

# Trabalho Prático 1

Departamento de Eletrónica, Telecomunicações e Informática  
Universidade de Aveiro  
Web Semântica 2022

Filipe Gonçalves  
98083

Gonçalo Machado  
98359

João Borges  
98155

April 10, 2023

## 1 Introdução

A World Wide Web revolucionou a maneira como as pessoas comunicam e partilham informação entre elas. No entanto, a enorme quantidade disponível na Web deu origem a vários desafios, não só na sua gestão, mas também endender esta informação.

Para tentar resolver estes desafios, o campo da Web Semântica (WS) emergiu como uma forma de fazer a Web mais organizada e com mais significado. A WS envolve o uso de standards e tecnologias que dão estrutura e significado aos recursos da web, tornando mais eficiente a pesquisa, partilha e análise da informatica.

Neste trabalho prático, onde nos foi pedido para desenvolvermos um sistema de informação baseado na web, escolhemos utilizar informação sobre **anime**.

Anime, palavra japonesa derivada da palavra inglesa *animation*, é um tipo de animação feita à mão ou gerada por computadores com origem japonesa. Este estilo de animação tem vindo a ganhar cada vez mais popularidade ao longo do tempo, sendo que nos últimos anos, as grandes plataformas de streaming como Netflix e Disney+ aumentaram em larga escala a sua oferta de anime.

No desenvolvimento deste trabalho, foi utilizado:

- **Python/Django** - Programação da aplicação
- **RDF/NT/N3** - Formatação dos dados
- **Triplestore GraphDB** - Repositório de dados
- **SPARQL** - Pesquisa e alteração de dados na triplestore

Durante o desenvolvimento deste sistema, tivemos como objetivos desenvolver um sistema:

- Com o maior nível possível de exploração e inter-relação entre as tecnologias mencionadas previamente
- Com uma interface fácil e intuitiva para o utilizador
- Com um dataset rico e com dados bastante relacionados entre si, assim como a exploração destas relações
- Desenvolvido modularmente, separando os dados, a lógica e a apresentação

## 2 Dados, suas fontes e transformação

### 2.1 Dados

O dataset que utilizamos foi retirado do Kaggle[3] e contempla mais de 30 colunas com informação diferente de cada anime. No entanto, e por muito interessante que todas esta informação seja importante, nem toda é necessária para o nosso caso de uso.

O nosso objetivo com este dataset é termos boas ligações entre os dados de modo a pudermos usufruir de todas as vantagens do formato RDF.

De maneira muito simplista, apresentamos a seguir as colunas que achamos principais e fundamentais para o nosso uso:

Title	Characters	Voice Actors	Adaptation	Sequel
Openings	Opening Artists	Endings	Ending Artists	Prequel

Existe mais informação que não está na lista anterior, mas é apresentada no dataset como atributos para as entidades que mais tarde falaremos.

### 2.2 Fontes

O dataset, como foi dito anteriormente, foi retirado do Kaggle[5], porém os dados que contém foram retirados do MyAnimeList[1](MAL).

MAL é a maior base de dados e comunidade de Anime e Manga mundial, com mais de 4.4 milhões de **animes** inseridos e 775 000 **mangas** inseridos em 2008, e com mais de 120 milhões de visitas por mês em 2015 [7].

De acordo com isto, sabemos que os nossos dados são legítimos e confiáveis.

## 2.3 Transformação

De modo a usarmos estes dados em formato RDF, tivemos de transformar o ficheiro .csv do dataset e criar tuplos de acordo com cada entidade, predicado e objeto.

Assim, criamos 5 tipos diferentes de entidades, com diversos atributos e predicados:

- Anime

Predicado	Objeto
adapted_from	Adaptation
age_rating	Age Rating
demographic	Demographic
duration	Duration
genre	Genres
made_by	Studios
num_episodes	Number of Episodes
num_members	Number of Members
popularity	Popularity
rank	Rank
score	Score
source	Source
status	Status
title	Title
type	Type
website	Website
premiered	Premiered
ending	Entidade - Ending Song
opening	Entidade - Opening Song
sequel	Entidade - Sequel Anime
prequel	Entidade - Prequel Anime
starring	Entidade - Characters
voiced_at	Entidade - Voice Actors

- Voice Actors

Predicado	Objeto
name	Name
played	Entidade - Characters

- Characters

Predicado	Objeto
name	Name
role	Role

- Openings e Endings

Predicado	Objeto
name	Name
played_by	Entidade - Opening/Ending Artist

- Opening Artists e Ending Artists

Predicado	Objeto
name	Name

Depois de criadas as entidades e os seus atributos, decidimos que devíamos guardar em formato N-Triples para testes e para fazermos debugging do nosso frontend e queries do SPARQL.

