

Trabalho Prático #1

NBA

Roberto Rolão de Castro - 107133
Tiago Caridade Gomes - 108307
Sara Figueiredo Almeida - 108796
Joaquim Vertentes Rosa - 109089

Web Semântica - 2024/2025
Mestrado em Engenharia Informática



Conteúdo

1	Introdução ao Tema	4
1.1	Arquitetura da Aplicação e Tecnologias Utilizadas	4
2	Dados, suas fontes e sua transformação	6
3	Operações sobre os dados (SPARQL)	8
4	Funcionalidades da Aplicação (UI)	22
5	Funcionalidades de Administrador (Autenticação e Operações CRUD)	31
6	Conclusões	34
7	Configurações para executar a app	35
7.1	Execução Manual	35
7.2	Execução com Docker	36
7.3	Ficheiros Importantes	37

Lista de Figuras

2.1	Modelo de dados	7
2.2	Entidades e as suas relações	7
4.1	Página Inicial com informações gerais e notícias	22
4.2	Continuação da primeira página	22
4.3	Página da listagem dos jogadores com secção de filtros	22
4.4	Continuação da página de listagem dos jogadores	22
4.5	Página onde se podem ver detalhes de um jogador em específico	23
4.6	Vizualização da carreira do jogador	23
4.7	Grafo com as conexões ao longo da carreira do jogador	24
4.8	Colegas de carreira do jogador	24
4.9	Página de informação sobre as equipas	25
4.10	Detalhes da equipa	25
4.11	Jogadores que jogaram na equipa selecionada	25
4.12	Página com a listagem das diferentes temporadas	26
4.13	Mostra detalhes da temporada, as equipas da mesma e os seus jogadores	26
4.14	Página com a listagem das arenas da NBA	27
4.15	Vizualizador dos detalhes da arena selecionada	27
4.16	Mapa das arenas	28
4.17	Página onde se pode comparar detalhes técnicos dos jogadores selecionados	28
4.18	Vizualização das estatísticas	29
4.19	Vizualização das estatísticas	29
4.20	Vizualização das estatísticas	29
4.21	Grafo de conexos dos jogadores de uma dada temporada	29
4.22	Grafo de conexos dos jogadores de uma dada temporada - Detalhes	29
4.23	Quiz - Inserir Nome e Pontuações	30
4.24	Quiz	30
5.1	Formulário de login protegido	31
5.2	Modal de criação de jogador	32
5.3	Modal de edição de jogador	32
5.4	Botões de admin visíveis após login	33

1. Introdução ao Tema

O presente trabalho prático insere-se no âmbito da unidade curricular de Web Semântica e tem como principal objetivo o desenvolvimento de um sistema de informação baseado na web, recorrendo a tecnologias como RDF, SPARQL, GraphDB e Django. O sistema centra-se no domínio da NBA (National Basketball Association), uma das ligas desportivas mais conhecidas a nível mundial, caracterizada por uma estrutura rica em entidades e relações: jogadores, equipas, arenas, temporadas e estatísticas.

A escolha da NBA como tema deve-se à sua complexidade estrutural e à diversidade de dados, o que proporciona um excelente caso de estudo para a modelação semântica e exploração interativa de informação. Os dados utilizados foram obtidos através de uma API da NBA cedida por um docente da Universidade. Este dataset teve origem num dataset do Kaggle, mas sofreu algumas modificações e alterações para garantir a qualidade e quantidade suficiente dos dados. Para podermos utilizar os dados da API, foram desenvolvidos scripts personalizados que permitiram extrair os dados em formato JSON, transformá-los em ficheiros CSV e, posteriormente, converter essa informação para o formato RDF, adequado à sua inserção no triplestore GraphDB.

O sistema resultante permite ao utilizador explorar e cruzar diferentes dimensões da informação da NBA, através de uma interface web desenvolvida em Django. A aplicação inclui funcionalidades como a navegação por temporadas, equipas e jogadores, a visualização de redes de relações entre atletas, mapas de arenas e um dashboard de estatísticas, entre outros. Este projeto visa assim demonstrar o potencial da Web Semântica na construção de aplicações informativas, interativas e interligadas.

1.1 Arquitetura da Aplicação e Tecnologias Utilizadas

A aplicação foi desenvolvida com base no framework **Django**, seguindo o padrão de arquitetura nativo **MVT (Model-View-Template)**. As *views* tratam os pedidos dos utilizadores e executam queries SPARQL ao triplestore **GraphDB**, enquanto os templates HTML são usados para apresentar os dados de forma dinâmica e interativa. Foi também definido o *model QuizScore*, responsável por armazenar os dados de cada sessão do quiz desenvolvido, permitindo construir um *leaderboard* com as melhores pontuações, visível diretamente na interface do utilizador.

A estrutura do sistema é composta por múltiplos *endpoints* definidos no `urls.py`, que encaminham os pedidos para funções específicas no `views.py`. Estas funções constroem e executam queries SPARQL, processam os resultados e renderizam os templates com os dados.

Tecnologias e Bibliotecas Utilizadas

- **Django** – framework principal para o desenvolvimento do backend e gestão de templates;
- **GraphDB** – base de dados semântica (triplestore) onde os dados RDF da NBA foram carregados e são consultados via SPARQL;
- **SPARQLWrapper** – biblioteca Python usada para construir e executar queries SPARQL a partir das views;
- **Bootstrap** – framework CSS utilizada para o design responsivo e moderno da interface;
- **Chart.js** – biblioteca de visualização de dados usada no dashboard estatístico para gerar gráficos interativos;

- **Leaflet.js** – utilizada para apresentar as arenas da NBA num mapa interativo;
- **vis-network** – usada para representar graficamente a rede de ligações entre jogadores (grafo interativo);
- **AOS (Animate on Scroll)** – adiciona animações visuais suaves ao scroll da página;
- **JavaScript + Fetch API** – utilizado em várias funcionalidades interativas (quiz, filtros, leaderboard, CRUD de jogadores);
- **Docker** – a aplicação foi conteinerizada para facilitar a sua execução e portabilidade;
- **GraphDB Workbench** – interface gráfica usada para explorar e validar os dados RDF carregados.

Esta combinação de tecnologias permitiu criar uma aplicação rica e informativa, baseada em dados semânticos, com uma interface moderna e funcionalidades interativas.

2. Dados, suas fontes e sua transformação

Como referido anteriormente, a informação utilizada para desenvolver o projeto no seu integral foi obtida a partir de uma API cedida por um docente da Universidade, tendo tido a sua origem no Kaggle mas sofrendo algumas modificações e adições de dados.

- `getData.py` – responsável por consumir os dados da API e gerar os respetivos ficheiros CSV;
- `getStatics.py` – script complementar que extrai estatísticas e dados adicionais;
- `rdfConverter.py` – encarregado da conversão dos dados CSV para triplos RDF, utilizando o módulo `simplegraph.py` e exportando os resultados no formato `Notation3 (.n3)`;
- `simplegraph.py` – biblioteca auxiliar, cedida numa das aulas práticas, usada para facilitar a criação, manipulação e exportação de triplos RDF.

As principais entidades representadas nos dados incluem:

- **nba:Player**: nome, data de nascimento, altura, peso, posição, escola, ano de draft, nacionalidade, fotografia;
- **nba:Team**: nome atual, nomes alternativos, acrónimos, logótipos, cidade, conferência, divisão, arena;
- **nba:Season**: ano ou etiqueta identificadora da época;
- **nba:Participation**: ligação entre jogador, equipa, temporada e tipo de época (Regular/-Playoffs);
- **nba:Arena**: nome, localização, capacidade, ano de abertura, coordenadas geográficas, equipa da casa.

A conversão para RDF seguiu uma estrutura semântica coerente, com base na especificação de prefixos próprios (<http://example.org/nba/>), o que permitiu representar os dados como um **grafo de conhecimento interligado**. O ficheiro resultante em formato `.n3` foi carregado no triplestore **GraphDB**, onde passou a ser consultável através de consultas SPARQL. A estrutura **N3** foi escolhida por ser mais legível e menos redundante.

Assim, este processo permitiu transformar dados originalmente não estruturados num modelo semântico explorável, viabilizando uma navegação rica e contextualizada através da aplicação web desenvolvida.

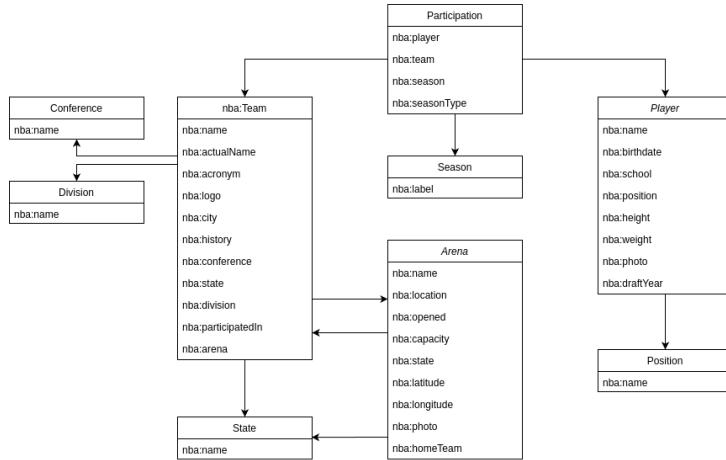


Figura 2.1: Modelo de dados

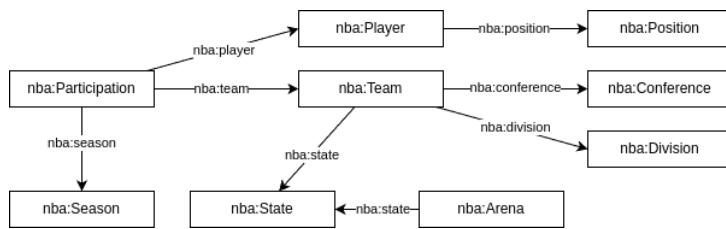


Figura 2.2: Entidades e as suas relações

3. Operações sobre os dados (SPARQL)

A aplicação web recorre a múltiplos endpoints para obter, consultar e explorar os dados RDF. Cada um desses endpoints chama uma função em `views.py`, que por sua vez executa uma ou mais consultas SPARQL sobre o repositório GraphDB. Nesta secção apresenta-se o mapeamento entre os principais URLs da aplicação, as funções Python responsáveis e as respetivas queries SPARQL.

