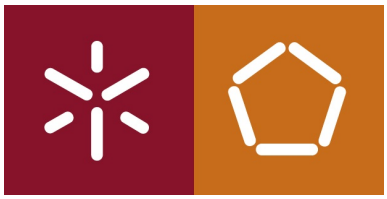

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Processamento e Representação de Conhecimento

AnimeList

Francisco José Moreira Oliveira, a78416

14 de Junho de 2019

Resumo

O presente trabalho foi desenvolvido no âmbito da unidade curricular Processamento e Representação de Conhecimento do 4º ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho, sendo que esta tem como objetivo a aprendizagem prática de conceitos dos desenvolvidos associados ao perfil de Processamento de Linguagens e Conhecimento.

O trabalho consiste na escolha de um tema, na procura de um *dataset* relativo ao mesmo e no seu desenvolvimento de forma a ser possível disponibilizar a sua informação de forma fácil e acessível numa interface web interativa.

Conteúdo

Capítulo 1

Introdução

O projeto desenvolvido consiste na utilização das abordagens desenvolvidas na unidade curricular de PRC de modo que apartir de um *dataset*, cujo tema escolhido é *Anime* (animações japonesas), apresentar a informação contida no *dataset* sobre o tema de forma fácil e interativa numa interface web, com a auxilio de uma ontologia para representação de conceitos e relações desse domínio.

Deste modo, podemos distinguir alguns componentes de relevância cujo desenvolvimento consideramos fundamental, como o uso da ferramenta *Protégé* que auxiliou na criação da ontologia, o *GraphDB* que serviu como base de dados do sistema, o *NodeJS* que serviu de *backend* recolhendo a informação disponibilizada pelo *GraphDB* usando queries *SPARQL* e o *Vue* que serviu de *frontend* disponibilizando toda informação numa interface web interativa.

Assim sendo, tendo as várias componentes disponíveis é relevante entender brevemente a metodologia usada para a resolução do problema. Primeiro, vamos começar por fazer uma breve contextualização do tema escolhido, depois passamos à criação da ontologia, do tratamento do *dataset* e respetiva conversão para *turtle* e a sua importação para o *GraphDB*. Depois, por último, o desenvolvimento do *backend* e *queries* associadas e do *frontend* utilizando *Vue* para mostrar a informação.

Capítulo 2

Tema

O tema escolhido para este projeto foi *Anime*[1], que é resumidamente animação japonesa. Este tema foi escolhido por ser uma tema que achei interessante e com potencial, gosto pessoal do tema e a existência de informação na Internet. Os *Animes* são frequentemente transmitidos por televisão e em algumas ocasiões por CD/Blu-ray, tem vários géneros, produtores e estúdios, de forma muito semelhante a filmes.

Dada toda a variedade existente, *Anime* contem uma grande quantidade de informação tornando o volume de dados do *dataset* aceitável para o desenvolvimento do projeto.

2.1 Dataset

O *dataset* utilizado[2] foi retirado do Kaggle. Foi utilizado o `anime_cleaned.csv` pois apesar da menor quantidade de *Anime*, continua a ser um volume suficiente de dados, com o beneficio de ter uma maior densidade de informação (poucos nulos) e ter em geral informação mais correta.

Este *dataset* contem o titulo (original, inglês e japonês), géneros, *score*, estúdio, produtor, etc.

2.2 Ontologia

Com base na informação contida no *dataset* e uma breve análise da mesma, foi desenvolvida uma ontologia preparada para ser conter a informação mais útil e importante do *dataset*. Para isto utilizou-se a ferramenta utilizada nas aulas *Protégé*, e definimos a seguinte ontologia:

- Classes:

  Anime - classe que representa os *Animes*;

  Genre (género) - possíveis géneros de *Anime*;

  Producer (produtor) - produtores de *Anime*;

  Studio (estúdio) - estúdios de *Anime*;

- Relações (Object Properties):

  hasStudio - estudio que desenhou o *Anime*;

  designedBy - relação inversa de hasStudio;

[1] `https://pt.wikipedia.org/wiki/Anime`

[2] `https://www.kaggle.com/azathoth42/myanimelist#anime_cleaned.csv`

hasProducer - produtor que produziu o *Anime*;

produced - relação inversa de hasProducer;

hasGenre - genero do *Anime*;

refersTo - relação inversa de hasGenre;

