

MoviesDB

Universidade de Aveiro

Inês Justo, Miguel Neves,
Rafael Maio, Raquel Rainho



MoviesDB

Departamento de Eletrónica, Telecomunicações e Informática
Engenharia de Dados e Conhecimento

Universidade de Aveiro

(84804) inesjusto@ua.pt
(67453) miguelneves@ua.pt
(84909) rafael.maio@ua.pt
(84891) raquel.a.rainho@ua.pt

22 de dezembro de 2019

Conteúdo

1	Introdução	1
2	Dados, suas fontes e sua transformação	2
2.1	IMDB 5000 Movie Dataset	2
2.2	RDF	2
3	Operações sobre os dados (SPARQL)	4
3.1	Operações SELECT	4
3.1.1	Filtros de pesquisa	4
3.1.2	Pesquisa por texto	5
3.1.3	Apresentação dos dados	6
3.2	Operações UPDATE	6
3.3	Operações ASK	8
3.4	Biblioteca python s4api	9
4	Publicação de dados semânticos através de RDFa	10
4.1	Movie_page.html	10
4.2	Actor_profile.html e Director_profile.html	11
4.3	News.html	13
5	Integração de dados da Wikidata	14
6	Funcionalidades da Aplicação (UI)	15
6.1	Filmes (<i>Movies</i>)	15
6.2	Atores (<i>Actors</i>)	18
6.3	Realizadores (<i>Directors</i>)	19
6.4	Notícias (<i>News</i>)	20
7	Conclusões	21
8	Configuração para Executar a Aplicação	22

Capítulo 1

Introdução

MoviesDB é uma aplicação Web criada com o intuito de apresentar informação relativa ao mundo cinematográfico, tendo sido dividida em quatro partes principais:

- Uma secção de filmes, onde é apresentada uma lista de filmes e respetivos detalhes (tais como o ano de lançamento, géneros, pontuação, entre outros);
- Uma secção de atores, onde são listados todos os atores que participam nos filmes existentes na aplicação;
- Uma secção de realizadores, semelhante à de atores, onde são apresentados todos os realizadores dos filmes existentes;
- Uma secção de notícias, com informação atualizada sobre o mundo cinematográfico.

Para o desenvolvimento da aplicação foram utilizadas as seguintes tecnologias: Python/Django, RDF no formato N3, SPARQL, Triplestore GraphDB e RDFa.

Este projecto partilha o tema com o anterior realizado na UC, tendo sido reaproveitado grande parte da sua interface e dos dados utilizados (necessitando estes de serem adaptados ao novo formato).

Capítulo 2

Dados, suas fontes e sua transformação

2.1 IMDB 5000 Movie Dataset

O *dataset* "movie_metadata.csv" (no directório "xml") foi obtido na comunidade de *data scientists* e *machine learners*, Kaggle. Embora contendo uma grande variedade de informação, este *dataset* não serviria um dos objetivos propostos no seu formato original. Deste modo, no projecto realizado anteriormente (em <https://github.com/ijusto/Data-and-Knowledge-Engineering-Project-1.git>), a transformação do CSV em XML era fundamental, pelo que se criou o ficheiro "movies.xml", bem como o ficheiro "movies_short.xml".

Tendo em mente a página de informação de cada actor e realizador envolvidos nos filmes apresentados, foram guardados no ficheiro "people.xml" as fotos e biografias, retirados da base de dados IMDb, referentes a cada pessoa presente no ficheiro "movies.xml".

2.2 RDF

De forma a modelar os dados conforme o objectivo do projecto, foi decidido usar o formato N3 do RDF com o auxílio de vocabulários baseados em URIs. Para conseguir que o modelo de dados se apresentasse neste formato, foi criado um *script* - o ficheiro "xmlToRDF.py" -, onde se converteu o ficheiro "movies_short.xml", escrito em XML, para o formato pretendido. Para além da conversão deste ficheiro, foi também convertido o ficheiro "people.xml", também este escrito em XML. Finalmente, foram adicionados manualmente os triplos referentes aos géneros e *ratings*, com o objetivo de tornar as consultas mais rápidas, resultando no ficheiro "movies.n3".

```
@prefix mov: <http://moviesDB.com/entity/mov>.  
@prefix person: <http://moviesDB.com/entity/person>.  
@prefix genres: <http://moviesDB.com/entity/genres>.  
@prefix ratings: <http://moviesDB.com/entity/ratings>.  
@prefix predicate: <http://moviesDB.com/predicate>.
```

Figura 2.1: Exemplos de URIs

```

genres:horror predicate:name "Horror".
genres:mystery predicate:name "Mystery".
genres:thriller predicate:name "Thriller".
genres:action predicate:name "Action".
genres:adventure predicate:name "Adventure".
genres:fantasy predicate:name "Fantasy".
genres:sci-fi predicate:name "Sci-Fi".
genres:romance predicate:name "Romance".
genres:animation predicate:name "Animation".
genres:comedy predicate:name "Comedy".
genres:family predicate:name "Family".
genres:musical predicate:name "Musical".
genres:western predicate:name "Western".
genres:drama predicate:name "Drama".
genres:history predicate:name "History".
genres:crime predicate:name "Crime".
genres:war predicate:name "War".
genres:biography predicate:name "Biography".
genres:music predicate:name "Music".
ratings:r predicate:name "R".
ratings:pg predicate:name "PG".
ratings:pg_13 predicate:name "PG-13".
ratings:g predicate:name "G".