Na totalidade do projeto implementámos queries **SELECT** para a maioria das funcionalidades da aplicação, à exceção de queries **ASK** para validar as respostas do quiz e queries **INSERT**, **UPDATE** e **DELETE** que tornam possível adicionar, atualizar os dados e eliminar um jogador caso o user seja *admin*.

De seguida são detalhadas todas as queries construídas ao longo do projeto:

Endpoints e Consultas

/ – Página inicial

- **Função:** `home_page(request)`
- **Descrição:** Retorna estatísticas globais e participações por temporada para a página inicial.
- **SPARQL (estatísticas globais):**

```
PREFIX nba: <http://example.org/nba/>
SELECT (COUNT(DISTINCT ?player) AS ?players)
       (COUNT(DISTINCT ?team) AS ?teams)
       (COUNT(DISTINCT ?season) AS ?seasons)
       (COUNT(?p) AS ?participations)
WHERE {
  ?p nba:player ?player ;
    nba:team ?team ;
    nba:season ?season .
}
```

- **SPARQL (participações por temporada):**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?season (COUNT(?p) AS ?count) WHERE {
  ?p nba:season ?season .
}
GROUP BY ?season
ORDER BY DESC(?count)
LIMIT 5
```

/jogadores/filter/ – Filtro de jogadores

- **Função:** `filter_players(request)`
- **Descrição:** Filtra jogadores com base em critérios como nome, posição, equipa, nacionalidade, etc.

- SPARQL:

```
PREFIX nba: <http://example.org/nba/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT DISTINCT ?id ?nome ?position ?height ?weight ?draftYear ?birthdate ?bornIn ?school
WHERE {
    ?id rdf:type nba:Player ;
        nba:name ?nome .

    OPTIONAL { ?id nba:position ?posObj .
               ?posObj nba:name ?position . }
    OPTIONAL { ?id nba:height ?height . }
    OPTIONAL { ?id nba:weight ?weight . }
    OPTIONAL { ?id nba:draftYear ?draftYear . }
    OPTIONAL { ?id nba:birthdate ?birthdate . }
    OPTIONAL { ?id nba:bornIn ?bornIn . }
    OPTIONAL { ?id nba:school ?school . }
    OPTIONAL { ?id nba:photo ?photo . }

    OPTIONAL {
        ?participation nba:player ?id ;
            nba:team ?teamId ;
            nba:season nba:season_2022 .
        ?teamId nba:actualName ?teamName .
    }
}
ORDER BY ?nome
```

/jogadores/countries/ – Lista de nacionalidades

- **Função:** get_player_countries(request)
- **Descrição:** Retorna uma lista com todos os países de origem disponíveis.
- SPARQL:

```
PREFIX nba: <http://example.org/nba/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?bornIn WHERE {
    ?player rdf:type nba:Player ;
        nba:bornIn ?bornIn .
    FILTER(?bornIn != "")
}
ORDER BY ?bornIn
```

/jogadores/schools/ – Lista de escolas

- **Função:** get_player_schools(request)
- **Descrição:** Retorna uma lista com todas as escolas disponíveis.
- SPARQL:

```
PREFIX nba: <http://example.org/nba/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?school WHERE {
    ?player rdf:type nba:Player ;
        nba:school ?school .
    FILTER(?school != "")
}
ORDER BY ?school
```

/equipas/ – Lista de todas as equipas

- **Função:** get_all_teams()
- **Descrição:** Retorna uma lista de todas as equipas com dados visuais e históricos.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?team ?actualName ?name ?acronym ?logo WHERE {
    ?team a nba:Team ;
        nba:name ?name .
    OPTIONAL { ?team nba:actualName ?actualName . }
    OPTIONAL { ?team nba:acronym ?acronym . }
    OPTIONAL { ?team nba:logo ?logo . }
}
ORDER BY ?name
```

/temporadas/ – Lista de todas as temporadas

- **Função:** list_temporadas(request)
- **Descrição:** Lista todas as temporadas únicas presentes no grafo RDF.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?season WHERE {
    ?p nba:season ?season .
}
ORDER BY ?season
```

/participacoes/ – Lista de todas as participações

- **Função:** list_participacoes(request)
- **Descrição:** Retorna todas as participações, incluindo jogador, equipa, temporada e tipo de época.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?player ?team ?season ?seasonType WHERE {
    ?p nba:player ?player ;
        nba:team ?team ;
        nba:season ?season ;
```

```
    nba:seasonType ?seasonType .  
}  
ORDER BY ?season
```

/equipa/<id>/ – Página de equipa

- **Função:** pagina_equipa(request, id)
- **Descrição:** Retorna as informações de uma equipa e as participações dos jogadores por temporada.
- **SPARQL (info da equipa):**

```
PREFIX nba: <http://example.org/nba/>
```

```
SELECT ?name ?acronym ?logo ?city ?statename ?conference ?conferencename ?divison ?divi  
      ?team a nba:Team ;  
            nba:name ?name .  
      OPTIONAL { ?team nba:acronym ?acronym . }  
      OPTIONAL { ?team nba:logo ?logo . }  
      OPTIONAL { ?team nba:city ?city . }  
      OPTIONAL { ?team nba:state ?state . }  
      OPTIONAL { ?state nba:name ?statename . }  
      OPTIONAL { ?team nba:conference ?conference . }  
      OPTIONAL { ?conference nba:name ?conferencename . }  
      OPTIONAL { ?team nba:division ?division . }  
      OPTIONAL { ?division nba:name ?divisionname . }  
      OPTIONAL { ?team nba:arena ?arena . }  
      OPTIONAL { ?arena nba:name ?arenaname . }  
      FILTER(STR(?team) = "{team_uri}")  
}
```

- **SPARQL (jogadores por temporada):**

```
PREFIX nba: <http://example.org/nba/>
```

```
SELECT DISTINCT ?player ?playerName ?playerPhoto ?season ?seasonType WHERE { █  
      ?p nba:team ?team ;  
            nba:player ?player ;  
            nba:season ?season ;  
            nba:seasonType ?seasonType .  
      ?player nba:name ?playerName .  
      FILTER(STR(?team) = "{team_uri}")  
}  
ORDER BY ?season
```

/jogador/<id>/ – Página de jogador (carreira completa)

- **Função:** pagina_jogador(request, id)
- **Descrição:** Retorna os dados biográficos do jogador e a sua equipa atual.
- **SPARQL (dados do jogador):**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?name ?birthdate ?bornIn ?draftYear ?position ?height ?weight ?school ?photo WHERE {
    ?player a nba:Player ;
        nba:name ?name .
    FILTER(STR(?player) = "{jogador_uri}")
    OPTIONAL { ?player nba:position ?posObj.
        ?posObj nba:name ?position. }
    OPTIONAL { ?player nba:birthdate ?birthdate. }
    OPTIONAL { ?player nba:bornIn ?bornIn. }
    OPTIONAL { ?player nba:draftYear ?draftYear. }
    OPTIONAL { ?player nba:height ?height. }
    OPTIONAL { ?player nba:weight ?weight. }
    OPTIONAL { ?player nba:school ?school. }
    OPTIONAL { ?player nba:photo ?photo. }
}
```

- SPARQL (última equipa):

```
PREFIX nba: <http://example.org/nba/>
SELECT ?team ?teamName ?teamLogo ?season ?seasonType WHERE {
    ?p nba:player ?player ;
        nba:team ?team ;
        nba:season ?season ;
        nba:seasonType ?seasonType .
    ?team nba:actualName ?teamName ;
        nba:logo ?teamLogo .
    FILTER(STR(?player) = "{jogador_uri}")
}
ORDER BY DESC(?season)
LIMIT 1
```

/jogador/<id>/timeline/ – Linha do tempo da carreira

- Função: timeline_jogador(request, id)
- Descrição: Retorna a linha do tempo da carreira de um jogador.
- SPARQL:

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?season ?seasonLabel ?team ?teamName ?teamLogo ?seasonType WHERE {
    ?p nba:player ?player ;
        nba:season ?season ;
        nba:team ?team ;
        nba:seasonType ?seasonType .
    ?team nba:name ?teamName .
    OPTIONAL { ?team nba:logo ?teamLogo . }
    OPTIONAL { ?season nba:label ?seasonLabel . }
    FILTER(STR(?player) = "{player_uri}")
}
ORDER BY ?season
```