- Atributos (Data Properties):

  id - id de qualquer classe;

  label - nome de qualquer classe em string;

  title - titulo do *Anime*;

  title_english - titulo em inglês do *Anime*;

  title_japanese - titulo em japonês do *Anime*;

  img - URL para uma imagem do *Anime*;

  type - tipo do *Anime*;

  score - avaliação do *Anime*;

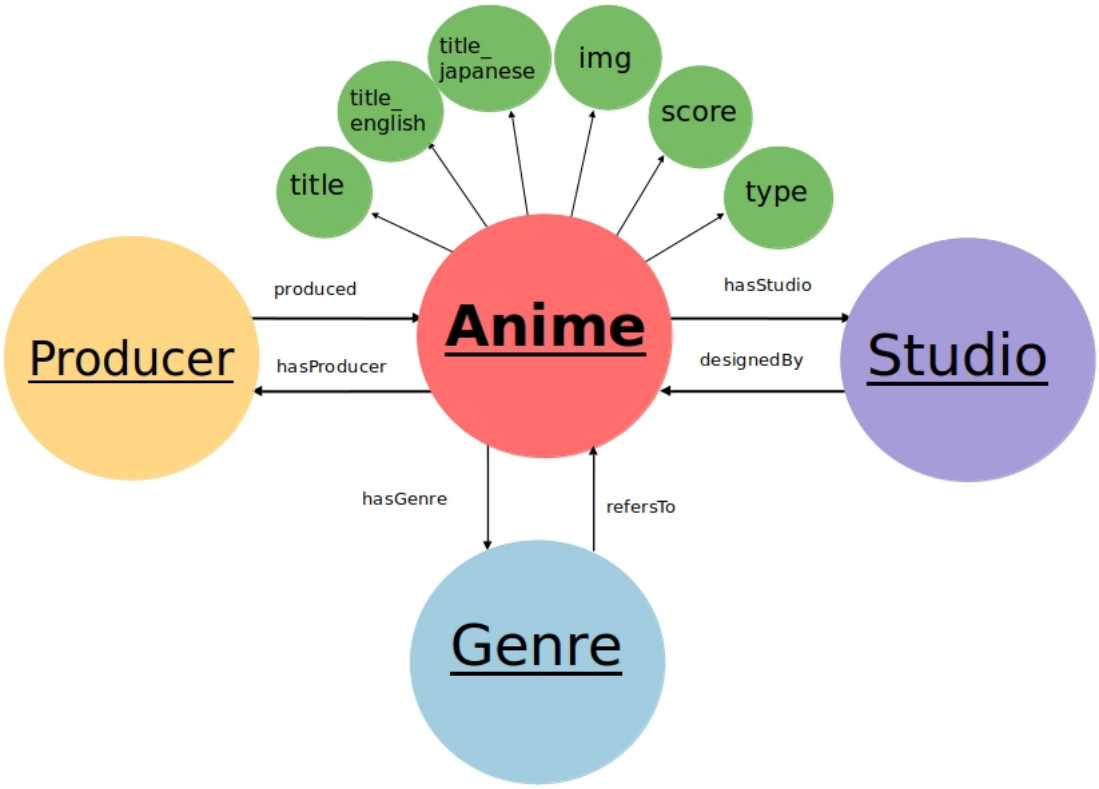

Figura 2.1: Grafo da ontologia criada.

Após ter a ontologia criada, foi possível então passar a fase seguinte do tratamento dos dados.

2.3 Tratamento de Dados

Para tratar dos dados e os juntar à ontologia criada desenvolveu-se um *script* em *Python* para tratar da informação. A metodologia usada foi percorrer o *dataset* linha a linha, retirando a informação das colunas pretendidas e depois gerar um *individual*, em *turtle*, com a informação recolhida. Depois de gerar todos os *individuals* corretamente, concatenou-se o resultado com o ficheiro das ontologias e criou-se assim o ficheiro do `povoamento.ttl` que irá ser importado no *GraphDB*.