```

Figura 2.2: Exemplos de géneros e ratings

```

mov:avatar
  predicate:name "Avatar";
  predicate:year "2009";
  predicate:score "7.9";
  predicate:actor
    person:CCH_Pounder,
    person:Joel_Moore,
    person:Wes_Studi;
  predicate:poster "https://m.media-amazon.com/images/M/MV5BMTYwOTEwMjAzMl5BMl5BanBnXkFtZTcwODc5MTUwMw@@._V1_UX182_CR0,0,182,268_AL_.jpg";
  predicate:director
    person:James_Cameron;
  predicate:genre
    genres:action,
    genres:adventure,
    genres:fantasy,
    genres:sci-fi;
  predicate:plot_keyword
    "avatar",
    "future",
    "marine",
    "native",
    "paraplegic";
  predicate:country "USA";
  predicate:language "English";
  predicate:rating ratings:pg_13;
  predicate:duration "178";
  predicate:budget "237000000".

```

Figura 2.3: Exemplo de um filme

```

person:Pierre_Morel
  predicate:name "Pierre Morel";
  predicate:profession "Director";
  predicate:image "https://m.media-amazon.com/images/M/MV5BMjE3NDcyNzY4Nl5BMl5BanBnXkFtZTcwNDg5NDUxMw@@._V1_UY317_C";
  predicate:bio "Pierre Morel was born on May 12, 1964 in France. He is known for his work on Os Gangs do Bairro 13

```

Figura 2.4: Exemplo de uma pessoa (ator e/ou diretor)

Capítulo 3

Operações sobre os dados (SPARQL)

3.1 Operações SELECT

Esta é a operação SPARQL mais utilizada e tem como objetivo consultar a triplestore GraphDB, retornando, de entre todas as ocorrências correspondentes à consulta solicitada, os dados pedidos. Em todas as suas utilizações, a cláusula WHERE permite que os resultados obtidos sejam apenas os triplos que obedecem às condições impostas, como conter um predicado específico.

3.1.1 Filtros de pesquisa

Quando são utilizados os filtros de pesquisa e/ou as opções de ordenação, a operação de SELECT aplicada tem como objetivo selecionar todos os triplos de um filme, usando a restrição FILTER para obter apenas os filmes pretendidos, dependendo do gênero, ano e/ou *rating*. Usando também o modificador ORDER BY possibilita ordenar estes resultados por ano, *score* ou título.

Caso sejam selecionados vários gêneros, são obtidos filmes com todos ou apenas algum(ns) deles, mas no caso de ser selecionado um rating e/ou ano específicos, os filmes obtidos contêm obrigatoriamente esses valores. (Por exemplo, sendo feita uma filtragem com *Genres: "Action", "Adventure"; Rating: "PG"; Year: "2001"*, são obtidos filmes com a classificação "PG", do ano 2000 e dos gêneros de ação e/ou aventura.)

```
query = """
PREFIX pred: <http://moviesDB.com/predicate/>
SELECT distinct ?title ?pred ?obj
WHERE {
    ?movie ?pred ?obj .
    ?movie pred:rating ?rating .
    ?movie pred:year ?year .
    ?movie pred:genre ?genre .
    ?rating pred:name ?rating_name .
    ?genre pred:name ?genre_name .
    ?movie pred:name ?title .
    ?movie pred:score ?score .
}
"""

query += """FILTER(?genre_name IN(" " + aux + " "))"""

query += """FILTER (?year = \"\" + request.POST['years'] + \"\")"""

query += """FILTER (?rating_name = \"\" + request.POST['ratings'] + \"\")"""

query += """ORDER BY ?\" + request.POST['orderby'].lower() + \"\""""

query += """}"""
```

Figura 3.1: Query de filtros de pesquisa

3.1.2 Pesquisa por texto

Na página inicial, responsável por listar os filmes, o utilizador pode pesquisar por palavras contidas no título ou nas palavras-chave (*plot keywords*) de um filme. Neste caso, a operação utilizada contém a restrição FILTER para restringir os resultados a apenas aqueles que contenham o texto pesquisado nos campos mencionados.

```
query = """
    PREFIX pred: <http://moviesDB.com/predicate/>
    SELECT distinct ?title ?pred ?obj
    WHERE {
        {
            ?movie ?pred ?obj .
            ?movie pred:director ?director .
            ?movie pred:name ?title .
            FILTER(CONTAINS(lcase(?title), \"\" + request.POST['search'].lower() + \"\"))
        } UNION {
            ?movie ?pred ?obj .
            ?movie pred:name ?title .
            ?movie pred:plot_keyword ?keywords .
            FILTER(CONTAINS(lcase(?keywords), \"\" + request.POST['search'].lower() + \"\"))
        }
    }
    """
```

Figura 3.2: *Query* de pesquisa de filmes

Nas páginas de atores e realizadores, o utilizador pode ainda pesquisar pelo nome de um destes. Ambas as pesquisas são feitas inserindo o nome, ou parte deste, de um determinado ator ou realizador. De forma idêntica à pesquisa mencionada anteriormente, a restrição FILTER permite obter como resultados apenas os atores/realizadores cujo nome contenha o texto introduzido.

```
query = """
    PREFIX pred: <http://moviesDB.com/predicate/>
    SELECT distinct ?name
    WHERE {
        ?movie pred:actor ?actor .
        ?actor pred:name ?name .
        FILTER (CONTAINS(lcase(?name), \"\" + request.POST['search'].lower() + \"\"))
    }
    """
```

Figura 3.3: *Query* de pesquisa de atores

```
query = """
    PREFIX pred: <http://moviesDB.com/predicate/>
    SELECT distinct ?name
    WHERE {
        ?movie pred:director ?director .
        ?director pred:name ?name .
        FILTER (CONTAINS(lcase(?name), \"\" + request.POST['search'].lower() + \"\"))
    }
    """
```

Figura 3.4: *Query* de pesquisa de realizadores

3.1.3 Apresentação dos dados

De forma a apresentar toda a informação relativa a um filme, a operação de pesquisa retorna todos os triplos a ele associados, sendo utilizada a restrição REGEX para obter apenas resultados cujo sujeito tenha o nome do filme desejado. A informação retirada destes é organizada conforme a relação (predicado) de cada item com o filme.