/temporada/<ano>/ – Página de temporada

- **Função:** pagina_temporada(request, ano)
- **Descrição:** Retorna todas as equipas e jogadores de uma temporada específica.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?team ?teamName ?player ?playerName ?seasonType WHERE {
    ?p nba:season ?season ;
        nba:team ?team ;
        nba:player ?player ;
        nba:seasonType ?seasonType .
    ?team nba:name ?teamName .
    ?player nba:name ?playerName .
    FILTER(STR(?season) = "{season_uri}")
}
ORDER BY ?team ?player
```

/arenas/ – Lista de todas as arenas

- **Função:** list_arenas(request)
- **Descrição:** Retorna uma lista com todas as arenas e suas informações.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?arena ?name ?photo ?location ?opened ?capacity ?homeTeam ?homeTeamName
?arena a nba:Arena ;
       nba:name ?name .
OPTIONAL { ?arena nba:photo ?photo . }
OPTIONAL { ?arena nba:location ?location . }
OPTIONAL { ?arena nba:capacity ?capacity . }
OPTIONAL {
    ?arena nba:homeTeam ?homeTeam .
    ?homeTeam nba:actualName ?homeTeamName .
}
}
ORDER BY ?name
```

/arena/<id>/ – Página de Arena com detalhes

- **Função:** pagina_arena(request, id)
- **Descrição:** Retorna os dados completos de uma arena: nome, localização, capacidade, etc.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?name ?location ?opened ?capacity ?latitude ?longitude ?photo ?homeTeam ?homeTeamName
<{arena_uri}> a nba:Arena ;
       nba:name ?name ;
```

```
        nba:location ?location ;
        nba:homeTeam ?homeTeam .
OPTIONAL { <{arena_uri}> nba:opened ?opened. }
OPTIONAL { <{arena_uri}> nba:capacity ?capacity. }
OPTIONAL { <{arena_uri}> nba:latitude ?latitude. }
OPTIONAL { <{arena_uri}> nba:longitude ?longitude. }
OPTIONAL { <{arena_uri}> nba:photo ?photo. }
OPTIONAL { ?homeTeam nba:actualName ?homeTeamName. }
}
```

/mapa/arenas/ – Mapa interativo de arenas

- **Função:** mapa_arenas(request)
- **Descrição:** Retorna dados de coordenadas das arenas para visualização em mapa.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?arena ?name ?latitude ?longitude ?photo WHERE {
    ?arena a nba:Arena ;
        nba:name ?name ;
        nba:latitude ?latitude ;
        nba:longitude ?longitude .
OPTIONAL { ?arena nba:photo ?photo . }
}
```

/comparar/ – Comparador de Jogadores

- **Função:** comparar_jogadores(request)
- **Descrição:** Compara dois jogadores lado a lado.
- **SPARQL (dados do jogador):**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?name ?birthdate ?bornIn ?draftYear ?position ?positionName ?height ?weight ?school
?player a nba:Player ;
    nba:name ?name ;
    nba:birthdate ?birthdate ;
    nba:bornIn ?bornIn ;
    nba:draftYear ?draftYear ;
    nba:position ?position ;
    nba:height ?height ;
    nba:weight ?weight ;
    nba:school ?school ;
    nba:photo ?photo .
?position nba:name ?positionName .
FILTER(STR(?player) = "{jogador_uri}")
}
```

- **SPARQL (participações):**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?team ?teamName ?season ?seasonType WHERE {
    ?p nba:player ?player ;
        nba:team ?team ;
        nba:season ?season ;
        nba:seasonType ?seasonType .
    ?team nba:name ?teamName .
    FILTER(STR(?player) = "{jogador_uri}")
}
ORDER BY ?season
```

/grafo/jogador/<id>/ – Grafo de relacionamentos do jogador

- **Função:** grafo_jogador(request, id)
- **Descrição:** Retorna um grafo com as equipas onde o jogador atuou e seus companheiros de equipa.
- **SPARQL (nome do jogador):**

```
PREFIX nba: <http://example.org/nba/>

SELECT ?name WHERE {
    <{player_uri}> nba:name ?name .
}
```

- **SPARQL (equipas e temporadas):**

```
PREFIX nba: <http://example.org/nba/>

SELECT DISTINCT ?team ?teamName ?teamPhoto ?season ?seasonLabel WHERE {
    ?p nba:player <{player_uri}> ;
        nba:team ?team ;
        nba:season ?season .
    ?team nba:actualName ?teamName ;
        nba:logo ?teamPhoto .
    OPTIONAL { ?season nba:label ?seasonLabel . }
}
ORDER BY ?season
```

- **SPARQL (companheiros de equipa):**

```
PREFIX nba: <http://example.org/nba/>

SELECT DISTINCT ?teammate ?teammateName ?teammatePhoto ?team ?teamName ?season WHERE {
    ?p1 nba:player <{player_uri}> ;
        nba:team ?team ;
        nba:season ?season .

    ?p2 nba:player ?teammate ;
        nba:team ?team ;
        nba:season ?season .
```

```
?teammate nba:name ?teammateName .  
OPTIONAL { ?teammate nba:photo ?teammatePhoto . }  
?team nba:actualName ?teamName .  
  
FILTER(?teammate != <{player_uri}>)  
}  
ORDER BY ?season ?team ?teammateName
```

/jogador/<id>/companheiros/ – Companheiros de equipa do jogador

- **Função:** companheiros_jogador(request, id)
- **Descrição:** Retorna todos os companheiros de equipa do jogador agrupados por temporada e equipa.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>  
  
SELECT DISTINCT ?season ?seasonLabel ?team ?teamName ?teamLogo ?companion ?companionName  
    ?p1 nba:player <{player_uri}> ;  
        nba:team ?team ;  
        nba:season ?season ;  
        nba:seasonType ?seasonType .  
  
    ?p2 nba:player ?companion ;  
        nba:team ?team ;  
        nba:season ?season ;  
        nba:seasonType ?seasonType .  
  
    ?team nba:name ?teamName .  
    OPTIONAL { ?team nba:logo ?teamLogo . }  
  
    ?companion nba:name ?companionName .  
    OPTIONAL { ?companion nba:photo ?companionPhoto . }  
  
    OPTIONAL { ?season nba:label ?seasonLabel . }  
  
    FILTER(?companion != <{player_uri}>)  
}  
ORDER BY DESC(?season) ?teamName ?companionName
```

/rede/jogadores/ – Rede de Conexões entre Jogadores

- **Função:** rede_jogadores(request)
- **Descrição:** Gera um grafo com a rede de jogadores que jogaram juntos numa temporada.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>  
SELECT DISTINCT ?player ?playerName ?team ?season WHERE {  
    ?p a nba:Participation ;  
        nba:player ?player ;
```

```
nba:team ?team ;
nba:season ?season .
?player nba:name ?playerName .
FILTER(?season = <{full_season_uri}>)
}
LIMIT 50
```

/rede/jogadores/expand/<player_id>/ – Expansão de jogador na rede

- **Função:** expandir_jogador(request, player_id)
- **Descrição:** Expande a rede de conexões para um jogador específico.
- **SPARQL:**

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?player ?playerName ?team ?season WHERE {
    ?p a nba:Participation ;
        nba:player ?player ;
        nba:team ?team ;
        nba:season ?season .
    ?player nba:name ?playerName .
    FILTER EXISTS {
        ?p2 nba:player <{player_id}> ;
            nba:team ?team ;
            nba:season ?season .
    }
}
```

/stats/ – Dashboard Estatístico

- **Função:** stats(request)
- **Descrição:** Retorna dados agregados para visualização de gráficos.
- **SPARQL (participações por temporada):**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?season (COUNT(?player) AS ?total) WHERE {
    ?p nba:season ?season ;
        nba:player ?player .
} GROUP BY ?season ORDER BY ?season
```

- **SPARQL (jogadores por equipa):**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?team_id ?team_name (COUNT(DISTINCT ?player) AS ?total) WHERE {
    ?p nba:team ?team ;
        nba:player ?player .
    ?team nba:name ?team_name .
    BIND(STRAFTER(STR(?team), "_") AS ?team_id)
} GROUP BY ?team_id ?team_name
```

- **SPARQL (jogadores com mais temporadas):**

```
PREFIX nba: <http://example.org/nba/>
SELECT ?player ?name (COUNT(DISTINCT ?season) AS ?total) WHERE {
    ?p nba:player ?player ;
        nba:season ?season .
    ?player nba:name ?name .
} GROUP BY ?player ?name ORDER BY DESC(?total) LIMIT 10
```

- SPARQL (distribuição por posições):

```
PREFIX nba: <http://example.org/nba/>
SELECT ?position (COUNT(?player) AS ?total) WHERE {
    ?player a nba:Player ;
        nba:position ?position .
} GROUP BY ?position
```

- SPARQL (altura por posição):

```
PREFIX nba: <http://example.org/nba/>
SELECT ?position ?height WHERE {
    ?player a nba:Player ;
        nba:position ?position ;
        nba:height ?height .
}
```

- SPARQL (peso por posição):

```
PREFIX nba: <http://example.org/nba/>
SELECT ?position ?weight WHERE {
    ?player a nba:Player ;
        nba:position ?position ;
        nba:weight ?weight .
}
```

- SPARQL (jogadores por ano de nascimento):

```
PREFIX nba: <http://example.org/nba/>
SELECT ?birthdate WHERE {
    ?player a nba:Player ;
        nba:birthdate ?birthdate .
}
```

/game/ – Página do Quiz

- **Função:** quiz_page(request)
- **Descrição:** Renderiza o template `quiz.html`, incluindo o formulário de início, perguntas e placar de melhores pontuações.

/quiz/ – Gerador de Perguntas

- **Função:** quiz_questions(request)
- **Descrição:** Gera 2 tipos de perguntas aleatórias sobre jogadores e arenas com múltiplas escolhas.

- SPARQL (Player-Team-Season):

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?player ?playerName ?team ?teamName ?season ?seasonLabel WHERE {
    ?p nba:player ?player ;
        nba:team ?team ;
        nba:season ?season .
    ?player nba:name ?playerName .
    ?team nba:actualName ?teamName .
    ?season nba:label ?seasonLabel .
} LIMIT 300
```