Na figura 2.2 temos um exemplo de um visão do *Anime*'Fullmetal Alchemist: Brotherhood' e as suas relações com outras classes (genre, producer, studio).

Figura 2.2: Visual Graph de 'Fullmetal Alchemist: Brotherhood' no GraphDB.

2.4 Backend

Para o *backend* utilizou-se um servidor *NodeJS* que vai receber os pedidos efetuados pelo *frontend* aos quais vai devolver a resposta retirada do *GraphDB*.

Para responder a estes pedidos o servidor *NodeJS* vai buscar a informação ao *GraphDB* usando *queries*,

que foram previamente feitas, capazes de satisfazer os pedidos necessários. Os pedidos enviados pelo *NodeJS* seguem a API do *GraphDB* que retornam a sua resposta em JSON que é passada depois como resposta para o *frontend*.

De modo a facilitar a compreensão foram divididas as *queries* em diferentes ficheiros devido às diferenças entre elas. Há 3 ficheiros para 3 tipos de *queries*:

- **sparqlQuerys** *queries* sem argumentos que apenas são chamadas e devolvem resposta
- **sparqlQuerys1attr** *queries* com 1 argumento, normalmente um ID, usado para completar a *query* antes de a correr
- **sparqlQuerysVariable** *queries* com numero de argumentos variaveis, tanto podem receber ou não argumentos e executar q *query*

Cada tipo de *query* é tratada de forma diferente, tendo portanto uma route diferente para cada tipo.

Encontram-se em anexo no appendice A algumas *queries* de exemplo.

A primeira *query* retorna o top5 de animes, de acordo com o seu score. Esta *query* foi utilizada para preencher a homepage.

A segunda *query* serviu para obter a informação de qualquer individuo, usando o seu ID. Esta *query* foi utilizada em várias páginas sempre que era precisa informação de algo em especifico.

2.5 Frontend

Após ter a informação carregada no *GraphDB* e o *NodeJS* pronto começou-se a tratar do *frontend*, no qual foi usado *Vue*.

No *Vue* foi criada uma homepage, uma lista com todos os *Animes*, uma lista com todos os producers, uma lista com todos os studios e uma pagina para visualização de informação individual de um studio, producer ou *Anime* Foi também criado um sistema muito simples de perfis de utilizador.

Começando pelos perfis de utilizador, que foi a ultima componente realizada e acabou por ficar incompleta, esta permite fazer *login* usando um nome de utilizador. Depois de entrar num perfil ganhamos acesso a pagina do perfil, onde podemos ver informações da conta, nomeadamente os *Animes* favoritos.

É de referir que não existe autenticação, servindo apenas este sistema de *proof of concept*, e que algumas ações não foram implementadas (adicionar e remover favoritos, adicionar e apagar perfis), contudo mostra algum potencial e a ideia por trás. Foi utilizado um *JSON-Server* para guardar a informação dos perfis.

Passando as restantes páginas. Temos a homepage com um top5 de *Animes*. Esta página permite levar o utilizador rapidamente aos melhores *Animes* do momento. Temos a página individual de studio, producer ou *Anime* com informação detalhada referente ao mesmo. Temos a página com a lista de producers/studios, onde ambas seguem o mesmo formato, usando uma lista com search para facilitar a busca.

Por fim temos a página com a lista de *Animes*. Esta página possui não só search por texto, mas também filtros por género, produtor e estúdio, permitindo satisfazer muitas possiveis pesquisas.

Aqui temos a imagem 2.3, mostrando uma breve pesquisa, usando a search, e 2 filtros, obtendo 5 resultados que satisfazem as condições.

