer

def create_app() -> Flask:
    app = Flask("superset")
    config_module = os.environ.get(
        "SUPERSET_CONFIG", "superset.config"
```

⁵ <https://flask.palletsprojects.com/en/3.0.x/>


```

)
app.config.from_object(config_module)
app_initializer = app.config.get(
    "APP_INITIALIZER", SupersetAppInitializer
)(app)
app_initializer.init_app()

@app.route('/modok/')
def menu():
    return "ModoK custom route"

return app

```

Programa 7.6: Inicialização do Flask contendo o Superset e rotas customizadas.

Como é possível ver no Programa 7.6, a função inicializadora do contexto do Flask precisar ser modificada para permitir a coexistência do Apache Superset com as novas rotas da interface, nesse caso, /modok/. Mesmo assim, as alterações necessárias são pequenas, então a adaptação é plenamente possível e não interfere no restante da interface web a ser descrita nesta seção.

7.4.3 Interação com o sistema

Com o Superset e a nova interface coexistindo, a próxima etapa realizada foi integrá-la com os fragmentos que fazem a extração, o cálculo e o armazenamento das informações, ou seja, com o coração do sistema. A princípio, essa seria uma tarefa fácil. Estando todas as funções já implementadas, bastaria realizar suas chamadas de acordo com a requisição do usuário, e caberia mais ao *front-end* a responsabilidade para com a interface. O problema é que, neste caso, as operações realizadas pelo sistema não são instantâneas, muito pelo contrário, pois a inserção de um jogador pode demorar até dois minutos. Por isso, se o servidor esperasse a extração ser finalizada para responder ao usuário, é possível que a maioria dos *browsers* atuais abortasse a conexão antes de receber a resposta alegando tempo limite excedido.

A solução para este problema foi utilizar uma *thread* para cada operação a ser realizada, de modo a não congelar as ações do servidor enquanto ele trabalha e permitir que a resposta seja fornecida à medida que as informações são obtidas. A vantagem de se utilizar Python nesses casos é que é muito fácil paralelizar uma função, basta usar a biblioteca *multiprocessing*, que também foi usada no sistema na implementação das requisições de múltiplas extrações de dados de jogadores ou times.

O procedimento é o seguinte: o usuário faz a requisição de certa operação; o processo é criado, mas não inicializado; o servidor devolve uma página de carregamento para o usuário e, por fim, a nova *thread* é iniciada e a operação começa. Conforme o andamento dos processos evolue, a tela de carregamento atualiza e, quando tudo é finalizado, o usuário é redirecionado para uma rota

contendo a lista de entidades alteradas e um código de status para indicar se a operação funcionou ou não. Vale notar que, por conta do esquema de *threads*, mesmo que alguma inserção gere uma exceção e não possa ser finalizada, o servidor continua operando normalmente, ele apenas avisará ao usuário que essa entidade não foi adicionada com sucesso.

No Programa 7.7, é exibida a criação de uma *thread* a cada requisição ao servidor. `operation_manager` e `operation_info` cuidam da operação a ser realizada, então o importante aqui é perceber que a resposta é devolvida ao usuário com *yield* em vez do *return*, de modo que, apesar da tela de carregamento ser recebida pelo usuário, só depois dessa resposta que a nova *thread* é iniciada, por meio de `process.start()`.

```
def process_request(operation_manager, operation_info):

    process = Process(target=operation_manager.start_working,
                     args=[operation_info])

    yield render_template("api-loading.html")

    process.start()
```

Programa 7.7: Inicialização de uma *thread* para realizar as operações de inserção.

Para o gerenciamento das inserções de times e campeonatos, o método é mais simples, já que são operações praticamente instantâneas. Para tal, o sistema precisa ler três arquivos JSON presentes no repositório, em que cada arquivo contém os identificadores correspondentes aos times/campeonatos a serem adicionados (`add.json`), ignorados (`ignore.json`) ou aqueles aguardando alguma decisão (`wait.json`). O usuário requisita uma dessas tabelas e visualiza as entidades presentes nela. Ele pode então decidir alterar alguma entidade de lugar, por exemplo, se ele decide que algum time que estava em espera deva ir para o `add.json`. Após solicitar a alteração e enviar ao servidor, na próxima requisição da tabela esse time já estará no seu novo lugar. Para facilitar a inserção dos times na tabela de adição (`add.json`), é possível copiar todos os identificadores para o *clipboard* (CTRL + C) para que sejam colocados no campo de entrada da operação de inserção de maneira fácil. Note que as tabelas também devem acompanhar a evolução do BD, já que um clube que acabou de ser adicionado deve ser removido de qualquer tabela que ele possivelmente esteja.

```
{
  "14701": {
    "name": "Santos B",
    "NationalTeam": false,
    "TriggerPlayer": [
      302785
    ]
  }
}
```

```

    },
  }
}

```

Programa 7.8: Exemplo da estrutura contida nos arquivos JSON de gerenciamento de inserções.

No Programa 7.8, é possível ver um exemplo da estrutura que gerencia as inserções de times. Nesse caso, o clube “Santos B”, de ID 14701 e referenciado pelo jogador 302785, está contido na tabela `ignore.json` pois trata-se do time secundário do Santos e por isso ele não deve ser inserido no BD.

7.4.4 Resultado

Menu

A página inicial é dividida em três blocos de conteúdo, cada um contém um texto e um botão que redireciona o usuário à outra página. O primeiro é dedicado ao Apache Superset. No segundo bloco, está a descrição da página de operações, onde o usuário consegue fazer as inserções ou remoções de jogadores e de times. No terceiro, é descrita a página de gerenciamento, onde o usuário pode encontrar as tabelas contendo novos times e campeonatos a serem adicionados ou ignorados. A Figura 7.10 mostra uma captura de tela da página inicial.



Figura 7.10: Página inicial da interface

Operações

A página de operações permite que o usuário defina exatamente qual operação deseja realizar, e de que modo. Primeiro, ele escolhe entre inserção/atualização ou remoção. Depois, se a operação tratará de jogadores ou de times. Por fim, ele escolhe qual será seu método de entrada de IDs, ou um arquivo ou um texto. Caso escolha a primeira opção, ele poderá fazer o *upload* do arquivo de inserção, do contrário ele poderá inserir em um campo de texto os IDs separados por vírgula.

Após escolher a operação, o usuário clica no botão “Submit” e é redirecionado para a página de carregamento. A Figura 7.11 mostra uma captura de tela da página de operações.

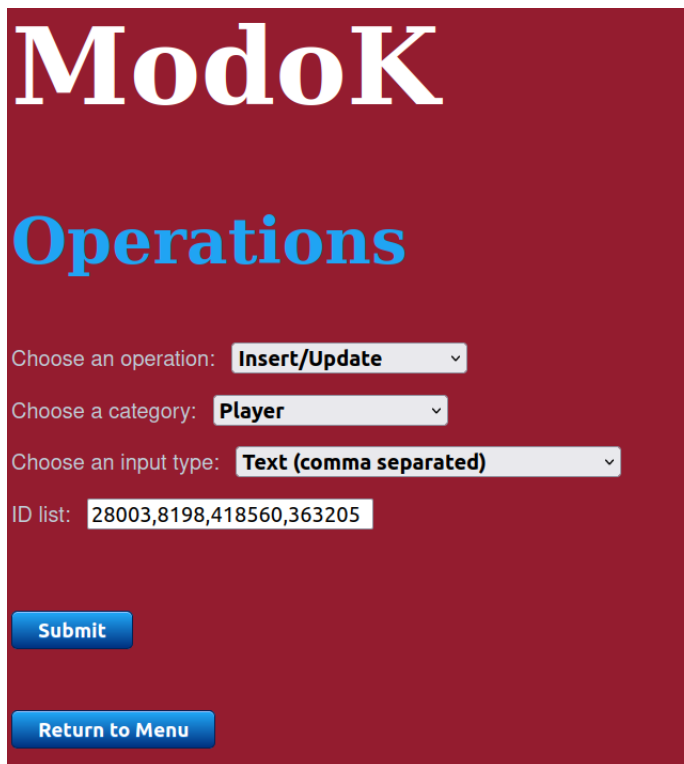


Figura 7.11: *Página de operações da interface*

Gerenciador

Na página de gerenciamento, o usuário deve indicar qual tabela gostaria de gerenciar, Add, contendo os times ou campeonatos a serem inseridos, Ignore, contendo os times ou campeonatos a serem ignorados, ou Wait, que representa as entidades aguardando alguma decisão. Depois, o usuário escolhe se o gerenciamento vai ser de times ou campeonatos. A tabela exibida contém todas as informações úteis para o gerenciamento. No caso de times, apresenta o nome e ID do time, além de indicar se ele é seleção nacional ou não. No caso de torneios, são exibidos o nome e um indicador do tipo de campeonato. Para cada entidade exibida, o usuário pode selecionar para qual das três tabelas ela deve ser movida (ou mantida, caso a tabela atual seja escolhida). Quando ele termina de analisar as opções, pode enviar as alterações clicando no botão Submit. Outra funcionalidade dessa tela é a cópia dos IDs para o *clipboard*. Essa opção é importante caso o usuário deseje obter o ID dos times que quer inserir no BD e rapidamente se direcionar à tela de operações para inseri-los na entrada de texto.