- SPARQL (Arena-HomeTeam):

```
PREFIX nba: <http://example.org/nba/>
SELECT DISTINCT ?arenaName ?teamName WHERE {
    ?arena a nba:Arena ;
        nba:name ?arenaName ;
        nba:homeTeam ?team .
    ?team nba:actualName ?teamName .
} LIMIT 100
```

/quiz/check_answer/ – Verificar Resposta com ASK

- **Função:** check_answer(request)
- **Descrição:** Verifica se a resposta selecionada está correta utilizando uma query ASK.
- SPARQL (exemplo para Player-Team-Season):

```
PREFIX nba: <http://example.org/nba/>
ASK {{?
    player nba:name "{player_name}" .
    team nba:actualName "{selected}" .
    season nba:label "{season}" .
    ?p nba:player ?player ;
        nba:team ?team ;
        nba:season ?season .
}}
```

/quiz/check_correct_answer/ – Resposta Correta (fallback)

- **Função:** check_correct_answer(request)
- **Descrição:** Percorre as opções de resposta e retorna aquela que satisfaz a query ASK.
- SPARQL (exemplo para Arena-HomeTeam):

```
PREFIX nba: <http://example.org/nba/>
ASK {{?
    arena nba:name "{arena_name}" ;
```

```
nba:homeTeam ?team .  
?team nba:actualName "{selected}" .  
}}
```

/quiz/save_score/ – Armazenar Pontuação

- **Função:** submit_score(request)
- **Descrição:** Regista a pontuação de um jogador no modelo QuizScore após concluir o quiz.

/staff/jogadores/adicionar/ – Adicionar jogador

- **Função:** add_player(request)
- **Descrição:** Adiciona um novo jogador ao repositório usando SPARQL UPDATE.
- **SPARQL (INSERT):**

```
PREFIX nba: http://example.org/nba/  
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#  
INSERT DATA {  
  nba:{player_id} rdf:nba ;  
  nba "{data.get('name')}" .  
  # Optional properties added based on available data  
  # nba:{player_id} nba "{birthdate}" .  
  # nba:{player_id} nba "{data.get("bornIn")}" .  
  # nba:{player_id} nba {position_uri} .  
  # nba:{player_id} nba "{data.get("height")}" .  
  # nba:{player_id} nba "{data.get("weight")}" .  
  # nba:{player_id} nba "{data.get("draftYear")}" .  
  # nba:{player_id} nba "{data.get("school")}" .  
  # nba:{player_id} nba <{data.get("photo")}> .  
}
```

/staff/delete_player/<player_id>/ – Apagar jogador

- **Função:** delete_player(request, player_id)
- **Descrição:** Remove um jogador do repositório e todas as suas referências.
- **SPARQL (DELETE):**

```
PREFIX nba: http://example.org/nba/  
DELETE WHERE {  
  ?s ?p {player_uri} .  
};  
DELETE WHERE {  
  {player_uri} ?p ?o .  
}
```

/staff/jogadores/update/ – Atualizar jogador

- **Função:** update_player(request)
- **Descrição:** Atualiza um jogador existente usando SPARQL UPDATE.
- **SPARQL (DELETE parte 1):**

```
PREFIX nba: http://example.org/nba/
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
DELETE {
  {player_uri} ?p ?o .
}
WHERE {
  {player_uri} ?p ?o .
  FILTER(?p != rdf)
}
```

- **SPARQL (INSERT parte 2):**

```
PREFIX nba: http://example.org/nba/
PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
INSERT DATA {
  {player_uri} nba "{data.get('name')}" .
  # Optional properties added based on available data
  # {player_uri} nba "{birthdate}" .
  # {player_uri} nba "{bornIn}" .
  # {player_uri} nba {position_uri} .
  # {player_uri} nba "{data.get("height")}" .
  # {player_uri} nba "{data.get("weight")}" .
  # {player_uri} nba "{data.get("draftYear")}" .
  # {player_uri} nba "{data.get("school")}" .
  # {player_uri} nba <{data.get("photo")}> .
}
```

As consultas SPARQL apresentadas nesta secção são fundamentais para a extração e exploração da informação presente no grafo RDF. Através da sua integração com o backend em Django, estas operações possibilitam uma navegação semântica eficiente e dinâmica, permitindo ao utilizador aceder a dados complexos de forma estruturada e intuitiva. O uso combinado de endpoints bem definidos e queries SPARQL otimizadas garante a consistência e a expressividade das funcionalidades oferecidas pela aplicação.

4. Funcionalidades da Aplicação (UI)

A aplicação web desenvolvida oferece uma interface dinâmica e intuitiva que permite aos utilizadores explorar de forma interativa os dados da NBA disponíveis no dataset. As funcionalidades cobrem desde a simples listagem de entidades e dos seus atributos até visualizações gráficas das relações entre as mesmas e mapas interativos.

Funcionalidades Implementadas

- Página Inicial:** Apresenta indicadores globais (número total de jogadores, equipas, temporadas, participações), temporadas com mais atividade e integração com uma API externa de notícias sobre a liga americana de basquetebol.

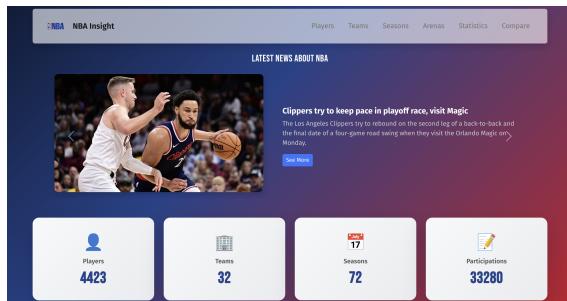


Figura 4.1: Página Inicial com informações gerais e notícias



Figura 4.2: Continuação da primeira página

- Listagem de Jogadores:** Exibe todos os jogadores únicos do grafo com informação básica. Inclui filtros por nome, posição, equipa, nacionalidade, escola, altura e ano de draft.

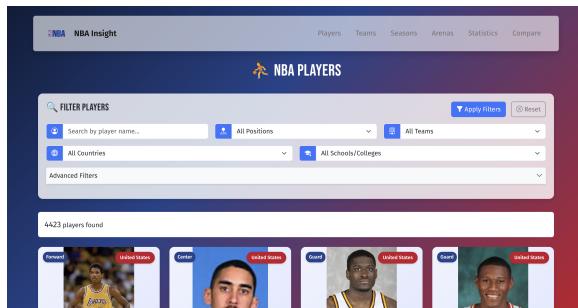


Figura 4.3: Página da listagem dos jogadores com secção de filtros

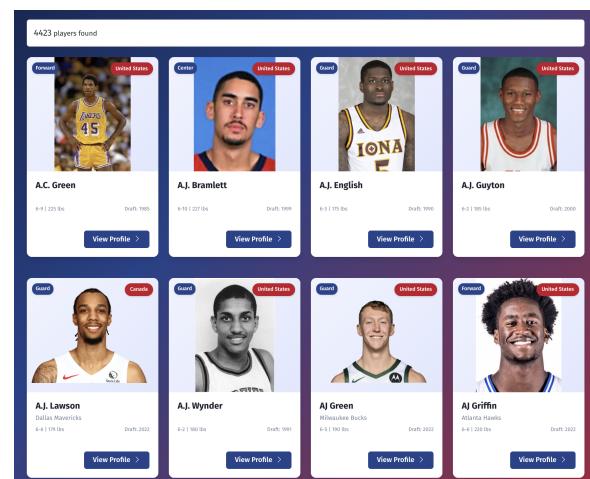
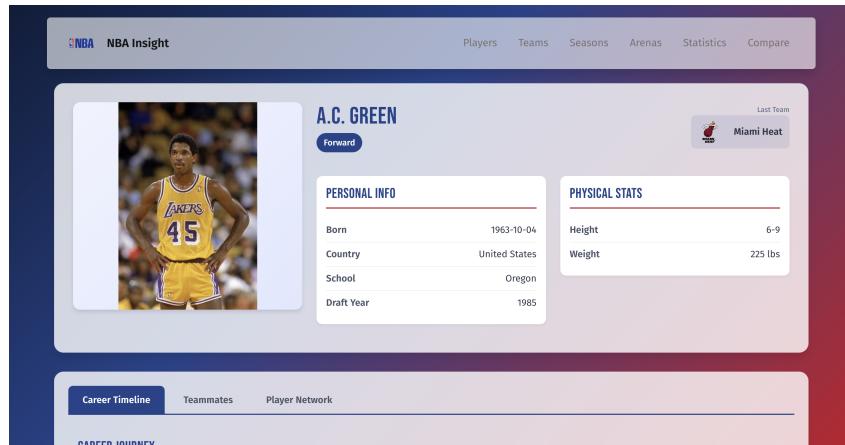


Figura 4.4: Continuação da página de listagem dos jogadores

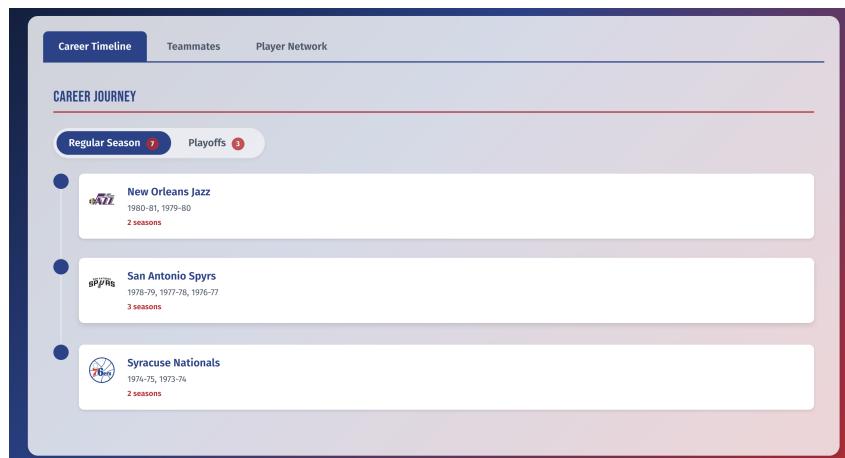
- **Página do Jogador:** Ao clicar num jogador, somos redirecionados para a sua página que mostra informações pessoais e profissionais, tais como a posição em que joga, a sua altura, etc.