A obtenção da informação relativa a um ator ou realizador é feita de maneira semelhante, e utilizada na apresentação das suas páginas pessoais.

```
query = """
    PREFIX mov: <http://moviesDB.com/entity/mov>
    prefix predicate: <http://moviesDB.com/predicate/>
    select ?pred ?obj where{
        ?film ?pred ?obj.
        ?film predicate:name ?name
        filter regex(?name,\"\""+movie.replace("/"," ").replace("!","")+\"\""+\"i\").
    }
    """
```

Figura 3.5: *Query* de pesquisa de um filme

```
for e in res['results']['bindings']:
    if "name" in e['pred']['value']:
        name = e['obj']['value']
    elif "year" in e['pred']['value']:
        year = e['obj']['value']
    elif "poster" in e['pred']['value']:
        poster = e['obj']['value']
    elif "score" in e['pred']['value']:
        score = e['obj']['value']
    elif "actor" in e['pred']['value']:
        movie_main_actors.append(e['obj']['value'])
    elif "director" in e['pred']['value']:
        movie_director = e['obj']['value']
    elif "genre" in e['pred']['value']:
        movie_genres.append(e['obj']['value'])
    elif "plot_keyword" in e['pred']['value']:
        plot_keywords.append(e['obj']['value'])
    elif "country" in e['pred']['value']:
        country = e['obj']['value']
    elif "language" in e['pred']['value']:
        language = e['obj']['value']
    elif "rating" in e['pred']['value']:
        rating = e['obj']['value']
    elif "duration" in e['pred']['value']:
        duration = e['obj']['value']
    elif "budget" in e['pred']['value']:
        budget = e['obj']['value']
```

Figura 3.6: Organização da informação resultante da *query* de pesquisa de um filme

3.2 Operações UPDATE

As operações UPDATE são responsáveis por atualizar a triplestore GraphDB. Podem ser de dois tipos, INSERT e DELETE, que correspondem à inserção e eliminação de dados, respetivamente. Estas atualizações podem ser feitas diretamente (por exemplo, na adição de um novo filme, a inserção do triplo que associa o seu ano é feita diretamente, explicitando o URI para a entidade correspondente ao filme e o literal correspondente ao ano - figura 3.7) ou indiretamente (por exemplo, na adição de um novo filme, a inserção do triplo que associa um dos seus géneros é auxiliada por uma "pesquisa" da entidade relativa a esse género - figura 3.8).

```

query = """
    PREFIX mov: <http://moviesDB.com/entity/mov>
    PREFIX pred: <http://moviesDB.com/predicate/>
    INSERT { mov:"" + clean_title + "" pred:genre ?genre }
    WHERE {
        ?movie pred:genre ?genre .
        ?genre pred:name ?name .
        FILTER (?name = \"\" + g + \"\")
    }
    """

```

Figura 3.7: Query de inserção direta de um novo filme

```

query = """
    PREFIX mov: <http://moviesDB.com/entity/mov>
    PREFIX pred: <http://moviesDB.com/predicate/>
    INSERT DATA {
        mov:"" + clean_title + " pred:name \"\" + request.POST['title'] + \"\" ;
        pred:year \"\" + request.POST['year'] + \"\" ;
        pred:duration \"\" + request.POST['duration'] + \"\" ;
        pred:director <http://moviesDB.com/entity/person/\"\" + director + \"\"> ;
        pred:actor <http://moviesDB.com/entity/person/\"\" + actor1 + \"\"> ,
                  <http://moviesDB.com/entity/person/\"\" + actor2 + \"\"> ,
                  <http://moviesDB.com/entity/person/\"\" + actor3 + \"\"> ;
        pred:budget \"\" + budget + \"\" ;
        pred:country \"\" + country + \"\" ;
        pred:language \"XXX\" ;
        pred:poster \"https://encrypted-tbn0.gstatic.com/images?q=tbn%3AAND9GcT16wQWF2p4_8GBLCN\" ;
        pred:score \"?\" .
    }
    """

```

Figura 3.8: Query de inserção indireta de um novo filme

Como se pode verificar na figura 3.9, a operação exemplificada é responsável por apagar todos os triplos relativos a um dado filme, passando este a não estar presente na base de dados e deixando assim de ser possível encontrá-lo na página web.

```

update = """
    prefix predicate: <http://moviesDB.com/predicate/>
    DELETE {?film ?pred ?obj.} where{
        ?film ?pred ?obj.
        ?film predicate:name ?name.
        filter regex(?name,\"\" + movie + \"\"\\",\"i").
    }
    """

```

Figura 3.9: Query de remoção de um filme

3.3 Operações ASK

São utilizadas operações ASK na validação dos dados de um novo filme a inserir na base de dados, para verificar se o(s) género(s) e *rating* introduzidos existem na base de dados.

```
query = """
    PREFIX gen: <http://moviesDB.com/entity/genres/>
    PREFIX pred: <http://moviesDB.com/predicate/>
    ASK {
        { gen:"" + g.lower() + " pred:name \"\" + g + "" . }
        UNION
        { ?movie pred:genre ?genre .
          | ?genre pred:name \"\" + g + "" .
        }
    }
    """
```