Por fim, e para termos um controlo dos dados mais "profissional" decidimos transformar os nossos dados em N3, usando a livreria para o python rdflib[6]:

```
g = Graph()
g.parse('animes.nt')

pred = Namespace("http://anin3/pred/")
ent = Namespace("http://anin3/ent/")

g.bind("pred", pred)
g.bind("ent", ent)

n3 = g.serialize(format='n3')

with open("animes.n3", "w") as f:
    f.write(n3)
```

Os prefixos usados foram **http://anin3/ent/** para as entidades e **http://anin3/pred/** para os nossos predicados.

Ambas estas transformações dos dados pode ser vista no ficheiro **dataset.py**.

Deste modo, conseguimos ver o grafo resultante de um **anime** com valores base na Figura 6:

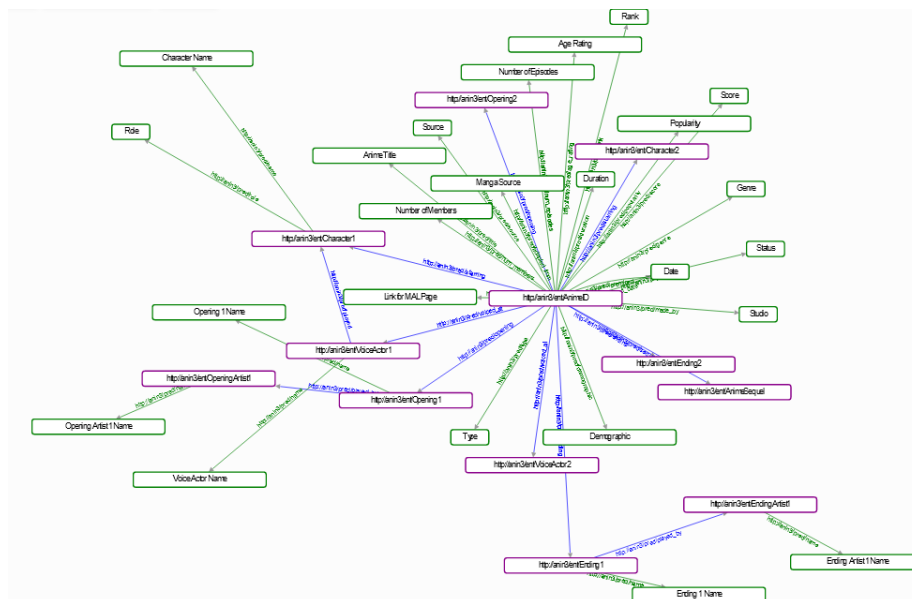


Figure 1: Grafo de um anime com valores base

Cada retângulo verde é um valor literal, cada retângulo roxo é uma entidade diferente, cada linha azul é uma propriedade do objeto e cada linha verde é uma propriedade de dados.

### 3 Operações sobre os dados - SPARQL

De modo a pudermos mostrar aos nossos utilizadores os nossos dados, executamos diferentes queries, criadas em SPARQL, no nosso repositório de GraphDB:

- Anime por Título

De modo ao utilizador poder ver a informação que quiser de um anime, criamos uma queries, que dado o título do anime, nos retorna todos os triplos relacionados ao anime.

```
PREFIX ent: <http://anin3/ent/>
PREFIX pred: <http://anin3/pred/>
SELECT *
WHERE {
  {
    ?anime pred:title "{title}" .
    ?anime ?pred ?object .
    FILTER (isliteral(?object))
  }
  UNION
  {
    ?anime pred:title "{title}" .
    ?anime ?pred ?object .
    ?object pred:name ?charname .
    ?object pred:role ?charrole .
    ?vc pred:played ?object .
    ?vc pred:name ?vcname .
  }
  UNION
  {
    ?anime pred:title "{title}" .
    ?anime ?pred ?object .
    ?object pred:name ?opname .
    ?object pred:played_by ?op .
    ?op pred:name ?opa .
  }
}
```

As duas uniões que fazemos vem de encontro com todas as relações importantes que cada anime tem:

- Todos os predicados com objetos que são literais (Exemplo: Rank, Score, Link, Título, ...)
- Todas as personagens (Characters) com os seus nomes, roles e os seus Voice Actors, com os seus nomes
- Todas as músicas de abertura/fecho com os seus nomes e artistas, com os seus nomes, respetivamente.

- Anime por Rank

Muito semelhante à querie anterior, mas com uma pequena diferença na primeira parte da query, que em vez de usarmos o título, usamos o rank:

```
{
  ?anime pred:rank "{rank}" .
  ?anime ?pred ?object .
  FILTER (isliteral(?object))
}
```

"isliteral" é uma função do SPARQL que verifica se o argumento de entrada é um literal ou não, e neste caso, igualmente como anteriormente, estamos a verificar todas as relações em que os objetos são literais.