A Figura 7.12 mostra uma captura da tela de gerenciamento, com a tabela Wait dos times.

Manager

Select the table to be managed: **Wait** ▾

Select the entity to be managed: **Teams** **Championships**

Copy IDs to clipboard

TEAMS

WAIT

NAME	ID	NATIONAL TEAM	ACTION
Arsenal Youth	50683	×	ADD <input type="checkbox"/> WAIT <input checked="" type="checkbox"/> IGNORE <input type="checkbox"/>
Rovers FC Youth	112605	×	ADD <input type="checkbox"/> WAIT <input checked="" type="checkbox"/> IGNORE <input type="checkbox"/>
Santos B	14701	×	ADD <input type="checkbox"/> WAIT <input checked="" type="checkbox"/> IGNORE <input type="checkbox"/>
Telford Utd	3694	×	ADD <input type="checkbox"/> WAIT <input checked="" type="checkbox"/> IGNORE <input type="checkbox"/>

Submit

Figura 7.12: *Página de gerenciamento da interface*

Páginas de status

O usuário pode ser redirecionado a três páginas secundárias que se referem a progressão do trabalho de inserção/atualização ou remoção, requisitados na tela de operações.

A página de carregamento é exibida enquanto o usuário aguarda a operação ser realizada. Ela apresenta a porcentagem do progresso até o momento, além de um temporizador indicando a duração da operação que está sendo aguardada.

A página de erro é exibida caso tenha ocorrido algum problema com o servidor durante a realização de uma operação.

A página de resposta é exibida após a finalização de uma operação, e indica todos os jogadores/times requisitados e o status da operação para cada um deles. Caso tenha havido algum problema com a inserção de algum deles, essa tela apresentará uma indicação ao usuário, para que ele tente refazer a operação ou investigue mais a fundo o problema.

Conclusão

Assim, é possível perceber que todos os objetivos da interface foram cumpridos, dado que o usuário pode interagir com as inserções automatizadas no banco, pela tela de operações, e também pode gerenciar as possíveis novas adições do banco, na tela de gerenciamento.

Capítulo 8

Discussão

Neste capítulo, com o auxílio dos gráficos e tabelas criados no *Apache Superset*, serão expostos os resultados da aplicação da fórmula para 768 jogadores. As próximas seções vão seguir a organização dos quatro *dashboards* do Superset produzidos para este trabalho, sendo que cada um deles trata de um tipo de resultado diferente. Infelizmente, para esta monografia, apenas capturas de tela podem ser usadas para exibir os *charts*, mas diretamente pela ferramenta de visualização é possível interagir com cada *chart*, de modo a exibir valores escondidos ou alterar a ordenação das tabelas, por exemplo, facilitando e enriquecendo a obtenção de informação.

8.1 Demografia do banco de dados

Antes de expor as pontuações ou realizar correlações entre os dados, é preciso entender quem são os jogadores, times e campeonatos que compõem o banco de dados. Esta seção tem como objetivo apresentá-los.

8.1.1 Campeonatos

Há 572 campeonatos registrados no BD, mas aproximadamente 20% deles são apenas edições diferentes de um mesmo torneio. Para a Copa do Mundo, por exemplo, existe um registro por edição, então até o momento são 22 instâncias diferentes representando a mesma competição. Esse tipo de ocorrência é mais comum com campeonatos de seleções, como a Copa América, a Eurocopa, a Copa das Confederações, entre outros.

Considerando todos os 572 torneios presentes no BD, 106 (18,53%) são continentais de seleções, 34 (5,94%) são continentais de clubes, 33 (5,77%) são mundiais de seleções, 25 (4,37%) são do Brasil, 21 (3,67%) da Alemanha, e assim por diante, até diversos países que não possuem nenhum campeonato registrado. Essa distribuição é representada pelo gráfico de setores exibido na Figura 8.1.

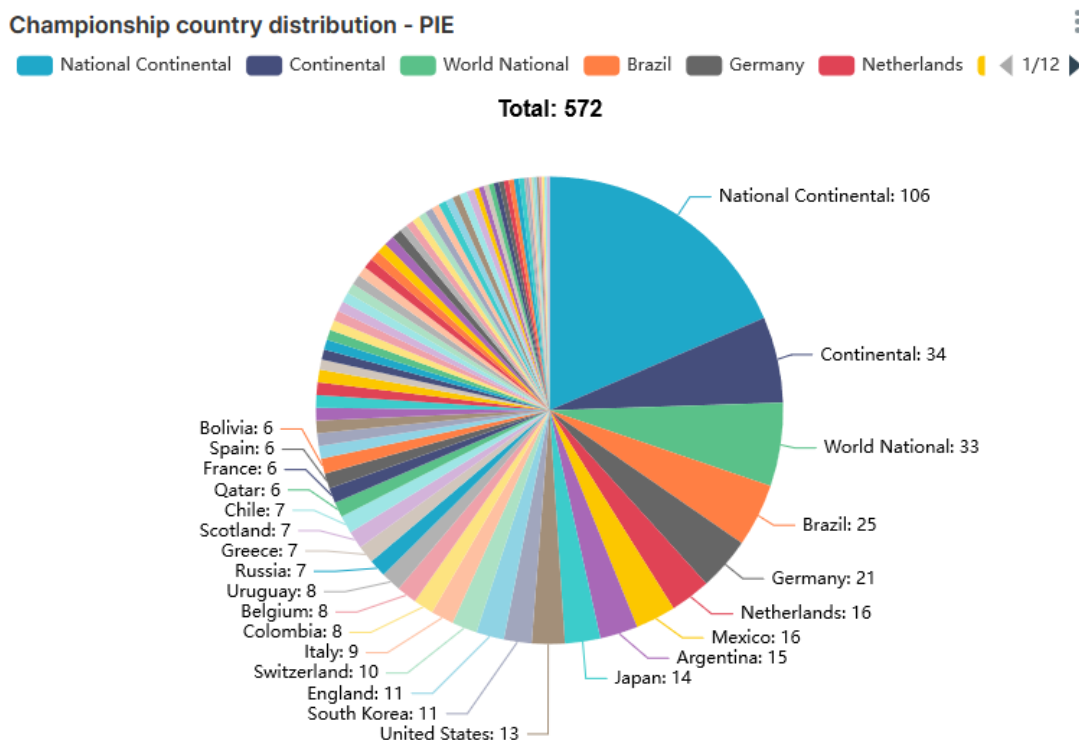


Figura 8.1: Gráfico de setores da distribuição dos campeonatos por países

Além disso, outro gráfico útil para analisar a demografia dos torneios é o que mostra diretamente em um mapa-múndi a quantidade de campeonatos de cada país. Ele também é valioso para exibir a distribuição dos campeonatos por continentes, na qual é possível perceber que a África é o continente com menos torneios inseridos no BD, enquanto a Europa e América do Sul são os dois com mais campeonatos. Na Figura 8.2, esse gráfico é exposto.