Figura 3.10: *Query* de validação dos géneros do novo filme

```
query = """
    PREFIX gen: <http://moviesDB.com/entity/ratings/>
    PREFIX pred: <http://moviesDB.com/predicate/>
    ASK {
        { gen:"" + rating + " pred:name \"\" + request.POST['rating'] + "" . }
        UNION
        { ?movie pred:rating ?rating .
          | ?rating pred:name \"\" + request.POST['rating'] + "" .
        }
    }
    """
```

Figura 3.11: *Query* de validação do rating do novo filme

Exemplo de dados válidos para um novo filme a inserir:

MovieDB

New movie

Title: *

Year: *

Director: *

First name: Last name:

Actor 1: *

First name: Last name:

Actor 2: *

First name: Last name:

Actor 3: *

First name: Last name:

Duration: *

Genres:

*

Rating: *

Country:

Budget:

*** - required**

Figura 3.12: Página para inserção de um filme na base de dados

3.4 Biblioteca python s4api

Com o intuito de consultar a base de dados presente na triplestore GraphDB e obter a informação necessária para ser mostrada nas diferentes funcionalidades da aplicação, ou até mesmo atualizar a base de dados, usou-se a biblioteca *python s4api*, que implementa as operações básicas de acesso, via *REST*. Os resultados das operações SPARQL são retornados no formato de um dicionário, através do qual os dados são transformados para serem utilizados da forma pretendida (figura 3.6).

Capítulo 4

Publicação de dados semânticos através de RDFa

Até recentemente, a informação apresentada nas páginas Web tem sido organizada majoritariamente para consumo humano. Apesar dos *browsers* conseguirem exibir facilmente a informação, os computadores não conseguem perceber o significado do conteúdo exposto.

A utilização de RDFa serve precisamente para especificar a semântica por trás da informação apresentada, de modo a poder ser interpretada pela máquina.

A aplicação MoviesDB faz uso de RDFa nas páginas HTML "movie_page.html", "actor_profile.html", "director_profile.html" e "news.html", utilizando o vocabulário de *schema.org* para descrever os tipos de dados.

4.1 Movie_page.html

A informação apresentada nesta página mostra os detalhes de um determinado filme, portanto foi utilizado o tipo "Movies". Para os atributos foram utilizadas as propriedades "name" (para o nome do filme), "datePublished" (para o seu ano de publicação), "duration" (para a sua duração), "genre" (para os géneros a que o filme pertence), entre outros. Foram também utilizadas as propriedades "actor" e "director" que, como se referem a pessoas, foram caracterizadas pelo tipo "Person".

```
<div vocab="http://schema.org/" typeof="Movie" class="col-md-12 align-items-center">
  <div class="d-flex flex-row" style="...">
    <div class="p-2">
      
    </div>
    <div class="p-2">
      <h1 property="name"><!--class="card-title"-->{{ movie_name }}
```

Figura 4.1: Excerto de código de "movie_page"

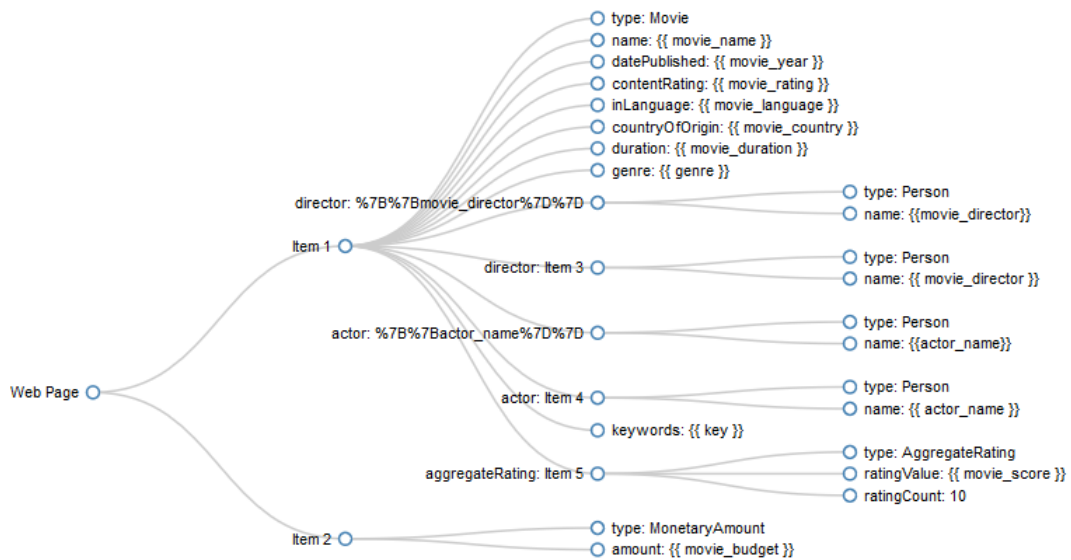


Figura 4.2: Visualização dos dados de "movie_page"