- Pesquisa por Nome

Caso o utilizador queira utilizar a search bar para poder pesquisar por um anime / character / voice actor, criamos uma query que faz exatamente isso:

```
PREFIX ent: <http://anin3/ent/>
PREFIX pred: <http://anin3/pred/>
SELECT ?charname ?title ?vname
WHERE {
  {
    ?s pred:title ?title .
    FILTER (contains(?title, "{text}"))
  }
  UNION
  {
    ?s pred:starring ?character .
    ?character pred:name ?charname .
    ?s pred:title ?title .
    FILTER (contains(?charname, "{text}"))
  }
  UNION
  {
    ?s pred:voiced_at ?vc .
    ?vc pred:name ?vname .
    ?s pred:title ?title .
    FILTER (contains(?vname, "{text}"))
  }
}
```

"contains" é uma função do SPARQL que verifica se a informação que está no segundo argumento de entrada está inserido no objeto como primeiro argumento de entrada.



- 10 Melhores Animes

De modo a encontrarmos a informação dos 10 melhores animes (melhores em termos de rank, quanto menor o rank, melhor o anime) para mostrar ao utilizador, criamos também uma query:

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX pred:<http://anin3/pred/>
PREFIX ent:<http://anin3/ent/>
SELECT ?title ?rk
WHERE {
    ?anime pred:rank ?rk .
    ?anime pred:title ?title .
    FILTER ( xsd:integer(?rk) < xsd:integer("11") )
} LIMIT 10
```

"xsd" é um predicado real [4] que nos traz novas "funções", tais como a que utilizamos: "integer", transforma o argumento de entrada num valor inteiro. Com isto podemos filtrar os 10 melhores animes, apenas verificando se o seu rank é menor que 11. Usamos também, "LIMIT 10", por acharmos mais seguro, já que o dataset tem alguns problemas.

- Generos de Anime

Caso o utilizador queira ver os todos os animes que sejam de um específico tema, primeiramente precisamos de conhecer todos os diferentes temas, e para tal, criamos uma query com esse objetivo.

```
PREFIX ent: <http://anin3/ent/>
PREFIX pred: <http://anin3/pred/>
SELECT DISTINCT ?genres
WHERE {
    {
        ?s pred:theme ?genres .
    }
    UNION
    {
        ?s pred:genre ?genres .
    }
}
```

Neste caso, nós estamos a juntar tudo o que é "theme" e "genre", porque achamos que para o utilizador não deveria haver diferença entre os dois.

- Animes por Genero

Caso o utilizador queira ver os animes que sejam de um específico tema, criamos uma query para obtermos esse resultado:

```
PREFIX ent: <http://anin3/ent/>
PREFIX pred: <http://anin3/pred/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?anime ?rank
WHERE {
  {
    ?anime pred:theme "{genre}" .
  }
  UNION
  {
    ?anime pred:genre "{genre}" .
  }
  {
    ?anime pred:rank ?rank .
  }
} ORDER BY ASC(xsd:integer(?rank)) LIMIT 20
```

Neste caso, usamos tanto a Interseção como a União de queries para encontrarmos os ranks de todos os animes que sejam daquele "genre" ou "theme" (a razão para a junção de dados foi explicada em cima), e vamos apenas retornar os 20 melhores animes, usando o ORDER BY de forma Ascendente pelo rank dos animes.

- Voice Actor por Nome

Para o utilizador puder visualizar todos os personagens que um Voice Actor fez, criamos a seguinte query que retorna o resultado em questão tendo como argumento o nome do Voice Actor.

```
PREFIX ent: <http://anin3/ent/>
PREFIX pred: <http://anin3/pred/>
SELECT ?character_name ?role ?animename
WHERE {{
  ?voice_actor pred:name "{nome}" .
  ?voice_actor pred:played ?character .
  ?character pred:name ?character_name .
  ?character pred:role ?role .
  ?anime pred:starring ?character .
  ?anime pred:title ?animename .
}}
```

Nesta query, utilizando o nome do Voice Actor, obtemos o seu URI, que depois é usado para obter todas as personagens cuja voz pertence ao Voice Actor. Depois vamos buscar informações sobre as personagens, nomeadamente o seu nome, role e o anime a que pertencem.

- Criação de Animes

Para o utilizador poder criar um novo anime de acordo com informação que queira adicionar, criamos a seguinte query:

```
PREFIX ent: <http://anin3/ent/>
PREFIX pred: <http://anin3/pred/>
INSERT DATA
{
    ent:{identification} pred:title "{title}" ;
    pred:rank "" ;
    pred:website "" ;
    pred:score "" ;
    pred:type "" ;
    pred:num_episodes "" ;
    pred:source "" ;
    pred:status "" ;
    pred:aired_date "" ;
    pred:age_rating "" ;
    pred:popularity "" ;
    pred:num_members "" ;
    pred:made_by "" ;
    pred:duration "" ;
    pred:premiered "" ;
    pred:demographic "" ;
    pred:genre "{genre}" ;
    pred:adaptated_from "" ;
}
```

Nesta query, utilizando todos os dados que o utilizador quiser adicionar ao anime, é criado um novo anime. É de realçar que cada valor literal é um placeholder para a informação do utilizador.

Também poderia ter sido criada uma query para adicionar personagens e as demais entidades, mas achamos que esta query demonstra que é possível criarmos algo pela API que estamos a utilizar, pelo GraphDB e por Django.

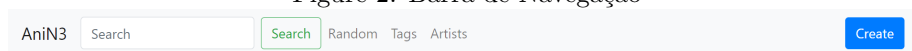
## 4 Funcionalidades da Aplicação - UI

A nossa UI de utilizador tem inúmeras páginas, em que cada uma contém informação diferente de acordo com cada query que anteriormente falamos. É de mencionar que todas as páginas utilizam a framework *Bootstrap*, pelo que são responsivas e adaptam-se a diferentes tamanhos de ecrã.

### 4.1 Barra de Navegação

Em todas as páginas está presente a barra de navegação, visível na figura 2, no topo do ecrã. Esta é constituída, da esquerda para a direita, por:

Figure 2: Barra de Navegação



- Nome do projeto - O utilizador pode clicar no nome para ser redirecionada à [Homepage](#);
- [Barra de Pesquisa](#);
- Random - Quando o utilizador clica neste botão, é redirecionada para uma página [Detalhes de anime](#) de um anime aleatório;
- [Genres](#);
- [Create](#).

## 4.2 HomePage

Figure 3: Homepage

**AniN3**  
An anime database based on N3 and GraphDB made by students in Universidade de Aveiro in context of the Web semantica classes

**Top 10 Anime**

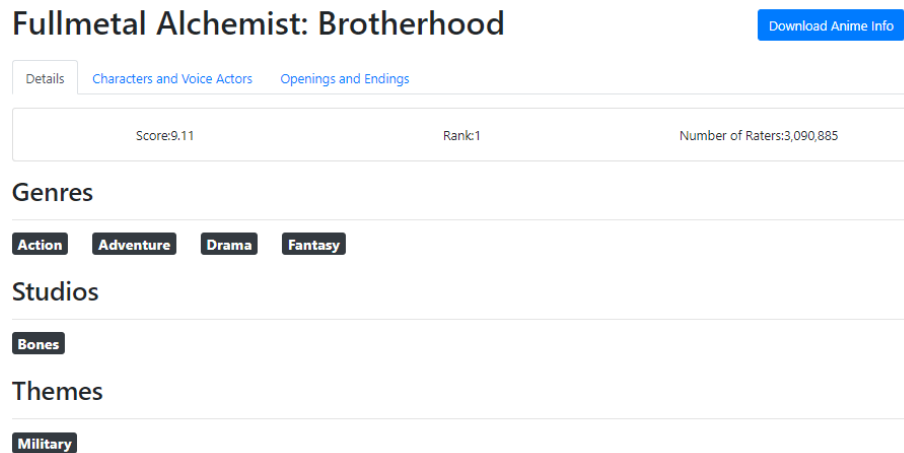
Rank: #1 Name: Fullmetal Alchemist: Brotherhood	Rank: #2 Name: Bleach: Sennen Kessen-hen	Rank: #3 Name: Steins;Gate	Rank: #4 Name: Gintama°
Rank: #5 Name: Kaguya-sama wa Kokurasetai: Ultra Romantic	Rank: #6 Name: Shingeki no Kyojin Season 3 Part 2	Rank: #7 Name: Gintama: The Final	Rank: #8 Name: Gintama'
Rank: #9 Name: Hunter x Hunter (2011)	Rank: #10 Name: Gintama': Enchousen		

A primeira página que um utilizador vê quando acede ao nosso sistema é a *Homepage*. Esta página, como é possível observar na figura 3, é constituída pelo título do sistema, seguida de uma breve descrição do sistema. Abaixo possui uma barra de pesquisa, que será mencionada e explicada com maior detalhe na sub-secção 4.6. Finalmente, estão visíveis os 10 animes com maior rank, ordenados por tal. Para obter estes dados foi utilizada a query ["10 Melhores Animes"](#).

Caso o utilizador deseje ter mais informações sobre um anime, é possível clicar no nome do anime, levando-o à página [Anime Details](#) do anime selecionado.

## 4.3 Anime Details

Figure 4: Página com detalhes gerais do anime "Fullmetal Alchemist: Brotherhood"



A página de **Detalhes de um Anime**, que se pode observar na figura 4, é usada para mostrar informações sobre um anime. No topo da página está o nome do anime cujos detalhes estão a ser vistos, assim como um botão que quando clicado transfere um ficheiro JSON com os dados do anime. A restante página possui 3 secções onde se encontra diferente informação sobre o anime, secções estas explicadas nas sub-subsecções abaixo. Para obter a informação usada nesta página, foi utilizada a query "[Animes por Título](#)"

### 4.3.1 Details

Nesta secção encontram-se detalhes gerais sobre o anime, tal como é possível ver na figura 4. Estes detalhes são:

- Score - Média das avaliações dadas por utilizadores ao anime. As avaliações podem ir de 0 a 10.
- Rank - Posição do anime em termos de **Score** entre todos os animes. No anime presente na figura 4, podemos visualizar que o anime se encontra no primeiro lugar do rank.