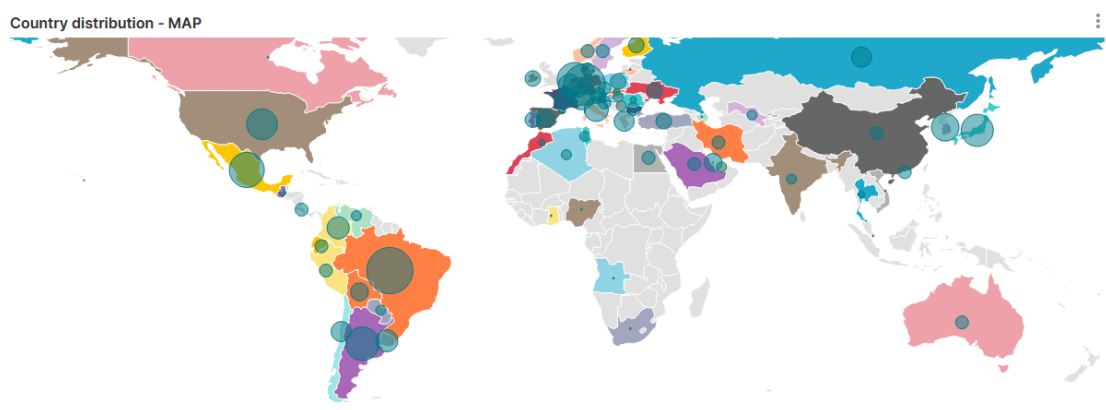


Figura 8.2: Mapa-múndi da distribuição dos campeonatos por países

Sobre o gráfico da Figura 8.2, é necessário apontar que o Superset não reconhece a Inglaterra e os outros países do Reino Unido individualmente, apenas o próprio Reino Unido, então a informação é incompleta para essa região.

De qualquer modo, pelo gráfico da Figura 8.1, é possível obter a informação de que a Inglaterra contém 11 (1,92%) torneios, Escócia 7 (1,22%), Irlanda do Norte 4 (0,69%) e País de Gales 3 (0,52%).

8.1.2 Times

Há 1050 times inseridos no BD, sendo que 103 (9,8%) são seleções nacionais e o restante são clubes. A maioria dos times são da Inglaterra, 77 (7,33%), seguidos da Itália com 69 (6,57%), Alemanha com 67 (6,38%), Espanha com 60 (5,71%), e assim por diante. O Brasil é o 7º país com mais times, com 38 (3,61%). A seguir estão os gráficos de setores (Figura 8.3) e o mapa-múndi representando a distribuição de times pelo planeta (Figura 8.4).

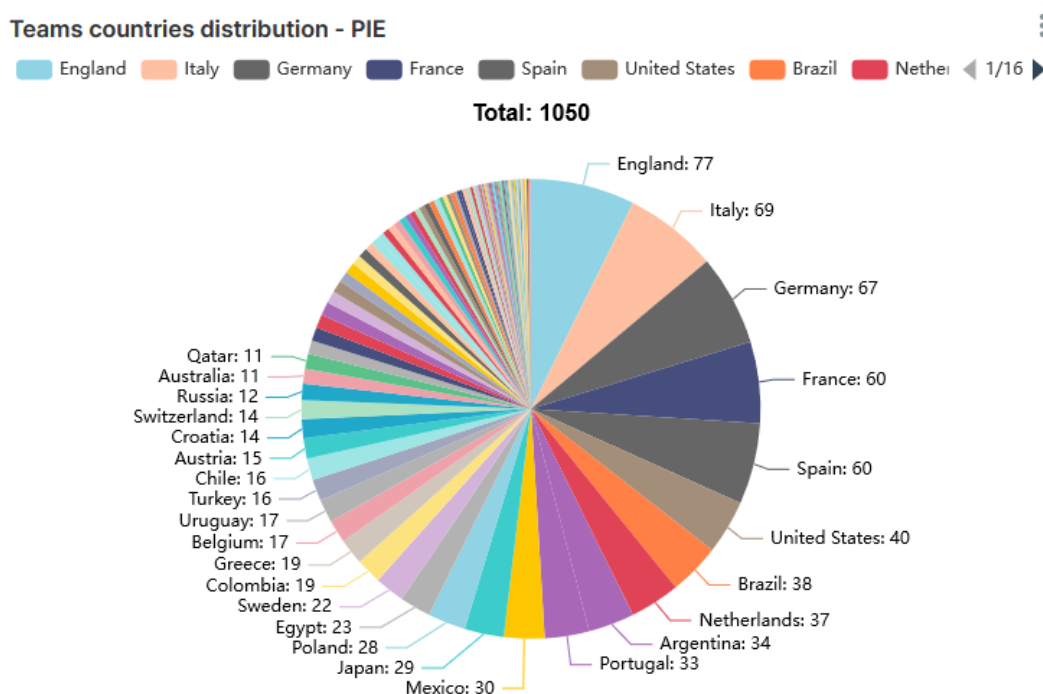


Figura 8.3: Gráfico de setores da distribuição dos times por países

8.1.3 Jogadores

Como citado anteriormente, foram inseridos 768 jogadores no BD. Para entender a composição demográfica desses atletas, é possível dividi-los considerando suas nacionalidades, suas idades/longevidade da carreira e suas posições dentro de campo.

Nacionalidade

Estão presentes jogadores de 42 nações diferentes, a vasta maioria sendo da Europa ou América do Sul. O país com mais jogadores é a Inglaterra com 56 (7,29%), depois a Itália e a Espanha com 53 (6,9%), França e Alemanha com

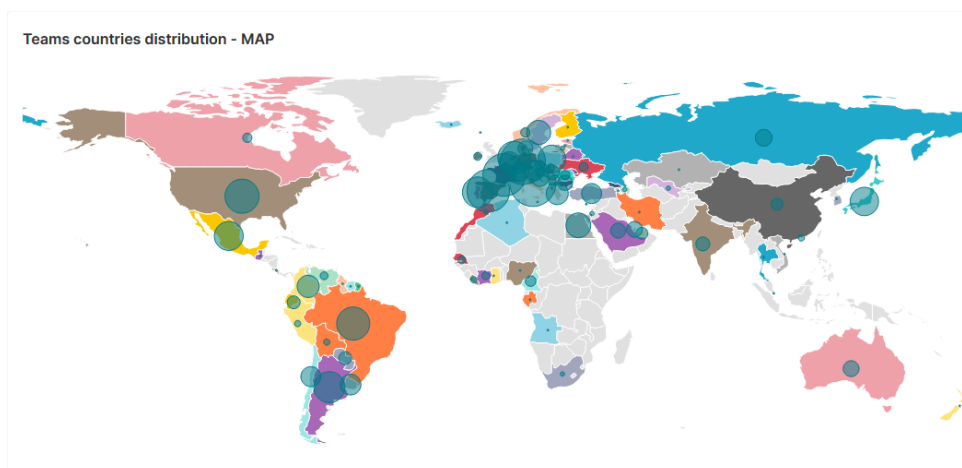


Figura 8.4: Mapa-múndi da distribuição dos times por países. Observação: Inglaterra contém 77 (7,33%) times, Escócia 9 (0,85%) , Irlanda do Norte 2 (0,19%) e País de Gales 4 (0,38%).

51 (6,64%) e o Brasil com 50 (6,51%), como mostra a Figura 8.5. No mapa da Figura 8.6, é possível perceber que os jogadores africanos são distribuídos em poucos países, principalmente o Egito com 19 atletas (2,4%), Camarões e Costa do Marim com 11 (1,43%), e, por fim, o Senegal com 10 (1,3%).

Idade e longevidade

Outras duas características importantes para compreender o perfil dos jogadores inseridos são as idades e longevidades de cada atleta. Um jogador que hoje possui 60 anos teve seu auge físico há aproximadamente 35 anos, logo, estaríamos tratando do futebol dos anos 80-90. Por outro lado, um jogador de 20 anos nem atingiu seu auge, e tratamos, portanto, do futebol da atualidade. A longevidade pode complementar essa informação, já que fornece dados acerca da maturidade da carreira dos atletas contidos no BD. Jogadores com 5 anos de carreira provavelmente não tiveram tanta chance de representar suas seleções em muitas copas, então as suas pontuações são naturalmente mais baixas. Enquanto isso, jogadores com mais de 20 temporadas na carreira provavelmente jogaram até próximo dos 40 anos de idade, logo, têm mais chances de terem mais títulos e participações importantes, levando a pontuações mais altas.