4.2 Actor_profile.html e Director_profile.html

Em "actor_profile"/"director_profile", é apresentada informação de um determinado ator/realizador, contendo uma foto, o seu nome e biografia. Para este caso, foi utilizado o tipo "Person" e as propriedades "image", "name" e "description".

```
<div vocab="http://schema.org/" typeof="Person" class="col-md-12 align-items-center" style="...">
  <h1 class="my-4">{{ title_name }}
  <!-- TODO category_selected -->
</h1>

  <div class="d-flex flex-row" style="...">
    <div class="p-2">
      
    </div>
    <div class="p-2">
      <h2 property="name">{{actor_name}}</h2>
    </div>
  </div>
  <div class="d-flex flex-row" style="...">
    <p property="description" align="justify">{{actor_bio}}</p>
  </div>
  <p typeof="Movie" style="...">
    <strong><i>Movies: </i></strong>
    {% for movieName in movies %}
      <a property="name" href="../../movie/{{movieName}}">
        {{movieName}}
      </a>
      <br/>
    {% endfor %}
  </p>
</div>
```

Figura 4.3: Excerto de código de "actor_profile"

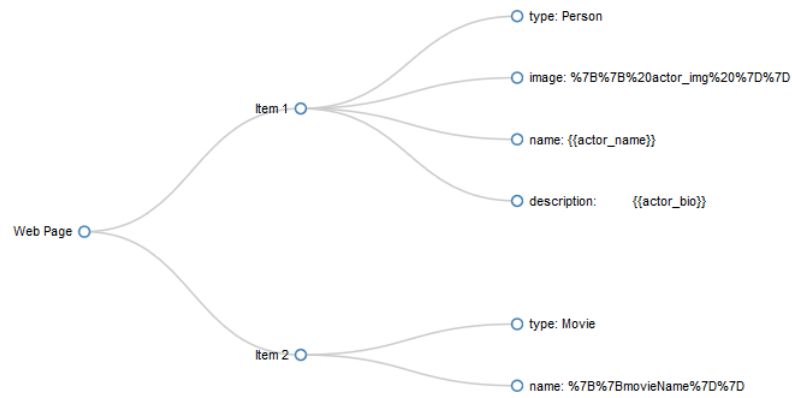


Figura 4.4: Visualização dos dados de "actor_profile"

```
<div vocab="http://schema.org/" typeof="Person" class="col-md-12 align-items-center" style="background-color: #f0f0f0;">
    <h1 class="my-4">{{ title_name }}
    <!-- TODO category selected -->
</h1>

    <div class="d-flex flex-row hcard" style="background-color: #fff; padding: 10px; margin-top: 10px;">
        <div class="p-2">
            
        </div>
        <div class="p-2">
            <h2 class="name">{{director_name}}</h2>
        </div>
    </div>

    <div class="d-flex flex-row" style="background-color: #fff; padding: 10px; margin-top: 10px;">
        <p property="description" align="justify">
            &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
        </p>
    </div>

    <p typeof="Movie" style="background-color: #fff; padding: 10px; margin-top: 10px;">
        <strong><i>Movies: </i></strong>
        <br/>
        {% for movieName in movies %}
            <a property="name" href="../../movie/{{movieName}}">{{movieName}}</a>
            <br/>
        {% endfor %}
    </p>
</div>
```

Figura 4.5: Excerto de código de "director_profile"

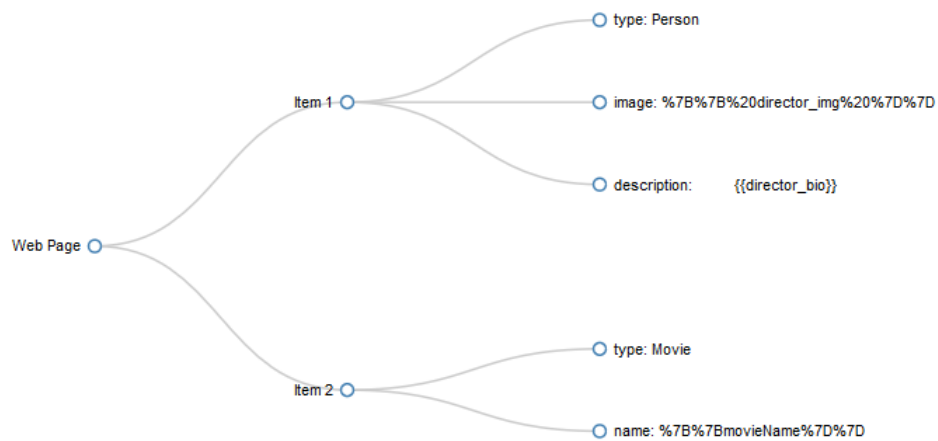


Figura 4.6: Visualização dos dados de "director_profile"

4.3 News.html

A escolha de um tipo que representasse esta página não foi trivial, no entanto optou-se por se utilizar "Article", visto que conseguia caracterizar bem a informação existente. Foram utilizadas as propriedades "name" (para o título da notícia), "author" (para o seu autor), "datePublished" (para a sua data de publicação) e "identifier" (para o *link* onde foi publicada a notícia).

```
<div vocab="http://schema.org/" typeof="Article" class="row">
  <div class="col-md-12 align-items-center" style="...">
    <br/>
    <br/>
    <h2>
      <a property="name" href={{ one_news_dic.arc_url }}>{{ one_news_dic.itemTitle }}</a>
    </h2>
    <br/>
    By <span property="author">{{ one_news_dic.autnamestr }}</span>
    <div class="card-footer text-muted" title={{ one_news_dic.pubdate }}>
      <span property="datePublished">{{ one_news_dic.pubdate }}</span>
      <br/>
      <a href={{ one_news_dic.pubin_name }}>
        <span property="identifier">{{ one_news_dic.pubin_name }}</span></a>
      </div>
    </div>
  </div>
```

Figura 4.7: Excerto de código de "news"