- Number of Raters - Número de utilizadores que avaliaram o anime.
- Genres - Todos os géneros a que o anime pertence.
- Studios - Todos os estúdios que produziram o anime.

Figure 5: Página com outros detalhes do anime "Fullmetal Alchemist: Brotherhood"

#### Other Information

<b>Type</b> TV	<b>Number of Episodes</b> 64
<b>Duration</b> 24 min per ep	<b>Source</b> Manga
<b>Aired Date</b> Apr 5, 2009 to Jul 4, 2010	<b>Status</b> Finished Airing
<b>Age Rating</b> R - 17+ (violence & profanity)	<b>Demographic</b> Shounen
<b>Adapted From</b> Fullmetal Alchemist	<b>Voiced At</b> <a href="http://anin3/ent/KugimiyaRie">http://anin3/ent/KugimiyaRie</a>
<b>Starring</b> <a href="http://anin3/ent/Greed">http://anin3/ent/Greed</a>	<b>Premiered</b>

- Themes - Todos os temas do anime.

Nesta secção possui ainda, na subsecção *Other Information*, as seguintes informações:

- Type - Tipo de anime, que pode ser **TV** (lançado na televisão), **ONA** (lançado em sites de streaming), ou **OVA** (episódios extra).
- Number of Episodes - Número de episódios.
- Duration - Duração de um episódio.
- Source - Origem do anime.
- Aired Date - Intervalo de tempo que o anime foi lançado.
- Status - Estado do anime, pode ser **Airing** (em lançamento), **Finished Airing** (lançamento acabado), etc.
- Age Rating - Idade mínima recomendada para assistir ao anime.
- Demographic - Demografia alvo do anime, pode ser por exemplo **Shounen** que são rapazes adolescentes, **Seinen** que são jovens homens adultos, etc.
- Adapted From - Nome do material de onde o anime foi adaptado.
- Premiered - Data de lançamento do anime.

### 4.3.2 Characters and Voice Actors

Figure 6: Página com personagens e atores de voz do anime "Fullmetal Alchemist: Brotherhood"

Fullmetal Alchemist: Brotherhood			
Details	Characters and Voice Actors	Openings and Endings	
Role: Main	Role: Main	Role: Main	Role: Main
Name: Elric, Edward Voice Actor: Park Romi	Name: Elric, Alphonse Voice Actor: Kugimiya Rie	Name: Elric, Alphonse Voice Actor: Kugimiya Rie	Name: Mustang, Roy Voice Actor: Miki Shinichiro
Role: Main	Role: Main	Role: Supporting	Role: Supporting
Name: Mustang, Roy Voice Actor: Okawa Tooru	Name: Mustang, Roy Voice Actor: Willingham Travis	Name: Mustang, Roy Voice Actor: Miki Shinichiro	Name: Mustang, Roy Voice Actor: Okawa Tooru
Role: Supporting	Role: Supporting	Role: Supporting	Role: Supporting
Name: Mustang, Roy Voice Actor: Willingham Travis	Name: Hughes, Maes Voice Actor: Fujiwara Keiji	Name: Greed Voice Actor: Nakamura Yuuichi	Name: Greed Voice Actor: Suwabe Junichi
Role: Supporting	Role: Supporting	Role: Supporting	Role: Supporting
Name: Hawkeye, Riza Voice Actor: Orikasa Fumiko	Name: Hawkeye, Riza Voice Actor: Neya Michiko	Name: Yao, Ling Voice Actor: Miyano Mamoru	Name: Armstrong, Alex Louis Voice Actor: Utsumi Kenji
Role: Main	Role: Main	Role: Supporting	Role: Supporting
Name: Rockbell, Winry Voice Actor: Takamoto Megumi	Name: Rockbell, Winry Voice Actor: Toyoguchi Megumi	Name: Rockbell, Winry Voice Actor: Takamoto Megumi	Name: Rockbell, Winry Voice Actor: Toyoguchi Megumi

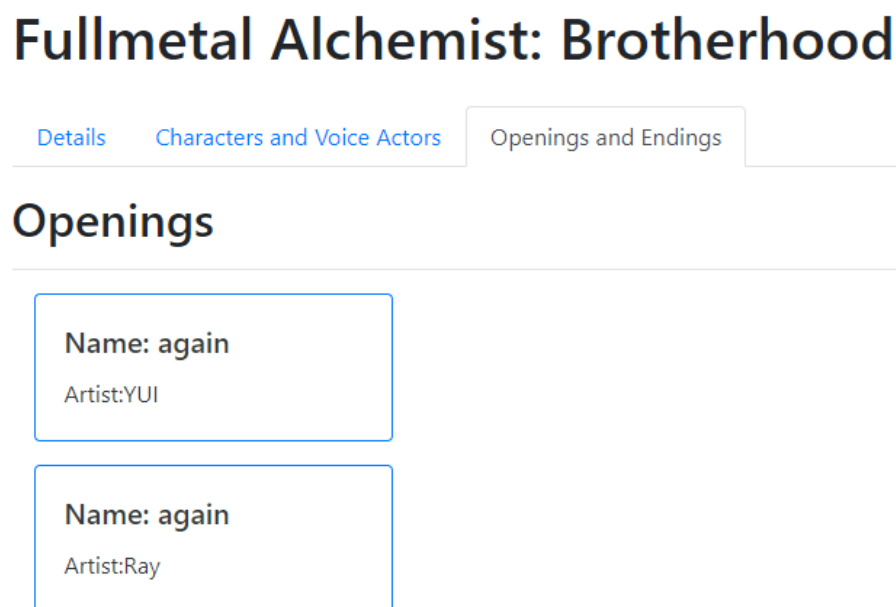
Nesta secção, que é possível visualizar na figura 6, encontram-se todos os personagens que existem no anime, assim como quem fez a voz deles. Em cada um dos cartões, está presente o nome do personagem, o nome do voice actor, e o papel do personagem no anime que pode ser "Main", ou seja **personagem principal**, ou "Supporting", ou seja **personagem secundário**.

Caso o utilizador deseje mais informações sobre um voice actor, pode clicar no nome e será redirecionado para a página [Voice Actor Details](#) do voice actor selecionado.



### 4.3.3 Openings and Endings

Figure 7: Página com as músicas de abertura e de fim do anime "Fullmetal Alchemist: Brotherhood"



Nesta secção, visível na figura 7, encontram-se todos os **Openings**, que são as músicas que são usadas nas aberturas dos episódios, e **Endings**, que são as músicas usadas no final dos episódios em conjunto com os créditos. Em cada um dos cartões encontra-se o nome da música assim como o artista a que ela pertence.