Ambos os gráficos estão divididos em faixas de valores. Para as idades, na Figura 8.7, a divisão começa em [0,20) anos, e nenhum jogador está nessa faixa. Em seguida existem 50 (6,51%) na faixa dos [20,30), 128 (16,67%) dos [30-40), 158 (20,57%) entre [40,50), 163 (21,22%) nos [50-60), 168 (21,88%) entre [60,80) e, por fim, 101 (13,15%) com mais de 80 anos. Para a longevidade, na Figura 8.8, nenhum jogador tem menos de 5 temporadas disputadas, 63 (8,20%) tem de 5 a 10 temporadas, 194 (25,26%) estão entre [10,15), 382 (49,74%) têm entre [15,20] e 129 (16,80%) disputaram mais de 20 temporadas.

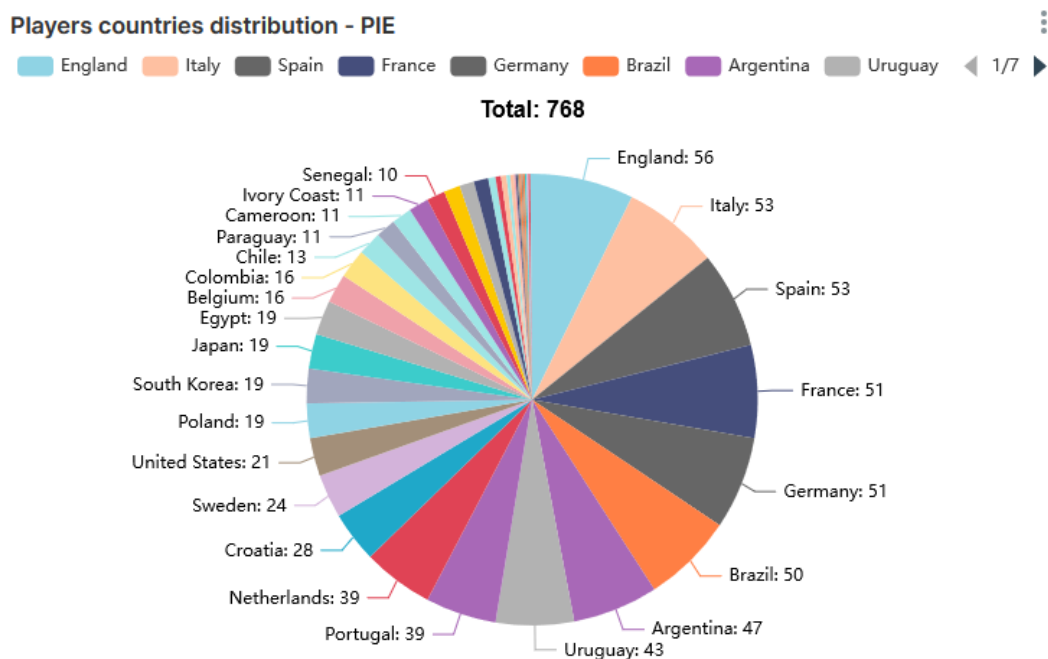


Figura 8.5: Gráfico de setores da distribuição dos jogadores por países

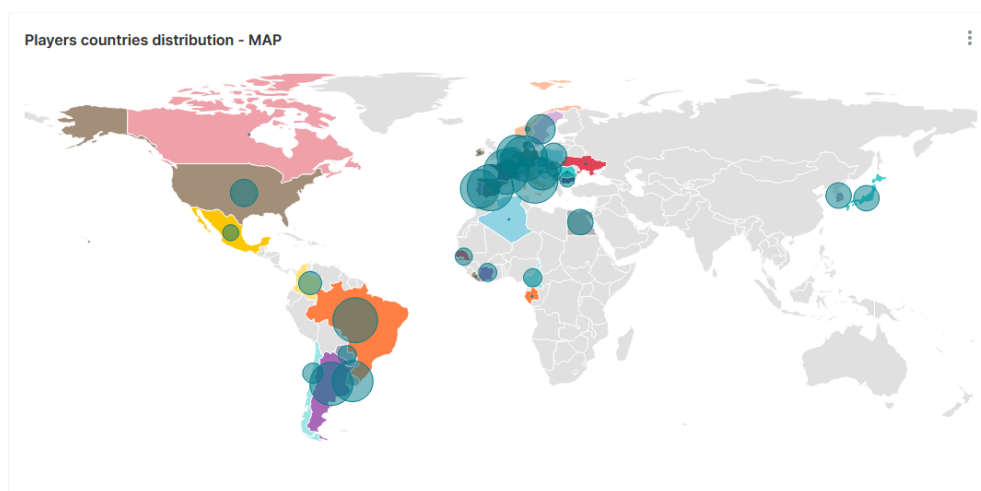


Figura 8.6: Mapa-múndi da distribuição dos jogadores por países. Observação: Inglaterra contém 56 (7.29%) times, Escócia 1 (0.13%), Irlanda do Norte nenhum e País de Gales 2 (0.26%).

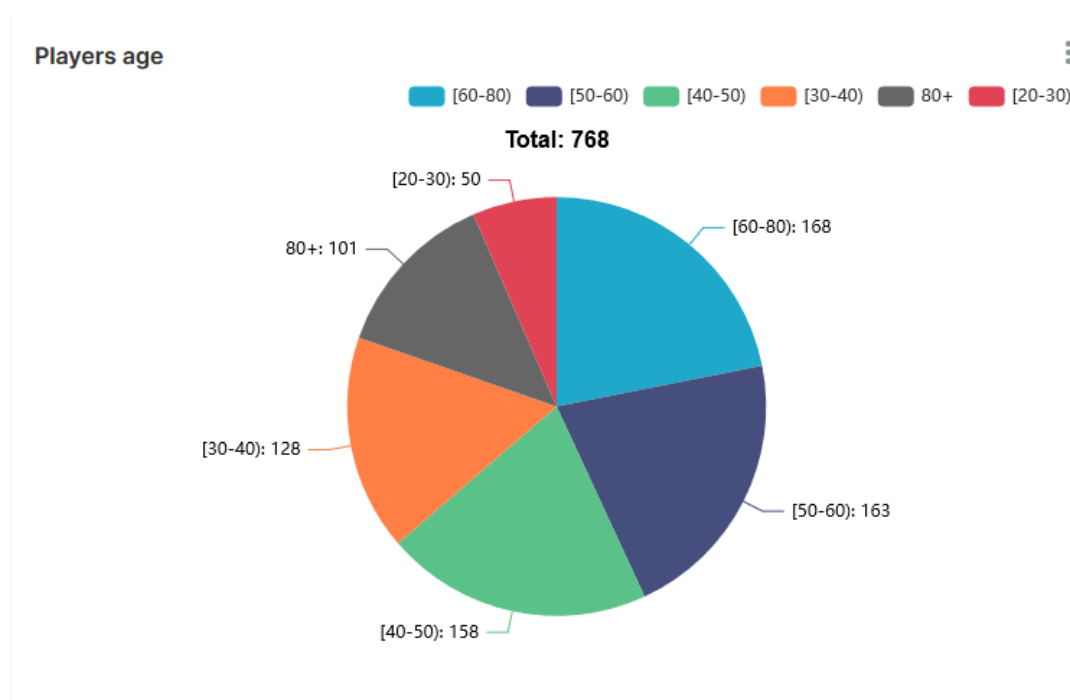


Figura 8.7: Gráfico de setores da distribuição de idade dos jogadores

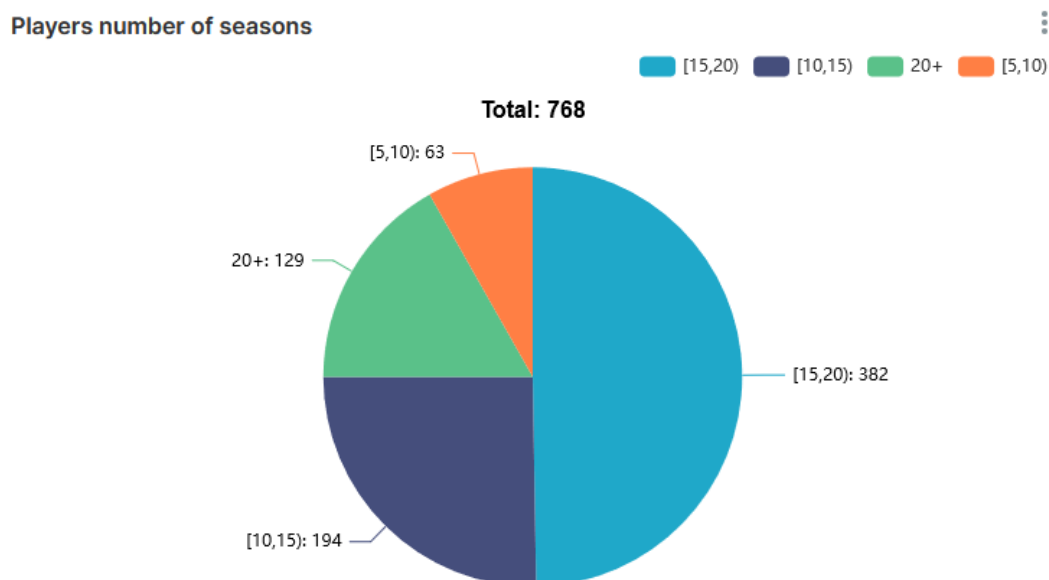


Figura 8.8: Gráfico de setores da distribuição de longevidade da carreira dos jogadores

Posição

Com relação à posição dos jogadores, há 4 grandes divisões: goleiros, zagueiros, meio-campistas e atacantes. No *Transfermarkt*, existem 17 nomenclaturas que diferenciam internamente cada uma das posições. Na Figura 8.9, estão contidas todas as nomenclaturas.