The screenshot shows the NBA Insight player profile for A.C. Green. At the top, there's a navigation bar with links to Players, Teams, Seasons, Arenas, Statistics, and Compare. Below the header, there's a large image of A.C. Green in a Lakers jersey. His name, "A.C. GREEN", is displayed in bold capital letters, with "Forward" underneath it. To the right, there's a section titled "Last Team" showing the Miami Heat logo. On the left, there's a "PERSONAL INFO" section with details like birth date (1963-10-04), country (United States), school (Oregon), and draft year (1985). On the right, there's a "PHYSICAL STATS" section with height (6-9) and weight (225 lbs). At the bottom of the profile page, there are tabs for "Career Timeline", "Teammates", and "Player Network".

Figura 4.5: Página onde se podem ver detalhes de um jogador em específico

- **Linha Temporal do Jogador:** Visualização da carreira do jogador ao longo das temporadas, com distinção entre Regular e Playoffs.



The screenshot shows the NBA Insight Career Timeline page for A.C. Green. At the top, there are tabs for "Career Timeline", "Teammates", and "Player Network". Below the tabs, there's a section titled "CAREER JOURNEY" with a heading "Regular Season 7" and "Playoffs 3". Three team entries are listed: "New Orleans Jazz" (1980-81, 1979-80, 2 seasons), "San Antonio Spurs" (1978-79, 1977-78, 1976-77, 3 seasons), and "Syracuse Nationals" (1974-75, 1973-74, 2 seasons). Each entry includes the team logo and years played.

Figura 4.6: Vizualização da carreira do jogador

- **Grafo de Relações do Jogador:** Mostra relações entre um jogador, as equipas onde jogou e os companheiros de equipa de uma forma mais gráfica.

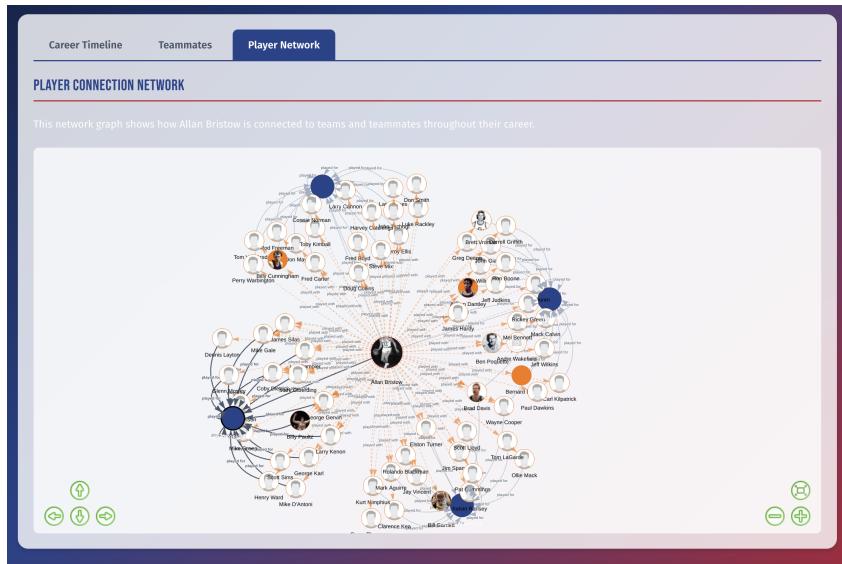


Figura 4.7: Grafo com as conexões ao longo da carreira do jogador

- **Companheiros de Equipa:** Apresenta os companheiros de equipa do jogador por época.

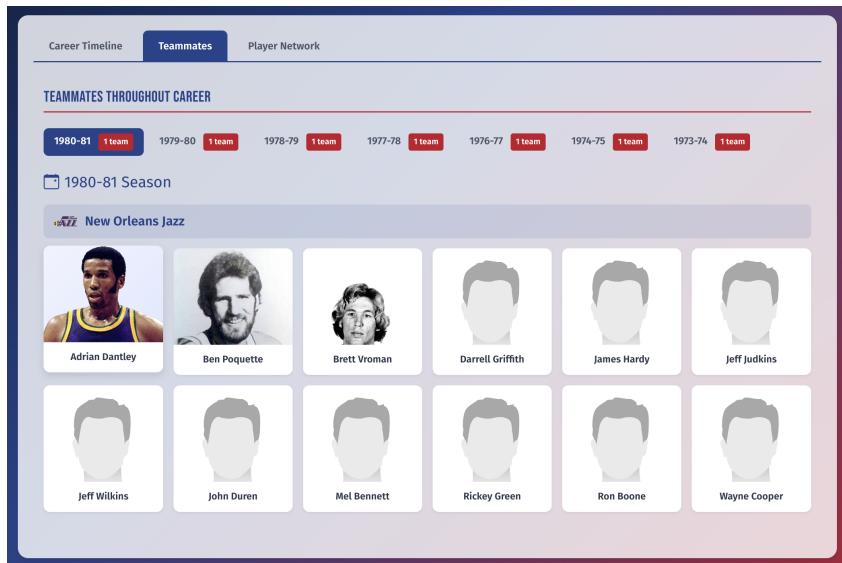


Figura 4.8: Colegas de carreira do jogador

- **Listagem de Equipas:** Passando à tab seguinte, vemos uma página com grid de equipas e os seus logótipos, acrónimos, nomes alternativos e histórico.

The screenshot shows the 'NBA TEAMS' section of the NBA Insight website. It features a grid of eight NBA teams per row. Each team card includes the team's logo, name, acronym, and a brief history or alternative names. A search bar is located at the top right of the grid.

Team	Acronym	History
Washington Wizards	CAP, WAS, CHP, BLT	Baltimore Bullets, Capital Bullets, Chicago Packers, Chicago Zephyrs
LA Clippers	SDC, BUF	Buffalo Braves, San Diego Clippers
Charlotte Hornets	CHE, CHA	
Sacramento Kings	KCK, SAC, CIN, ROC	Cincinnati Royals, Kansas City Kings, Rochester Royals
Detroit Pistons	DET, FTW	
Atlanta Hawks	ATL, MIH	Milwaukee Hawks, St. Louis Hawks
Los Angeles Lakers	LAL	
Brooklyn Nets	NJB, BKN	New Jersey Nets, New York Nets

Figura 4.9: Página de informação sobre as equipas

- **Página da Equipa:** Lista os detalhes da equipa e os jogadores que jogaram por ela, organizados por temporada.

The screenshot shows the 'MILWAUKEE HAWKS' team page. It displays the team's logo, name, and city (Atlanta). Below this, it shows the conference (Eastern Conference) and division (Southeast). A link to the arena information is also present. At the bottom, there is a section titled 'Historical Info' with links to other names and acronyms.

Figura 4.10: Detalhes da equipa

The screenshot shows the 'PLAYERS BY SEASON' section for the Milwaukee Hawks. It displays a grid of player names organized by season from 1951 to 1966. The grid is 6 rows by 4 columns. A note at the bottom indicates the data is from the 1950s project.

Season	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966
Art Burris	Bob Wilson	Cal Christensen	Charlie Black													
Dick Mehlen	Dillard Crocker	Don Boven	Don Otten													
Don Rehfeldt	Elmer Behnke	Gene Vance	James Owens													
Jerry Fowler	John McConathy	John Rennicke	Mel Hutchins													
		Nate DeLong	Walt Kirk													

Figura 4.11: Jogadores que jogaram na equipa selecionada

- **Listagem de Temporadas:** Nesta página são listadas todas as temporadas disponíveis de forma visual simples, com sistema de paginação.

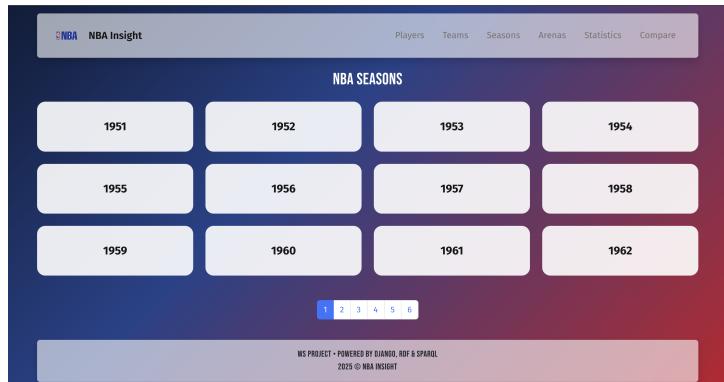


Figura 4.12: Página com a listagem das diferentes temporadas

- **Página da Temporada:** Mostra todas as equipas e jogadores participantes numa temporada específica.