## 4.4 Genres

Figure 8: Página com todos os *genres*”

### All genres

Genre: <a href="#">Military</a>	Genre: <a href="#">Psychological</a>	Genre: <a href="#">Time Travel</a>	Genre: <a href="#">Gag Humor</a>
Genre: <a href="#">Historical</a>	Genre: <a href="#">Parody</a>	Genre: <a href="#">Samurai</a>	Genre: <a href="#">School</a>
Genre: <a href="#">Romantic Subtext</a>	Genre: <a href="#">Gore</a>	Genre: <a href="#">Survival</a>	Genre: <a href="#">Music</a>

Nesta secção, que se pode observar na figura 8, encontram-se todos os *genres* que estão disponíveis. Estes dados foram obtidos através da query ”[Animes por Género](#)”. Caso o utilizador deseje, pode clicar num género à sua escolha, e será levado à página ”[Genre Page](#)”.

## 4.5 Genre Page

Figure 9: Página do gênero **Action**”

### Genre: Action

Rank: 1 Name: Fullmetal Alchemist: Brotherhood	Rank: 2 Name: Bleach: Sennen Kessen-hen	Rank: 4 Name: Gintama°	Rank: 6 Name: Shingeki no Kyojin Season 3 Part 2
Rank: 7 Name: Gintama: The Final	Rank: 8 Name: Gintama'	Rank: 9 Name: Hunter x Hunter (2011)	Rank: 10 Name: Gintama': Enchousen
Rank: 13 Name: Gintama.	Rank: 17 Name: Gintama	Rank: 20 Name: Gintama Movie 2: Kanketsu-hen - Yorozuya yo Eien Nare	Rank: 21 Name: Code Geass: Hangyaku no Lelouch R2
Rank: 24 Name: Gintama.: Shirogane no Tamashii-hen - Kouhan-sen	Rank: 27 Name: Kimetsu no Yaiba: Yuukaku-hen	Rank: 28 Name: Gintama.: Shirogane no Tamashii-hen	Rank: 29 Name: Kingdom 3rd Season

Esta página é utilizada para visualizar animes de um **genre**, ou gênero, particular. A informação para esta página é obtida usando a query ["Animes por Género"](#). Como podemos ver na figura 9, foi escolhido o genre *Action*, e são devolvidos os animes com maior rank que são deste gênero, sendo que o utilizador pode clicar no nome de um anime à sua escolha, sendo levado à página [Anime Details](#) do anime selecionado.

## 4.6 Search Page

Figure 10: Página com resultados da pesquisa de "Park"

### Searched for: Park

#### Animes:

Name: <a href="#">Amagi Brilliant Park</a>	Name: <a href="#">Amagi Brilliant Park: Nonbirishiteiru Hima ga Nai!</a>	Name: <a href="#">Ikebukuro West Gate Park</a>	Name: <a href="#">Amagi Brilliant Park: Wakuwaku Mini Theater - Rakugaki Backstage</a>
--	--	--	--

#### Characters:

Name: <a href="#">Parker, Barbara</a> Anime: <a href="#">Little Witch Academia</a>	Name: <a href="#">Park Director</a> Anime: <a href="#">Rilakkuma to Yuuenci</a>	Name: <a href="#">Park, Hyung Suk</a> Anime: <a href="#">Oemojisangjuui</a>	Name: <a href="#">Park, Jong Gun</a> Anime: <a href="#">Oemojisangjuui</a>
Name: <a href="#">Park, Ha-Neul</a> Anime: <a href="#">Oemojisangjuui</a>	Name: <a href="#">Park, Beom-jae</a> Anime: <a href="#">Oemojisangjuui</a>	Name: <a href="#">Parkes, Suzie</a> Anime: <a href="#">Ai no Wakakusa Monogatari</a>	Name: <a href="#">Parker, Angelo</a> Anime: <a href="#">Arte</a>
Name: <a href="#">Parks, Balgo Ywain</a> Anime: <a href="#">Burn the Witch</a>	Name: <a href="#">Park, Il-Pyo</a> Anime: <a href="#">The God of High School</a>	Name: <a href="#">Park, Mu-Jin</a> Anime: <a href="#">The God of High School</a>	Name: <a href="#">Park, Yu</a> Anime: <a href="#">Hikari to Mizu no Daphne</a>
Name: <a href="#">Park, Yu</a> Anime: <a href="#">Hikari to Mizu no Daphne Specials</a>	Name: <a href="#">Park, Bom</a> Anime: <a href="#">Hate You</a>	Name: <a href="#">Park, Sandara</a> Anime: <a href="#">Hate You</a>	

#### Voice Actors:

Name: <a href="#">Park Romi</a> Anime: <a href="#">Fullmetal Alchemist: Brotherhood</a>	Name: <a href="#">Park Romi</a> Anime: <a href="#">Bleach: Sennen Kessen-hen</a>	Name: <a href="#">Park Romi</a> Anime: <a href="#">Bleach</a>	Name: <a href="#">Park Romi</a> Anime: <a href="#">Shingeki no Kyojin Season 3 Part 2</a>
--	---	--	--