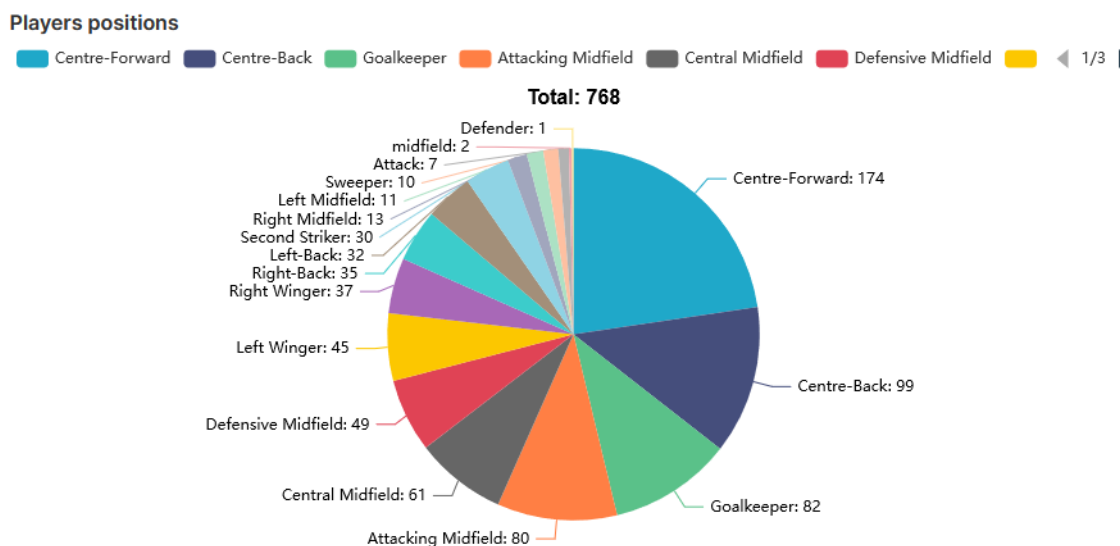


Figura 8.9: Gráfico de setores da distribuição da posição dos jogadores

É possível perceber que a posição mais comum é a de Centro-avante, com 174 (22,66%) jogadores, seguida do zagueiro central, com 99 (12,88%) jogadores, goleiro, com 82 (10,68%), meia-atacante com 80 (10,42%) e assim por diante. Em resumo, 293 (38,15%) são atacantes, 216 (28,12%) são meio-campistas, 177 (23,04%) são zagueiros e 82 (10,68%) são goleiros.

8.2 Pontuações

Antes de exibir as pontuações finais dos jogadores, é importante deixar claro duas informações: os registros são de 25 de outubro de 2023, e qualquer campeonato ou jogo depois dessa data não foi considerado. Além disso, apenas 2 jogadores tiveram seus registros manuais inseridos, Pelé e Zico, então diversos outros futebolistas do século XX podem ter suas informações incompletas, como Maradona, Garrincha e Gerd Muller, por exemplo, que provavelmente entrariam no top 10 caso estivessem completos.

Ressalvas feitas, a seguir são apresentados os resultados para os primeiros 20 jogadores. Primeiramente, no gráfico da Figura 8.10, estão os resultados totais, depois nas figuras 8.11, 8.12 e 8.13 estão os gráficos para cada um dos componentes, Individual, Títulos e Impacto, respectivamente.

8.2 | PONTUAÇÕES

Players' scores - Graph

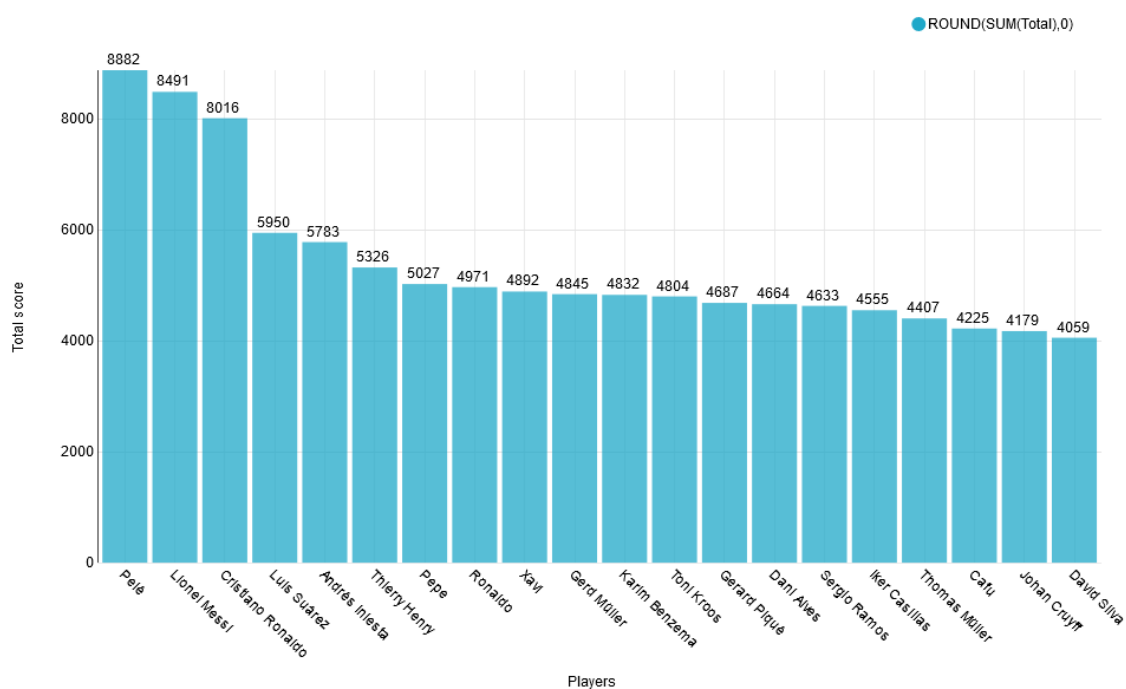


Figura 8.10: Top 20 jogadores em pontuação total.