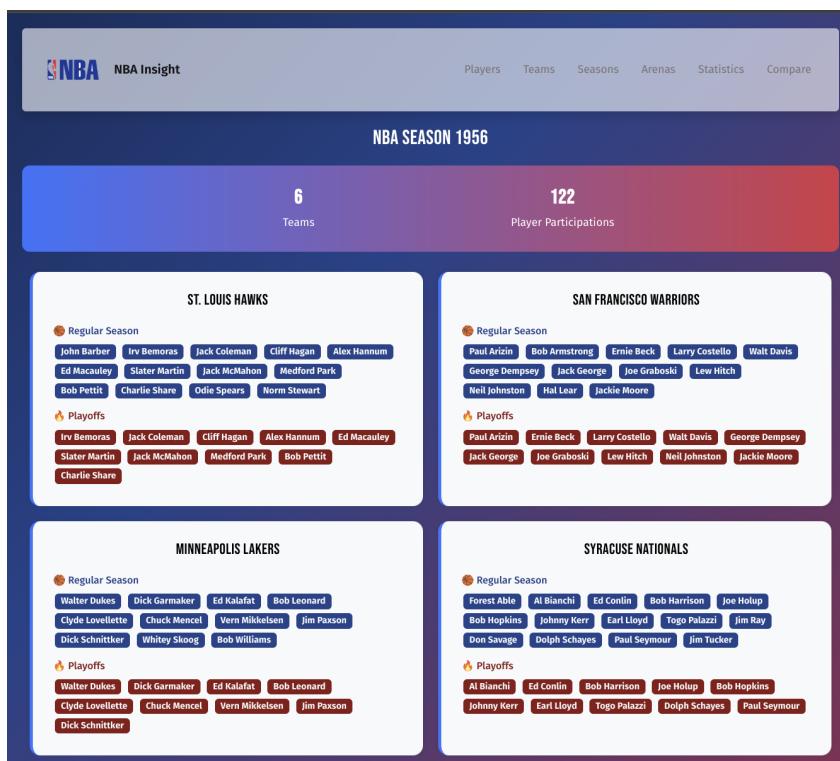


Figura 4.13: Mostra detalhes da temporada, as equipas da mesma e os seus jogadores

- **Listagem de Arenas:** Aqui são mostradas todas as arenas, sendo possível filtrá-las por localização, equipa da casa e lotação máxima. Ao clicar numa arena é aberta uma página com as suas informações específicas e um mapa com a sua localização.

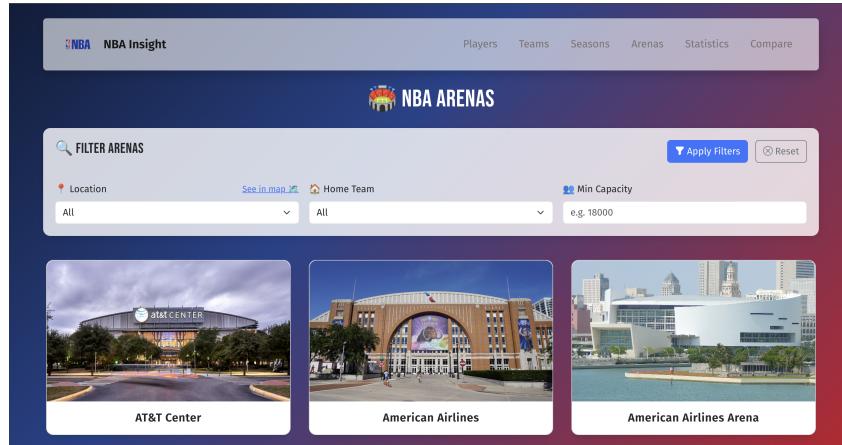


Figura 4.14: Página com a listagem das arenas da NBA

- **Pagina da Arena:** Mostra os detalhes de cada arena, incluindo localização, imagem, capacidade, coordenadas e equipa da casa.

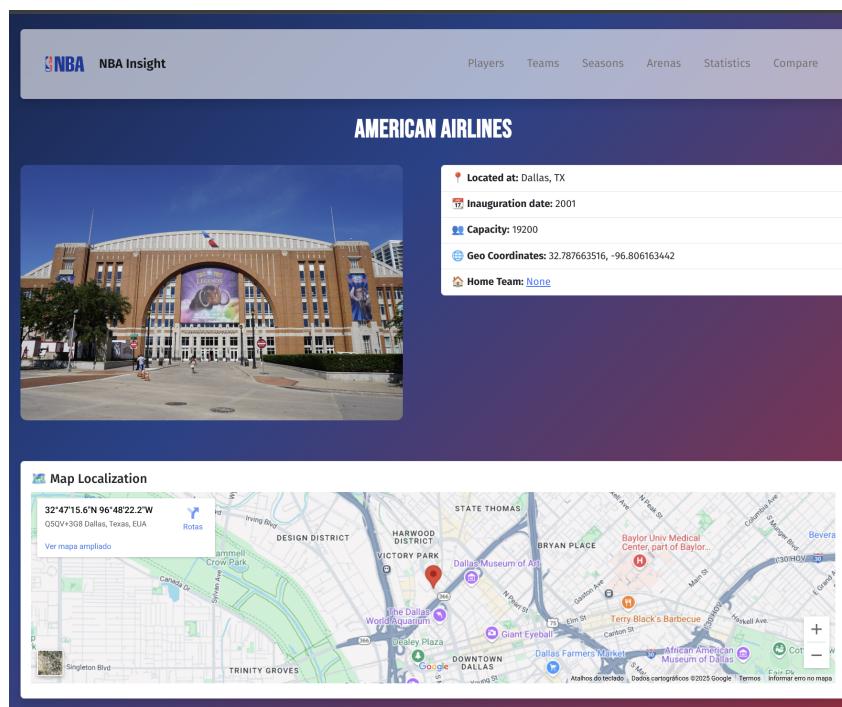


Figura 4.15: Vizualizador dos detalhes da arena selecionada

- **Mapa de Arenas:** Visualização interativa com um pin a identificar a localização de cada arena num mapa (Leaflet.js), incluindo a imagem e link para a página individual da arena. Esta página pode ser acedida através da HomePage ou da página de listagem das arenas.

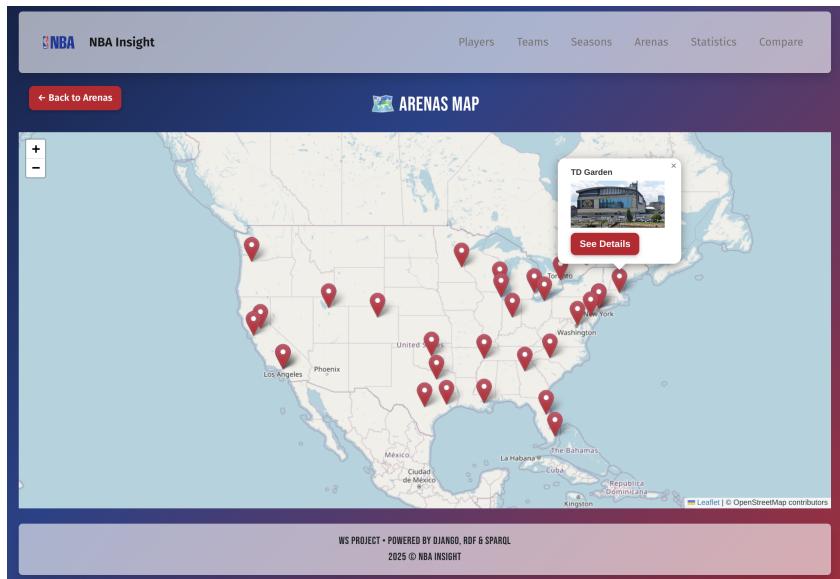


Figura 4.16: Mapa das arenas

- **Comparação de Jogadores:** Permite comparar dois jogadores lado a lado, destacando equipas, épocas e tipo de participações.

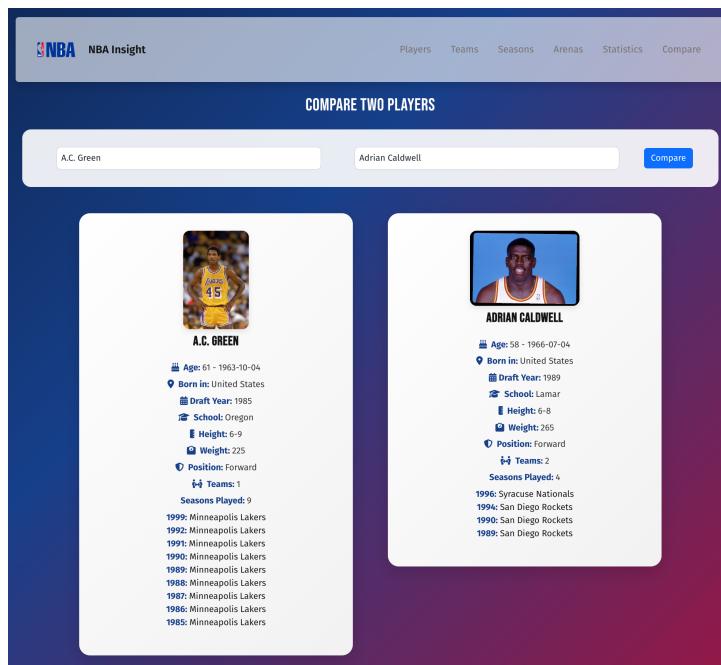


Figura 4.17: Pagina onde se pode comparar detalhes técnicos dos jogadores selecionados

- **Dashboard:** Inclui estatísticas e gráficos como:

- Participações por temporada;
- Jogadores por equipa;
- Top 10 jogadores com mais temporadas;
- Distribuição por posição;
- Altura média por posição (cm);

- Peso médio por posição (lbs);
- Distribuição de jogadores por ano de nascimento.