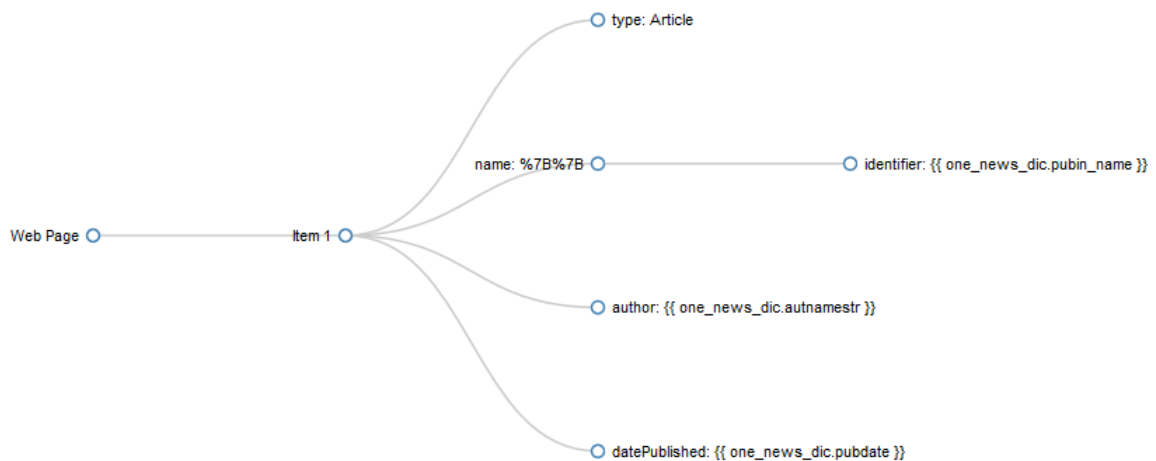


Figura 4.8: Visualização dos dados de "news"

Capítulo 5

Integração de dados da Wikidata

Com vista a alcançar um dos objetivos propostos para este trabalho, bem como pela sua utilidade e dada a facilidade de pesquisa sobre a mesma, foram integrados na aplicação final dados provenientes da Wikidata. No contexto do tema da aplicação, e tendo em conta o tipo de dados que não estão armazenados na base de dados do projeto, tomou-se a decisão de procurar dados relativos a notícias de filmes na base de dados da Wikidata.

Assim sendo, acedendo à página relativa ao item "news article", foi possível detetar quais as suas propriedades relevantes para visualização na página de notícias da aplicação. A forma utilizada para procurar apenas artigos de notícias sobre filmes foi filtrando os artigos cujo url contivesse a palavra "movies".



Figura 5.1: Excerto da página de News Article na Wikidata

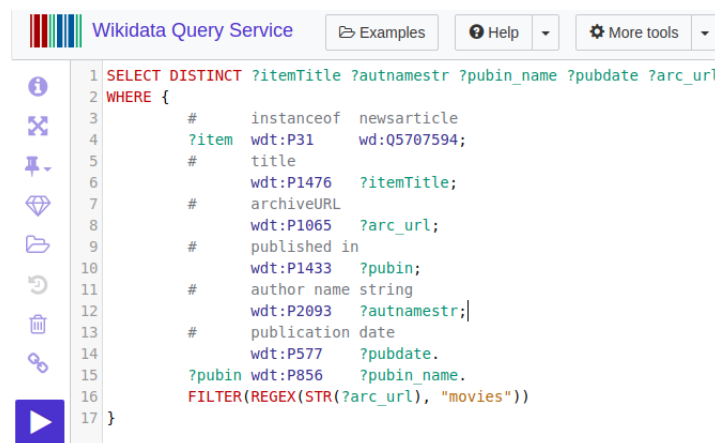


Figura 5.2: News Article query sobre os dados da Wikidata

Capítulo 6

Funcionalidades da Aplicação (UI)

Tal como referido no Capítulo 1, as funcionalidades da aplicação podem agrupar-se em quatro áreas principais, focando-se em filmes, atores, realizadores e notícias. É de referir que a navegabilidade entre estas pode ser realizada de diversas formas, pois todos os títulos de filmes e nomes de atores e realizadores redirecionam o utilizador para a sua página detalhada (caso existam na base de dados).

6.1 Filmes (*Movies*)

- Visualizar a lista de filmes existentes na base de dados;

Movies

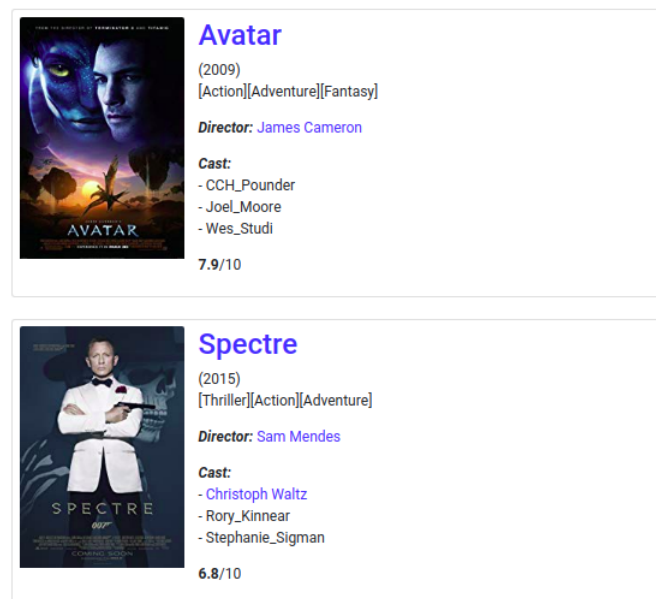


Figura 6.1: Exemplo de vista de filmes

- Pesquisar por filmes de um determinado género (*genre*), classificação (*rating*), ano de lançamento (*year*) ou qualquer combinação destes;
- Ordenar os filmes por título (*title*), ano de lançamento (*year*) ou pontuação (*score*);

Figura 6.2: Filtros e ordenação

- Pesquisar por texto, contido no título ou palavras-chave (*plot keywords*) de um filme;

Figura 6.3: Pesquisa de filmes com o termo "ocean"


- Adicionar um filme à base de dados, utilizando o botão respetivo (*New movie*);

Figura 6.4: Botão para adicionar um novo filme

- Aceder aos detalhes de um filme específico, clicando no seu título;
- Remover um filme da base de dados, utilizando o botão presente nos detalhes do filme (*Delete*).