Player C1 score - Graph

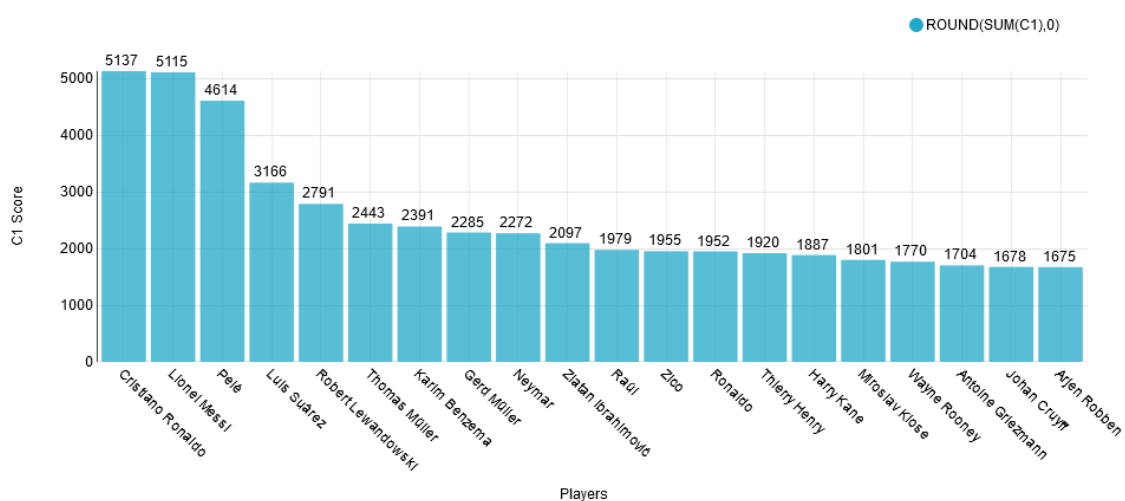


Figura 8.11: Top 20 jogadores em pontuação individual.

8.2 | PONTUAÇÕES

Player C2 score - Graph

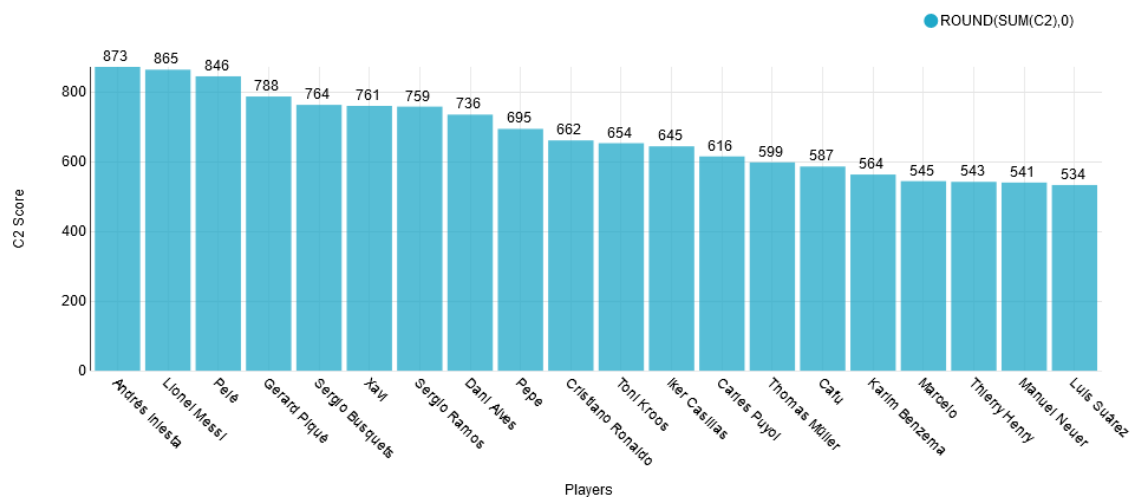


Figura 8.12: Top 20 jogadores em pontuação de títulos.

Player C3 score - Graph

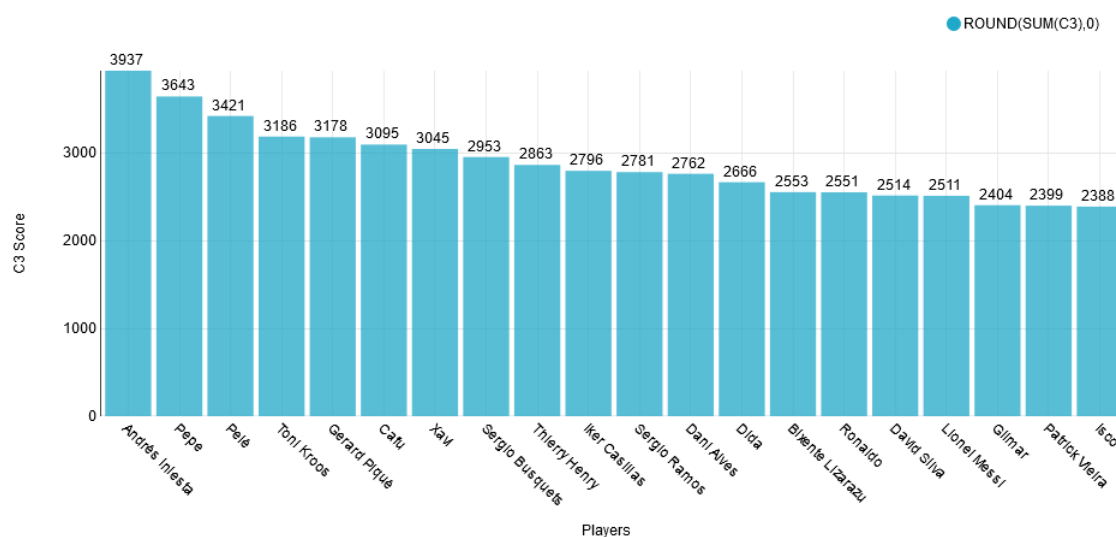


Figura 8.13: Top 20 jogadores em pontuação de impacto.

É possível perceber que o Pelé não lidera nenhum dos componentes, individualmente, mas quando somados, ele é o atleta com a maior pontuação total, a frente de Lionel Messi e Cristiano Ronaldo. Outro fenômeno é a liderança de Andrés Iniesta na componente de títulos e de impacto, por conta principalmente da Copa do Mundo com a Espanha e as 4 *Champions League* com o Barcelona, apesar de estar em 5º no ranqueamento da pontuação total.

Com esses gráficos também é possível perceber que a disputa entre Lionel Messi e Cristiano Ronaldo poderia ter sido diferente se o jogador português tivesse conseguido ganhar mais títulos com sua seleção, já que é no componente de impacto que há a maior diferença entre ele e o argentino. Ganhar uma Copa com Portugal, por exemplo, com certeza seria suficiente para que ocorresse a ultrapassagem, já que trata-se de um título inédito para essa seleção.

Com relação ao futuro, é possível que Lionel Messi supere Pelé, dependendo de quantas temporadas ainda jogar e se conseguir mais títulos com a seleção argentina, pois a diferença entre os dois atualmente é de 400 pontos.

Em [L. B. Salvador, 2023](#), os resultados para os 768 jogadores são exibidos por completo.

8.3 Análise específica por jogador

Para destrinchar a pontuação de um jogador, é possível visualizar todas as suas performances ao longo da carreira, tanto com relação à conquista de títulos como de estatísticas em campeonatos. A Figura 8.14 exibe um trecho da tabela que contém uma linha por campeonato disputado, onde as colunas representam, respectivamente, o jogador, a temporada, o campeonato, o time, e então cada um dos atributos individuais considerados na fórmula. Já na Figura 8.15, cada linha é um torneio conquistado, sendo que as colunas representam, respectivamente, o jogador, a temporada, o título vencido e o time.