Esta página é usada para mostrar os resultados de uma pesquisa usando a **barra de pesquisa** que se na "[Barra de Navegação](#)" ou na "[Homepage](#)". A informação da página é obtida através da query "[Pesquisa por Nome](#)". Como podemos ver na figura 10, após a pesquisa do nome *Park*, foram obtidos animes, personagens e voice actors que têm *Park* no seu nome, quer seja parcialmente ou totalmente.

## 4.7 Voice Actor Details

Esta página é usada para mostrar todos os personagens cuja voz foi feita por um **Voice Actor**, ou VA. Tal como é possível ver na figura 11, no topo da página encontra-se o nome do VA, sendo a restante página composta pelas personagens feitas pelo VA. No cartão é possível ver o nome da personagem, o anime a que pertence assim como o papel que teve no anime.


Figure 11: Página do Voice Actor "Park Romi"

### VA: Park Romi

Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: Brotherhood	Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist	Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: The Conqueror of Shamballa	Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: Brotherhood Specials
Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: Brotherhood - 4-Koma Theater	Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: Premium Collection	Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: The Sacred Star of Milos	Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: Reflections
Role: Main <b>Name: Elric, Edward</b> Fullmetal Alchemist: The Sacred Star of Milos Specials	Role: Main <b>Name: Hitsugaya, Toshiro</b> Bleach: Sennen Kessen-hen	Role: Main <b>Name: Hitsugaya, Toshiro</b> Bleach	Role: Main <b>Name: Hitsugaya, Toshiro</b> Bleach Movie 4: Jigoku-hen
Role: Main <b>Name: Hitsugaya, Toshiro</b> Bleach Movie 3: Fade to Black - Kimi no Na wo Yobu	Role: Main <b>Name: Hitsugaya, Toshiro</b> Bleach Movie 1: Memories of Nobody	Role: Main <b>Name: Hitsugaya, Toshiro</b> Bleach Movie 2: The Diamond Dust Rebellion - Mou Hitotsu no Hyourinmaru	Role: Main <b>Name: Hitsugaya, Toshiro</b> Bleach: The Sealed Sword Frenzy

## 4.8 Create

Figure 12: Página para adicionar um novo anime



The form is titled "Add Anime". It contains three input fields: "Title", "Genre", and "Score". Below the "Title" field is a blue button labeled "Add Anime".

Title	Genre	Score
<input type="text"/>	<input type="text"/>	<input type="text"/>

Esta página, que podemos ver na figura 12, é utilizada para inserir novos animes no dataset. O utilizador apenas necessita de inserir o **Nome** do anime, o **Gênero** do anime e a **Score** do anime e clicar no botão **Add Anime**. Para fazer a inserção dos dados, utilizamos a query "[Criação de Animes](#)". Após a inserção, o anime inserido pelo utilizador estará disponível para visualização.

## 5 Conclusões

Em conclusão, queremos dizer que foi um projeto interessante e com ferramentas que nunca tínhamos utilizado, o que tornou o nosso trabalho mais apelativo e que nos deu mais vontade de perceber e fazer.

Olhando para o que fizemos, podemos dizer que na transformação dos dados, por muito que tenhamos usado tanto N-Triples como N3, devíamos ter ido mais fundo na arquitetura de prefixos e das relações entre as entidades.

Também é de notar que ao usarmos diferentes tipos de queries, algumas de complexidade simples e outras mais avançadas, conseguimos manusear e apresentar os dados de uma maneira simples, mas efetiva.

A nossa UI, embora tenha um design simplista, é bastante intuitiva e demonstra de maneiras diferentes todas as relações entre os dados.

Com tudo isto, diríamos que este trabalho cumpre todos os requisitos pedidos para o sistema, e achamos que o nosso sistema mostra, de uma forma clara e intuitiva, todos os dados que construímos de uma muito boa forma.

Pode encontrar o nosso repositório de github, com o código do projeto em [FlipGoncalves/WebSemanticaTrabalho1](#)[2].

## 6 Configurações para executar a aplicação

Para executar a aplicação, deve seguir os seguintes passos:

- GraphDB

Iniciar o GraphDB no seu computador em `http://localhost:7200/` e certificar-se que não tem um repositório chamado "anin3" inicialmente.

- Django Application

Para iniciar a aplicação django, basta executar os seguintes comandos:

```
cd path/to/project/folder/  
cd WSPProject1/  
python3 manage.py runserver
```

De maneira a automatizar a utilização da nossa aplicação, a criação do repositório e posterior inserção de dados neste é feita automaticamente caso o repositório ainda não tenha sido criado. O código que foi escrito para automatizar este processo encontra-se no ficheiro *apps.py*, localizado na pasta *WebSemanticaTrabalho1/WSPProject1/app*, e é corrido uma vez quando a aplicação é inicializada. O funcionamento do código é o seguinte:

- O sistema envia um GET request ao GraphDB para verificar se o repositório já existe. Se o repositório já existir, é utilizado o repositório existente, não havendo assim problemas de terminar e inicializar a aplicação várias vezes.
- No caso do repositório não existir, é necessário ir buscar os caminhos absolutos para o ficheiro de configuração do repositório, **anin3-config.ttl**, e para o ficheiro com os dados que serão inseridos no repositório, **animés.nt**.
- O caminho absoluto do ficheiro com os dados é escrito no ficheiro de configuração.
- É feito um POST request ao GraphDB com o ficheiro de configuração como argumento, sendo este o request que cria e insere os dados automaticamente.



## References

- [1] MyAnimeList Co. *MyAnimeList*. URL: <https://myanimelist.net/>.
- [2] João Borges (98155) Filipe Gonçalves (98083) Gonçalo Machado (98359). *WebSemanticaTrabalho1*. URL: <https://github.com/FlipGoncalves/WebSemanticaTrabalho1>.
- [3] Kaggle INC. *Kaggle: Your Machine Learning and Data Science Community*. URL: <https://www.kaggle.com/>.
- [4] Jeff Z. Pan Jeremy J. Carroll. *XML Schema Datatypes in RDF and OWL*. URL: <https://www.w3.org/TR/swbp-xsch-datatypes/>.
- [5] Jerry Kim. *Anime2023February*. URL: [https://www.kaggle.com/datasets/gaeranbab/anime2023february24000?select=24000\\_anime\\_stats\\_022123.csv](https://www.kaggle.com/datasets/gaeranbab/anime2023february24000?select=24000_anime_stats_022123.csv).
- [6] RDFLib Team. *rdflib 6.3.2*. URL: <https://rdflib.readthedocs.io/en/stable/>.
- [7] Wikipedia. *MyAnimeList - Wikipedia*. URL: <https://en.wikipedia.org/wiki/MyAnimeList>.