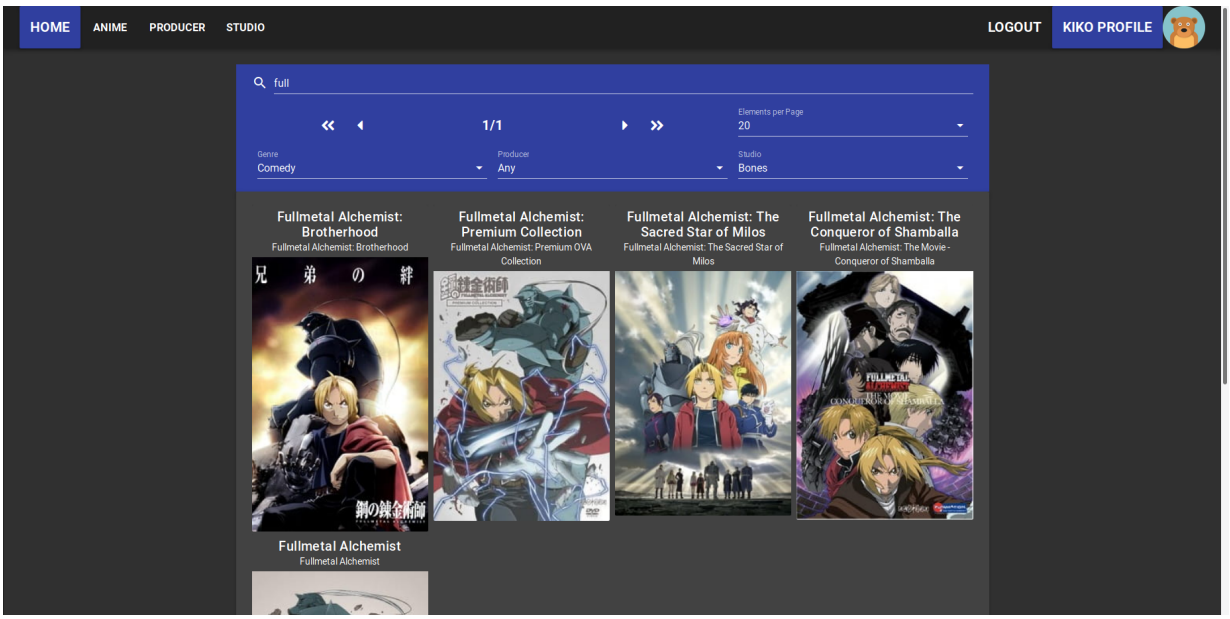

Figura 2.3: Tabela dos Animes.

2.6 Executar

Para executar todo o projeto é necessário ter as ferramentas referidas instaladas.

Estando isso feito, é necessario importar o `povoamento.ttl` para o *GraphDB* no repositório 'projetoBeta', e deixar o *GraphDB* a correr.

De seguida devem-se fazer o `npm install` na pasta backend, frontend e usersdb.

Finalmente apenas é necessario correr `npm start` na pasta backend para iniciar o *NodeJS*. Correr `npm run serve` na pasta frontend para iniciar o *Vue*. Correr `npm run json:server` na pasta usersdb para iniciar o *json-server*.

Todos estes comandos estão presentes no `makefile`.

Capítulo 3

Conclusão

O projeto detalhado neste relatório consistiu da escolha, tratamento e exploração do conhecimento presente num *dataset* relativo a *Animes*. Tal objetivo foi atingido através da criação de uma ontologia (modelo de dados que permite a representação de conhecimento) e, por fim, uma aplicação web para que o utilizador possa aceder à informação de forma simples, eficiente e intuitiva através das várias páginas, filtros e opções disponibilizadas na mesma.

A nível de ferramentas e tecnologias utilizadas recorreu-se a NodeJS, Vue e ao GraphDB o que permitiu aprimorar os conhecimentos adquiridos ao longo do semestre na unidade curricular na qual o projeto se insere, nomeadamente, Processamento e Representação de Conhecimento.

Por fim, faz-se uma apreciação positiva quanto ao trabalho desenvolvido, apesar de este ainda ter espaço para melhorias.

Apêndice A

Sparql Queries do NodeJS

```
top5animeByScore: ‘
PREFIX : <http://www.semanticweb.org/kiko/ontologies/2019/projeto#>
select distinct * where {
    ?anime a :Anime .
    ?anime :id ?id .
    ?anime :title ?title .
    ?anime :score ?score .
    ?anime :img ?img .
}
order by desc(?score)
limit 5‘

infoBy_id: function (id) {
    return ‘
PREFIX : <http://www.semanticweb.org/kiko/ontologies/2019/projeto#>
select distinct * where {
  :‘ + id + ‘ ?p ?o .
  FILTER ( ?p!=rdf:type )
}‘
  }
```