Full performance										
name	Season	Championship	Team	Matches	Goals	Assists	PPG	SavedPenalties	CleanSheets	Hattricks
Lionel Messi	2023	Freundschaftsspiele	Argentina	3	5	1	3	0	0	1
Lionel Messi	2023	WM-Qualifikation Südamerika	Argentina	3	3	0	3	0	0	0
Lionel Messi	2023	US Open Cup	Inter Miami CF	1	0	2	3	0	0	0
Lionel Messi	2023	Leagues Cup	Inter Miami CF	7	10	1	3	0	0	0
Lionel Messi	2023	Major League Soccer	Inter Miami CF	6	1	2	1.67	0	0	0
Lionel Messi	2022	UEFA Champions League	Paris Saint-Germain	7	4	4	1.86	0	0	0
Lionel Messi	2022	Coupe de France	Paris Saint-Germain	1	0	0	0	0	0	0
Lionel Messi	2022	Trophée des Champions	Paris Saint-Germain	1	1	0	3	0	0	0
Lionel Messi	2022	Ligue 1	Paris Saint-Germain	32	16	16	2.34	0	0	0
Lionel Messi	2022	WM-Qualifikation Südamerika	Argentina	9	1	0	2.33333	0	0	0
Lionel Messi	2022	CONMEBOL-UEFA Cup of Champions	Argentina	1	0	2	3	0	0	0
Lionel Messi	2022	Weltmeisterschaft 2022	Argentina	7	7	3	2.57143	0	0	0
Lionel Messi	2022	Freundschaftsspiele	Argentina	4	10	1	3	0	0	1
Lionel Messi	2021	Ligue 1	Paris Saint-Germain	26	6	15	2.12	0	0	0
Lionel Messi	2021	UEFA Champions League	Paris Saint-Germain	7	5	0	1.86	0	0	0
Lionel Messi	2021	Coupe de France	Paris Saint-Germain	1	0	0	0	0	0	0
Lionel Messi	2021	WM-Qualifikation Südamerika	Argentina	7	5	0	2.42857	0	0	1
Lionel Messi	2021	Copa América 2021	Argentina	7	4	5	2.71429	0	0	0

Figura 8.14: Trecho da tabela com a performance detalhada de Lionel Messi

8.3 | ANÁLISE ESPECÍFICA POR JOGADOR

Player achievements			
Player	Season	Championship	Team
Lionel Messi	2022	World Cup winner	Argentina
Lionel Messi	2014	World Cup runner-up	Argentina
Lionel Messi	2022	French champion	Paris Saint-Germain
Lionel Messi	2021	French champion	Paris Saint-Germain
Lionel Messi	2022	French Super Cup winner	Paris Saint-Germain
Lionel Messi	2006	Spanish Super Cup winner	FC Barcelona
Lionel Messi	2020	Spanish cup winner	FC Barcelona
Lionel Messi	2013	Spanish Super Cup winner	FC Barcelona
Lionel Messi	2004	Spanish champion	FC Barcelona
Lionel Messi	2012	Spanish champion	FC Barcelona
Lionel Messi	2016	Spanish Super Cup winner	FC Barcelona
Lionel Messi	2016	Spanish cup winner	FC Barcelona
Lionel Messi	2017	Spanish cup winner	FC Barcelona
Lionel Messi	2017	Spanish champion	FC Barcelona
Lionel Messi	2018	Spanish Super Cup winner	FC Barcelona
Lionel Messi	2018	Spanish champion	FC Barcelona

Figura 8.15: Trecho da tabela com os títulos de Lionel Messi

Score by Season					
name	Season	Total	Stats	Achievements	Impact
Lionel Messi	2023	69.11	69.11	0	0
Lionel Messi	2022	1069.39	436	167.01	466.38
Lionel Messi	2021	259.7	205.73	47.48	6.49
Lionel Messi	2020	215.36	156.98	6.19	52.19
Lionel Messi	2019	190.97	190.97	0	0
Lionel Messi	2018	452.48	332.46	28.1	91.91
Lionel Messi	2017	387.87	275.2	23.83	88.84
Lionel Messi	2016	516.33	393.81	37.04	85.49
Lionel Messi	2015	541.9	264.24	76.43	201.23
Lionel Messi	2014	913.71	532	104.52	277.19
Lionel Messi	2013	362.32	282.95	12.66	66.71
Lionel Messi	2012	422.84	354.1	14.84	53.9
Lionel Messi	2011	741.49	491.65	63.39	186.45
Lionel Messi	2010	568.97	316.13	64.44	188.39
Lionel Messi	2009	503.51	268.56	71.57	163.38
Lionel Messi	2008	527.01	219.09	61.32	246.6

Figura 8.16: Trecho da tabela com a pontuação por temporada de Lionel Messi

Com as figuras 8.14 e 8.15, portanto, é possível compreender exatamente cada elemento considerado nos dois primeiros componentes da fórmula: as estatísticas individuais e as conquistas coletivas. A terceira tabela, exposta na Figura 8.16, mostra as pontuações totais e de cada componente para todas as temporadas da carreira do jogador. Assim, pode-se analisar o auge de sua carreira, a melhor temporada individualmente, o ano com mais títulos, e assim por diante. Para o caso de Lionel Messi, por exemplo, a temporada com maior pontuação total foi a de 2022, quando ele ganhou a Copa do Mundo, enquanto a melhor temporada individualmente foi 2014, na qual ele teve 102 participações em gols (gols + assistências). A tabela da Figura 8.14, quando ordenada por número de *hat-tricks*, mostra que no Campeonato Espanhol (*La Liga*) de 2011, o argentino marcou 8 *hat-tricks* e incríveis 50 gols. Cristiano Ronaldo, em 2014, teve marca semelhante, com 8 *hat-tricks* e 48 gols.

8.4 Análise específica por temporada

Por fim, a última análise criada para este trabalho ordena as pontuações de todos os jogadores por temporada, de forma a possibilitar a comparação entre esse resultado e as premiações existentes, como a Bola de Ouro da revista *France Football*, que é a disputa mais prestigiada no mundo futebolístico. Na Figura 8.1, está a comparação dos vencedores da fórmula e da revista francesa no século XXI. O ano exibido é o da premiação, que condecora os jogadores da temporada anterior, logo, o prêmio de 2023 corresponde a temporada 2022-2023 (hemisfério norte). Além disso, ao lado do nome de cada jogador, na coluna da direita, está a posição dele de acordo com a fórmula.

Ano	Vencedor fórmula	Vencedor <i>France Football</i>
2023	Lionel Messi	Lionel Messi (1°)
2022	Jorginho	Karim Benzema (3°)
2021	Riyad Mahrez	Lionel Messi (14°)
2020	Mohamed Salah	-
2019	Kylian Mbappé	Lionel Messi (11°)
2018	Cristiano Ronaldo	Luka Modric (12°)
2017	Cristiano Ronaldo	Cristiano Ronaldo (1°)
2016	Claudio Bravo	Cristiano Ronaldo (2°)
2015	Toni Kroos	Lionel Messi (3°)
2014	Cristiano Ronaldo	Cristiano Ronaldo (1°)
2013	Cesc Fàbregas	Cristiano Ronaldo (23°)
2012	Lionel Messi	Lionel Messi (1°)
2011	David Villa	Lionel Messi (16°)
2010	Éric Abidal	Lionel Messi (3°)
2009	Xavi	Lionel Messi (13°)
2008	Carlos Tevez	Cristiano Ronaldo (2°)
2007	Marco Materazzi	Kaká (20°)
2006	Ronaldinho	Fabio Cannavaro (156°)
2005	Deco	Ronaldinho (24°)
2004	Thierry Henry	Andriy Shevchenko (40°)
2003	Ronaldo	Pavel Nedvěd (67°)
2002	Víctor Aristizábal	Ronaldo (154°)
2001	Bixente Lizarazu	Michael Owen (31°)
2000	Pavel Nedved	Luís Figo (57°)

Tabela 8.1: Tabela comparativa entre o melhor por temporada, de acordo com a fórmula e de acordo com a *France Football*

De acordo com essa comparação, a fórmula e a revista concordaram com 4 escolhas para o melhor do mundo, em 2023, 2017, 2014 e 2012. Além disso, pode-se perceber que, de 2011 até 2023, o resultado da fórmula chegou mais próximo da decisão da revista, já que a média da posição dos vencedores da Bola de Ouro, de acordo com a fórmula, é de 6,7, enquanto de 2000 até 2010 é

51,54.¹

Essa discrepância pode ser resultado de uma alteração, ao longo dos anos, no critério de escolha para o melhor jogador do mundo, ou simplesmente porque, no início do século XXI, não se tinha uma dominância de dois jogadores em relação ao restante, como ocorreu com Lionel Messi e Cristiano Ronaldo, então muito mais atletas tinham chances de ganhar do que posteriormente na era Messi-Cristiano.