MovieDB

Movies News Actors Directors



FROM THE DIRECTOR OF TERMINATOR 2 AND REANIMATED

AVATAR

EXPERIENCE IT IN IMAX 3D

Avatar

Year: 2009

Rating: PG-13

Language: English

Country: USA

Duration: 178 min

Budget: 237000000

Genres: [Action] [Adventure] [Fantasy]

Director: [James Cameron](#)

Cast:
Main actors: [CCH_Pounder] [Joel_Moore] [Wes_Studi]
Secondary actors: None

Plot keywords: [avatar] [future] [marine] [native] [paraplegic]

Score: 7.9/10

Delete

Figura 6.5: Página detalhada do filme "Avatar"

6.2 Atores (*Actors*)

- Visualizar todos os atores dos filmes existentes na base de dados;

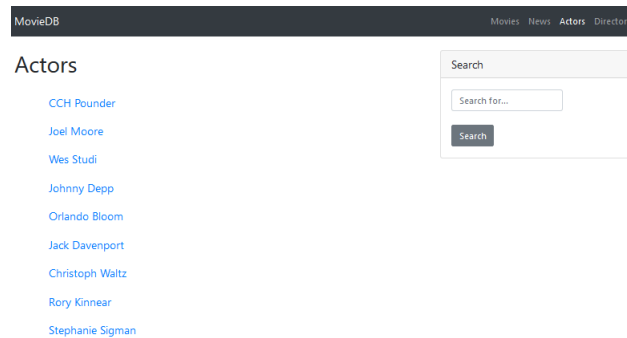


Figura 6.6: Exemplo de atores que participam em filmes existentes na base de dados

- Pesquisar atores pelo seu nome;

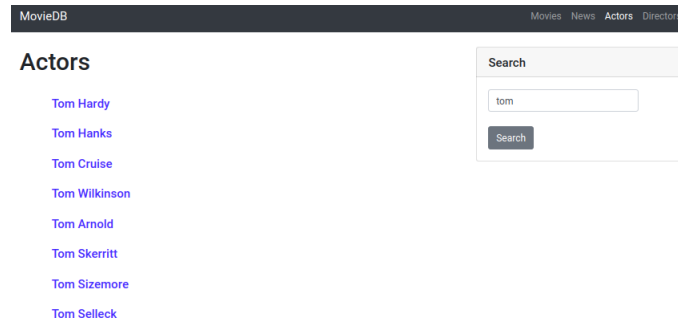


Figura 6.7: Pesquisa de atores cujo nome contém "tom"

- Aceder ao perfil de um ator (que contém a sua mini-biografia e uma fotografia), clicando no seu nome;
- Verificar, no perfil do ator, quais os filmes em que participa.

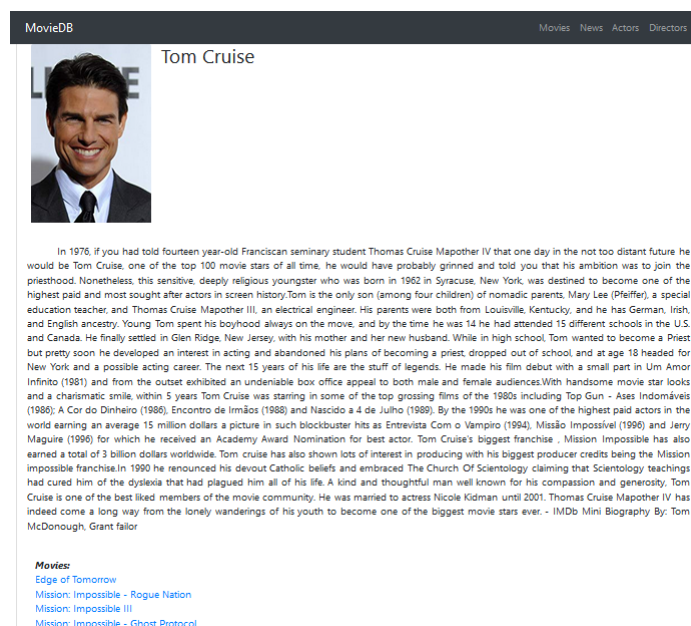


Figura 6.8: Detalhes do ator "Tom Cruise" e lista de filmes em que participou

6.3 Realizadores (*Directors*)

- Visualizar todos os realizadores dos filmes existentes na base de dados;