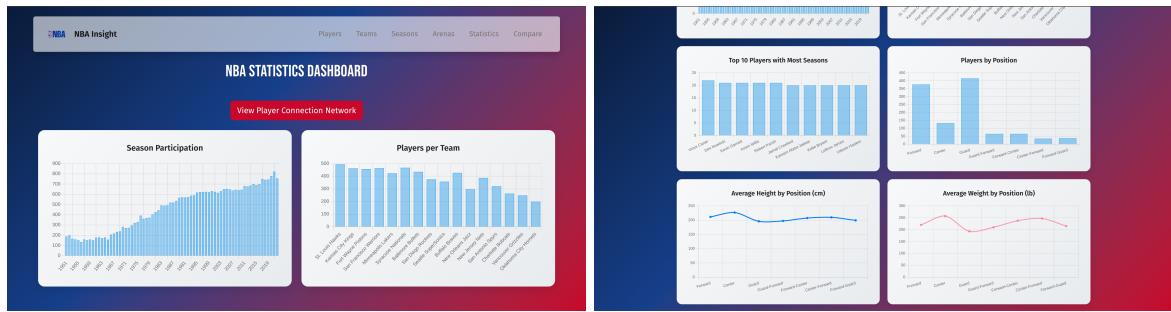


Figura 4.18: Vizualizão das estatísticas

Figura 4.19: Vizualização das estatísticas



Figura 4.20: Vizualização das estatísticas

- **Rede de Conexões:** Visualização de grafo com todos os jogadores que jogaram juntos, com opção de expandir nós ao clicar.

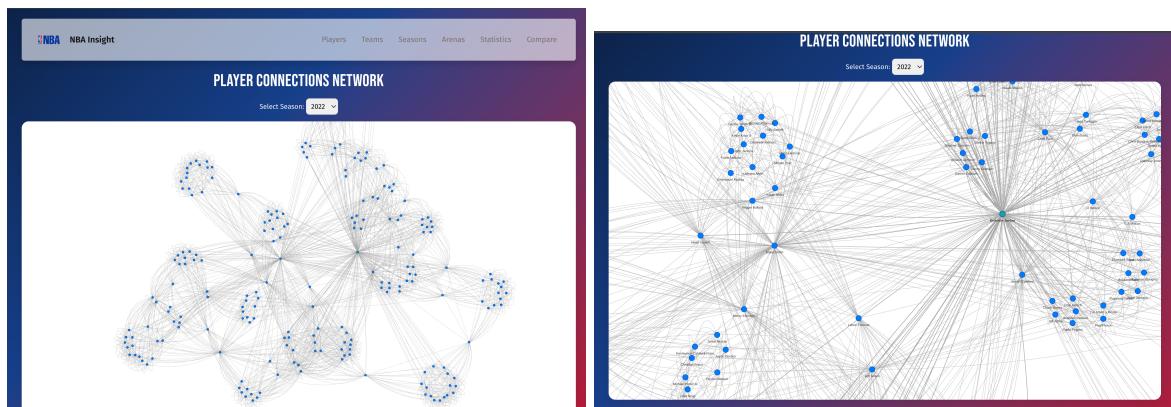
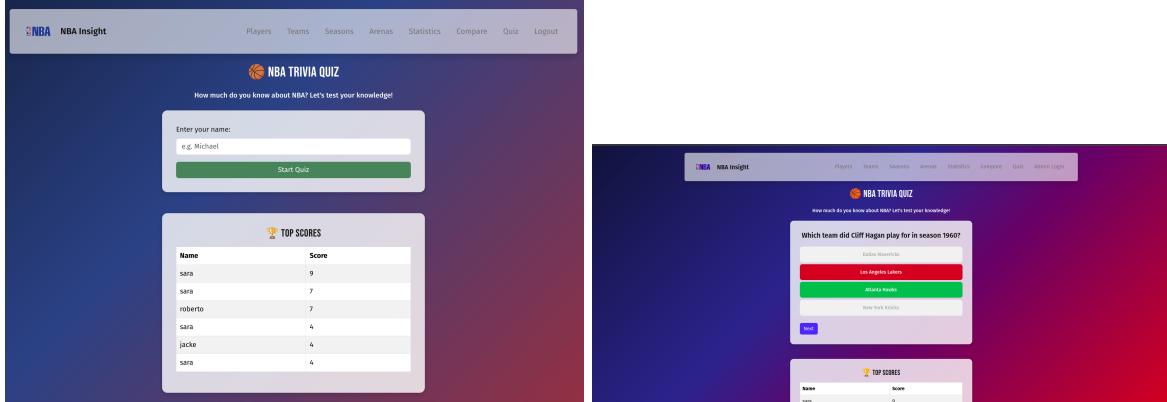


Figura 4.21: Grafo de conexos dos jogadores de uma dada temporada

Figura 4.22: Grafo de conexos dos jogadores de uma dada temporada - Detalhes

- **Jogo Interativo (Quiz NBA):** Funcionalidade lúdica onde o utilizador responde a perguntas de escolha múltipla sobre jogadores, equipas e arenas. As perguntas e respostas são geradas dinamicamente com queries SPARQL (SELECT) e as respostas são verificadas com queries SPARQL (ASK).



The figure consists of two side-by-side screenshots of the NBA Insight application's Quiz section. Both screenshots have a dark blue header bar with the NBA logo and 'NBA Insight' text, and a navigation menu with links: Players, Teams, Seasons, Arenas, Statistics, Compare, Quiz, and Logout.

Screenshot 1 (Left): Shows the 'NBA TRIVIA QUIZ' page. It asks 'How much do you know about NBA? Let's test your knowledge!'. Below is a form to 'Enter your name:' with placeholder 'e.g. Michael' and a green 'Start Quiz' button. To the right is a 'TOP SCORES' table:

Name	Score
sara	9
sara	7
roberto	7
sara	4
jackie	4
sara	4

Screenshot 2 (Right): Shows the Quiz interface after starting. It displays a trivia question: 'Which team did Cliff Hagan play for in season 1960?'. Three options are shown in colored boxes: Los Angeles Lakers (red), Atlanta Hawks (green), and New York Knicks (purple). Below the question is a 'Next' button.

Figura 4.23: Quiz - Inserir Nome e Pontuações

Figura 4.24: Quiz

Integração Visual e Design

A aplicação utiliza um design moderno com **Bootstrap**, animações via **AOS.js**, elementos responsivos e filtros dinâmicos em JavaScript. A navegação é simples e rápida, permitindo explorar ligações entre entidades com fluidez.

5. Funcionalidades de Administrador (Autenticação e Operações CRUD)

Esta secção descreve as funcionalidades de administrador implementadas para utilizadores autenticados com permissões de `staff`. As operações de criação, edição e remoção de entidades RDF são realizadas dinamicamente a partir da interface web e traduzidas diretamente em queries SPARQL. A autenticação e segurança são garantidas com a biblioteca `django.contrib.auth` e as credenciais de acesso são:

- Username: NBAAdmin
- Password: adminpass123

Autenticação de Utilizador

A autenticação é então realizada com a biblioteca `django.contrib.auth`. Apenas utilizadores com a flag `is_staff` conseguem aceder às funcionalidades administrativas. A view processa as credenciais e, se válidas, ativa a sessão com:

```
user = authenticate(request, username=username, password=password)
if user is not None and user.is_staff:
    login(request, user)
```

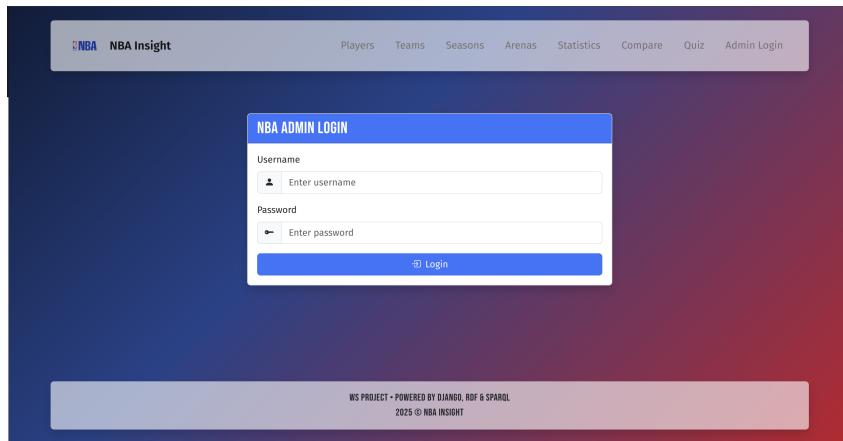


Figura 5.1: Formulário de login protegido

Criação de Jogador

A criação de jogadores é feita através de um modal acessível apenas a administradores. O formulário aceita todos os campos relevantes e envia os dados via `fetch` para o backend, que traduz o pedido em SPARQL:

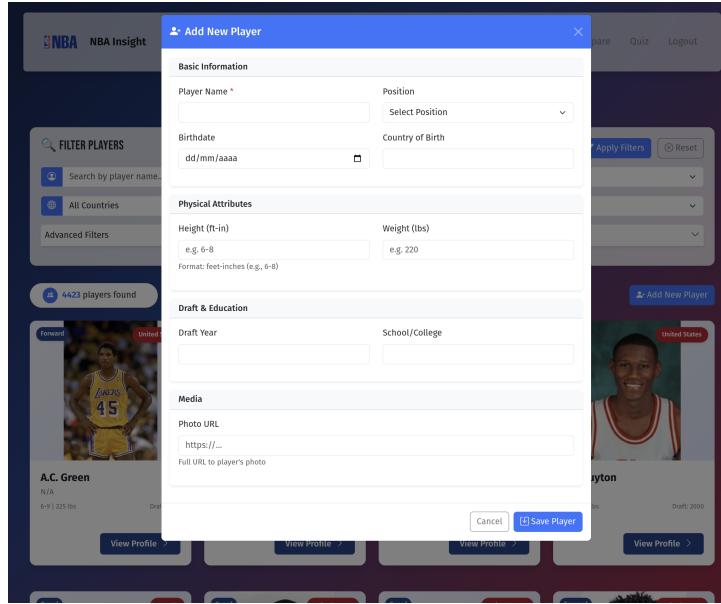


Figura 5.2: Modal de criação de jogador

Esta abordagem garante que cada novo jogador é imediatamente representado como um recurso completo no grafo RDF.

Atualização de Jogador

Para editar dados, é exibido um modal pré-preenchido. Cada alteração gera uma query DELETE/INSERT SPARQL que atualiza a propriedade:

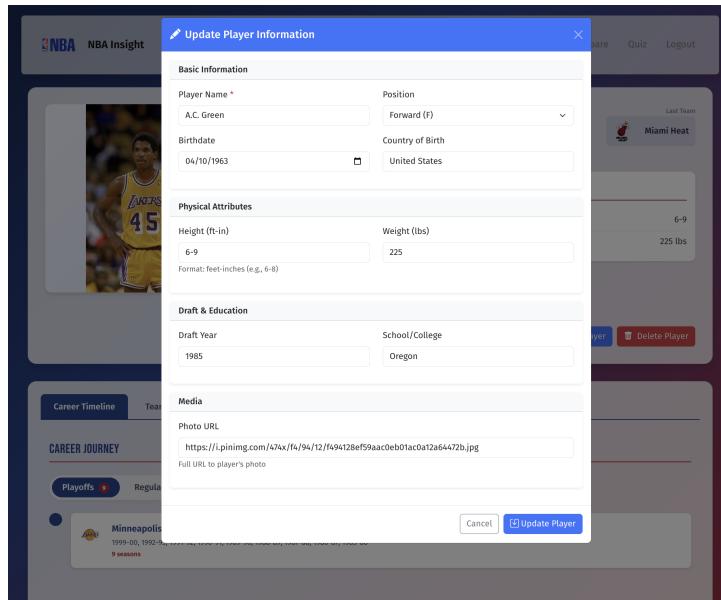


Figura 5.3: Modal de edição de jogador

Este mecanismo permite alterações seguras e precisas a qualquer propriedade RDF do jogador.

Remoção de Jogador

A remoção de jogadores é protegida por confirmação. A ação elimina todos os triplos associadas ao jogador:

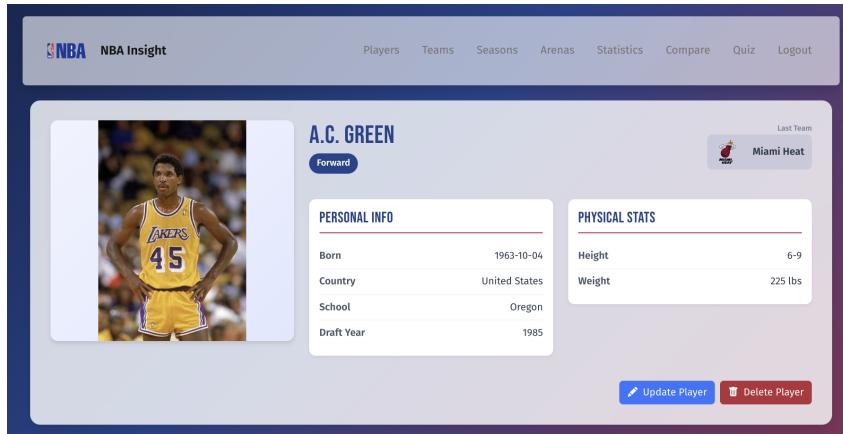


Figura 5.4: Botões de admin visíveis após login

6. Conclusões

O desenvolvimento deste projeto permitiu uma exploração aprofundada das potencialidades que a Web Semântica oferece, e aplicá-las num domínio real e complexo como é o da NBA.

A transformação de dados não estruturados para um modelo semântico coerente foi um passo fundamental para possibilitar queries expressivas e dinâmicas. A estruturação num modelo RDF permitiu, implementar funcionalidades avançadas como visualização de redes de jogadores, dashboards estatísticos, filtros complexos e até um jogo de perguntas que é gerado por queries SPARQL.

A componente administrativa, que inclui autenticação e operações CRUD, demonstrou a viabilidade de manter e evoluir o conhecimento representado de forma semântica através de uma interface amigável.

Este trabalho evidencia como os princípios da Web Semântica podem ser aplicados com sucesso para construir aplicações ricas em informação, relacionamentos e funcionalidades.

7. Configurações para executar a app

O projeto encontra-se totalmente funcional e pode ser executado de duas formas distintas:

- **Execução Manual**
- **Execução com Docker**

Em ambas as abordagens, o sistema cria automaticamente o repositório no GraphDB e carrega os dados RDF na primeira execução.

Repositório do Projeto

O código-fonte encontra-se disponível publicamente no GitHub em:

```
github.com/RobertoCastro391/WS_Projects
```

Requisitos

- - Ter o triplestore GraphDB instalado e a correr e criar um ambiente virtual onde instalar as dependências, no caso da primeira opção de execução;
- Docker e docker-compose instalados, no caso da segunda opção de execução;
- Porta 7200 (GraphDB) e 8000 (Django) livres no sistema;

7.1 Execução Manual

Passos

1. Clonar o repositório e aceder à pasta do projeto:

```
git clone https://github.com/RobertoCastro391/WS_Projects.git
cd WS_Projects/NBAProject
```

2. Criar e ativar um ambiente virtual:

```
python3 -m venv venv
source venv/bin/activate          # Em Linux/macOS
venv\Scripts\activate             # Em Windows
```

3. Instalar as dependências:

```
pip install -r requirements.txt
```

4. Iniciar o servidor Django:

```
python manage.py runserver
```

Configuração Automática do GraphDB

Na execução manual, o ficheiro `startup.py` é chamado automaticamente pelo Django ao arrancar a aplicação. Este ficheiro:

- Aguarda até o GraphDB estar acessível;
- Verifica se o repositório `NBA_G4` já existe;
- Se não existir, cria o repositório com base no ficheiro `data/nba-config.ttl`;
- Carrega automaticamente os dados RDF do ficheiro `data/nba_triples.n3`.

Este processo é iniciado via `apps.py` ao arranque da aplicação.

O utilizador pode aceder:

- à aplicação Django via: `http://localhost:8000`
- ao GraphDB via: `http://localhost:7200`

7.2 Execução com Docker

Passos

1. Clonar o repositório e aceder à pasta do projeto:

```
git clone https://github.com/RobertoCastro391/WS_Projects.git
cd WS_Projects/NBAProject
```

2. Iniciar os serviços com o Docker Compose:

```
docker compose up --build
```

Serviços

O Docker Compose configura e executa os seguintes serviços:

- **GraphDB**: serviço de triplestore na porta 7200;
- **Django**: aplicação web na porta 8000;
- **graphdb-init**: container temporário que:
 - Cria o repositório `NBA_G4` com base no ficheiro `data/nba-config.ttl`;
 - Carrega os dados RDF a partir de `data/nba_triples.n3`;
 - Termina automaticamente após a configuração.

Acesso

- Interface Web da Aplicação: `http://localhost:8000`
- Interface do GraphDB (Workbench): `http://localhost:7200`

7.3 Ficheiros Importantes

- `docker-compose.yml`: orquestra os serviços (Django, GraphDB, init);
- `Dockerfile`: constrói a imagem da aplicação Django;
- `docker/graphdb-init.sh`: cria o repositório e importa os dados RDF, quando a app é executada pelo docker;
- `data/nba-config.ttl`: configura o repositório no GraphDB;
- `data/nba_triples.n3`: contém os dados RDF em formato N3;
- `app/startup.py`: configuração automática usada na execução manual;
- `requirements.txt`: lista de dependências Python.

Carregamento automático de dados RDF

No primeiro arranque, o script `graphdb-init.sh` é executado automaticamente e:

1. Cria o repositório GraphDB com base no ficheiro `repo-config.ttl`;
2. Carrega o ficheiro RDF `nba_triples.n3` no repositório;
3. O serviço termina após o carregamento, e os dados ficam disponíveis para consulta.

Nota

Caso os dados RDF sejam alterados, o utilizador poderá reconstruir o projeto com o comando:

```
docker-compose up --build --force-recreate
```