Outra questão que pode explicar as diferenças entre a fórmula e a premiação tem a ver com a Copa do Mundo. É sabido que essa competição tem um peso gigantesco nas decisões de Bola de Ouro, porém, como em geral o torneio ocorre no fim da temporada europeia (julho), é mais difícil saber qual edição da premiação levará em conta esse campeonato, se é a do ano em que ela efetivamente ocorreu ou no seguinte. Em 2002, por exemplo, o prêmio foi dado a Ronaldo, que ganhou a Copa desse mesmo ano pela seleção brasileira, porém, se considerarmos apenas a temporada 2001-2002, que não inclui a Copa, ele está posicionado em 154°, de acordo com fórmula. O mesmo pode explicar o ranqueamento de Fabio Cannavaro em 2006, ano em que foi campeão mundial pela Itália e que mesmo assim ficou na 156° posição.

¹ Isso quer dizer que, em média, de 2011 até 2023 o ganhador da Bola de Ouro ocuparia a 7° posição no ranqueamento da fórmula enquanto, de 2000 até 2010 ele ocuparia a 52° posição.

Capítulo 9

Conclusão

As discussões entre apaixonados pelo futebol sempre existirão, principalmente porque a visão de cada um a respeito do esporte é diferente, e aquele que é o melhor para alguém pode não ser para o outro. Mesmo com uma fórmula matemática, que fornece números para embasar argumentos, a maneira como o futebol é enxergado ao longo dos países, épocas e indivíduos nem sempre é uniforme. Por isso, é praticamente impossível chegar em um consenso com relação a alguma opinião a respeito de qualquer assunto desse esporte. Ainda assim, a avaliação sistemática de carreiras de jogadores, implementada neste trabalho, fornece uma grande quantidade de dados para incrementar as análises, e não há porque negar que ela é ao menos uma possibilidade de enriquecer essas discussões sem fim.

Ao longo desta monografia, foram apresentadas descrições e resultados de cada porção e etapa do desenvolvimento do sistema, desde a extração dos dados, no Capítulo 4, até a exibição dos gráficos e tabelas resultantes, no Capítulo 8. Dos três objetivos iniciais, propostos no Capítulo 1, pode-se dizer que todos foram cumpridos por este sistema.

O primeiro objetivo era a criação de algum software capaz de aplicar a fórmula proposta por [E. P. Salvador et al. \(2022\)](#). Ele pôde ser cumprido à medida que, com as etapas de extração (Capítulo 4), armazenamento (Capítulo 3) e cálculo (Capítulo 5), a fórmula foi aplicada integralmente, de maneira automatizada e em massa, para os mais de 700 jogadores, 1000 times e 500 campeonatos registrados no banco de dados.

O segundo objetivo se referia à produção de uma comparação entre os resultados provenientes da fórmula e algum outro método de mensuração da performance dos jogadores. Essa comparação foi apresentada na Seção 8.4, em que há o confronto entre os resultados do prêmio mais cobiçado pelos futebolistas, que é a Bola de Ouro, com as pontuações advindas da fórmula, temporada a temporada, desde o início do século XXI. Nesse mesmo sentido, a implantação do grafo também fornece uma estrutura adicional para incrementar as possibilidades de arguição e comparação envolvendo os dados indexados.

Assim, a métrica descrita na Seção 6.4 corrobora com esse objetivo relacionando uma informação implícita nos registros de carreiras dos jogadores com as pontuações advindas da fórmula.

O terceiro objetivo consistia em possibilitar que pessoas sem grande conhecimento de computação pudessem interagir com os diferentes elementos do sistema, e ele também foi cumprido com o que foi descrito ao longo de todo o Capítulo 7. A ideia é que, para todas as interações possíveis entre usuário e o sistema, existem (ou foram desenvolvidas) interfaces gráficas para auxiliá-lo, desde as inserções no BD operacional, as interações com o grafo e também a exploração dos resultados produzidos pela fórmula.

Com relação a possíveis evoluções deste trabalho, a primeira etapa com certeza seria realizar mais inserções no banco de dados, de forma que seja possível cobrir todos os jogadores atuando na elite do futebol, desde o início do século passado até o presente momento. Além disso, se fossem utilizadas outras fontes além do *Transfermarkt*, seria possível ter mais confiança na qualidade e completude dos dados inseridos, sem que fosse necessária a inserção manual dos registros faltantes. Por fim, para incrementar não o sistema, mas a análise produzida a partir de seus resultados, poderiam ser feitas comparações mais individualizadas entre os jogadores, principalmente com atletas de épocas distintas, a fim de destrinchar as diferenças entre os diversos contextos do esporte ao longo das décadas.

Dessa maneira, conclui-se que o sistema produzido neste trabalho pode servir como base para análises envolvendo o futebol, produzidas por diversos agentes em diferentes áreas do conhecimento, motivados pelos elementos históricos, estatísticos ou puramente futebolísticos inerentes às informações tratadas. Mais que isso, é possível afirmar que, mesmo se não houvesse fórmula e cálculo de pontuações, apenas a estruturação e armazenamento desses dados já poderia ser considerados valiosos para esses agentes.

Referências

- [Dijkstra 1959] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. Em: *Numerische mathematik* 1.1 (1959), pgs. 269–271 (citado nas pgs. 8, 32).
- [Elmasri e Navathe 2010] Ramez Elmasri e Shamkant Navathe. *Fundamentals of Database Systems*. 7ª ed. Addison-Wesley Publishing Company, 2010 (citado nas pgs. 5, 6).
- [FIFA 2023] FIFA. *The Best FIFA Football Awards*. 2023. url: <https://www.fifa.com/fifaplus/en/the-best-fifa-football-awards> (acesso em 20/11/2023) (citado na pg. 1).
- [Football 2023] France Football. *Ballon d’Or*. 2023. url: <https://www.francefootball.fr/ballon-d-or/> (acesso em 20/11/2023) (citado na pg. 1).
- [Garcia-Molina et al. 2008] Hector Garcia-Molina, Jeffrey D. Ullman e Jennifer Widom. *Database Systems: The Complete Book*. 2ª ed. Prentice Hall Press, 2008 (citado na pg. 6).
- [Koenigsberg et al. 2020] Aaron Koenigsberg, Jarred Pilgrim e Joseph Baker. “Generational differences in the ranking pathways of top 100 ranked golfers”. Em: *J Sports Sci* (2020) (citado na pg. 1).
- [Sadalage e Fowler 2012] Pramod J. Sadalage e Martin Fowler. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional, 2012 (citado na pg. 6).
- [E. P. Salvador et al. 2022] Emanuel Pericles Salvador, Denilson de Menezes Santos e Sonny Allan Silva Bezerra. “Modk: formula for determining the best season and career of a football player by objective indicators”. Em: *The Open Sports Sciences Journal* (2022) (citado nas pgs. 1, 2, 21, 75).
- [L. B. Salvador 2023] Lorenzo Bertin Salvador. *Repositório público do sistema ModoK*. 2023. url: <https://gitlab.com/lorenzobs/sistema-modok-publico> (acesso em 25/11/2023) (citado nas pgs. 1, 2, 16, 55, 71).

REFERÊNCIAS

- [Tarjan 1971] Robert Tarjan. “Depth-first search and linear graph algorithms”. Em: *12th Annual Symposium on Switching and Automata Theory (swat 1971)*. 1971, pgs. 114–121. doi: [10.1109/SWAT.1971.10](https://doi.org/10.1109/SWAT.1971.10) (citado na pg. 8).
- [Xia *et al.* 2018] Vincent Xia, Kavirath Jain, Akshay Krishna e Christopher G. Brinton. “A network-driven methodology for sports ranking and prediction”. Em: *Institute of Electrical and Electronics Engineers Inc* (2018) (citado na pg. 1).