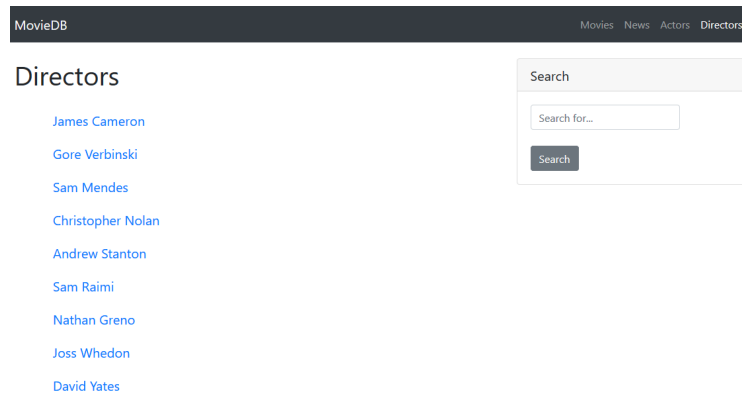


Figura 6.9: Exemplo de realizadores que dirigem filmes existentes na base de dados

- Pesquisar realizadores pelo seu nome;

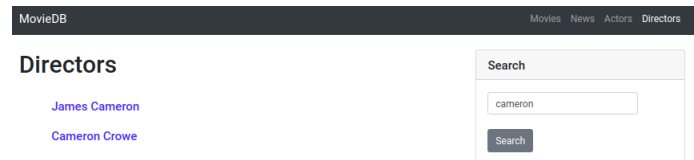


Figura 6.10: Pesquisa de realizadores cujo nome contém "cameron"

- Aceder ao perfil de um realizador (que contém a sua mini-biografia e uma fotografia), clicando no seu nome;
- Verificar, no perfil do realizador, quais os filmes que dirige.

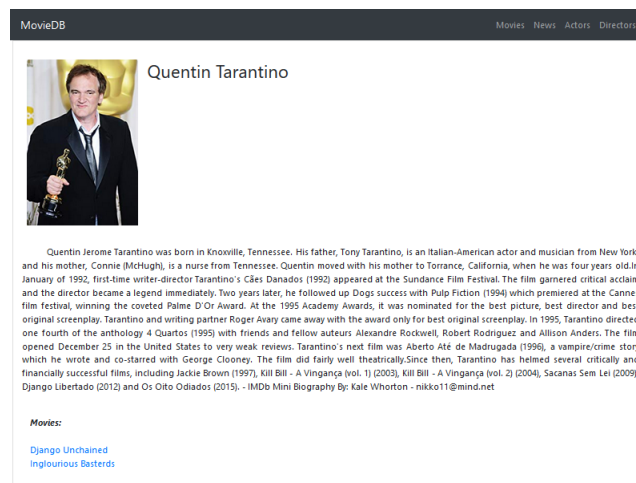


Figura 6.11: Detalhes do realizador "Quentin Tarantino" e lista de filmes que dirigiu

6.4 Notícias (*News*)

- Apresentar cabeçalhos das últimas notícias cinematográficas, com redirecionamento para a fonte destas, onde é possível consultar a notícia completa.

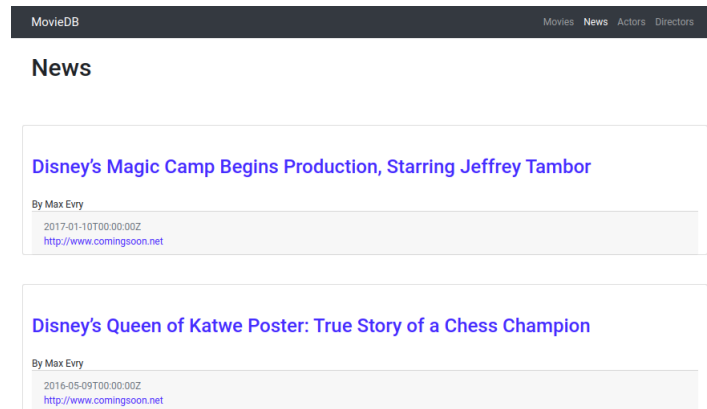


Figura 6.12: Exemplo de cabeçalhos de notícias cinematográficas

Capítulo 7

Conclusões

O projecto apresentado permitiu atingir todos os objectivos propostos, pois foi dado um propósito real para todas as tecnologias propostas e utilizadas.

Foi desenvolvido em grande parte o conhecimento na área da informação e da organização e representação de dados, um ponto fundamental neste e em qualquer projeto na área das tecnologias de informação.

Em suma, a aplicação desenvolvida permite um fácil e intuitivo acesso às principais informações sobre filmes, atores e realizadores, através de boa organização, apresentação de dados e navegabilidade.

Capítulo 8

Configuração para Executar a Aplicação

1. Através da aplicação GraphDB, criar uma base de dados com o *Repository ID* "moviesDB", *Base URL* "http://moviesDB.com" e restantes definições pré-definidas, e importar o ficheiro "movies.n3" (ou, no caso de se desejar utilizar uma base de dados menor, para acelerar as operações e consequentemente a navegação pelo *website*, importar o ficheiro "movies_short.n3"), que se encontra em "webproj\app\data\n3";
2. Instalar as dependências do projecto listadas com a respectiva versão no ficheiro requirements.txt (no directório "webproj") - através do PyCharm ou utilizando o comando `"pip install -r requirements.txt"`.
3. Inicializar a aplicação em si, através do PyCharm, e aceder a `http://127.0.0.1:8000/` no browser.

Acrónimos

CSV Coma Separated Values

XML Extensible Markup Language

RDF Resource Description Framework

N3 Notation 3

RDFa Resource Description Framework Attributes

SPARQL Simple Protocol and RDF Query Language

URI Uniform Resource Identifier

HTML HyperText Markup Language

Bibliografia

- [1] Kaggle (<https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset>)
- [2] RDFa 1.1 Primer - Third Edition (<https://www.w3.org/TR/rdfa-primer/#introduction>)
- [3] Wikidata (<http://www.wikidata.org>)