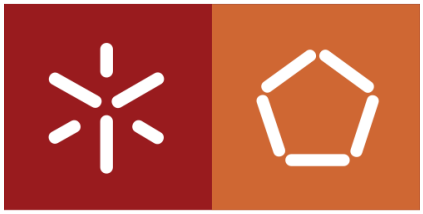

Universidade do Minho
Escola de Engenharia

Processamento de Representação de Conhecimento
Ontologia Musical - UMinhoMusic

Diego Porto PG37149
Ricardo Neves A78764
Rui Costa A79947

Junho 2020

Resumo

Neste relatório, iremos mostrar quais os passos que seguimos para o bom desenvolvimento do trabalho prático proposto. Este constitui a construção de uma ontologia musical e implementação de uma aplicação Web para apresentação dos dados ao utilizador. De salientar que esta aplicação destina-se apenas à consulta de dados da ontologia, sem possibilidade de modificações.

Conteúdo

1 Introdução

Neste trabalho prático, inserido na Unidade Curricular de Processamento e Representação de Conhecimento, foi construída uma ontologia completa e um servidor para apresentação ao utilizador de toda a informação de uma forma simples e organizada, denominado **UMinhoMusic**.

Assim, depois de uma breve discussão entre os elementos, o grupo decidiu debruçar-se sobre a categoria de músicas, reunindo várias informações acerca dos seus artistas e álbuns. Isto devido à grande quantidade de dados disponível e que, bem consolidados e moldados, podem gerar uma ontologia bastante extensa e interessante.

Posto isto, iremos apresentar, em primeiro lugar, os detalhes da ontologia utilizada para o projeto, seguida pelo raciocínio e método na extração dos dados, e, por fim, as características da aplicação Web, composta pelo *backend* e *frontend*.

2 Ontologia

Com o tema do trabalho prático já bem estabelecido, o grupo criou a ontologia na ferramenta Protégé, aconselhado pelo professor, e que se trata de um sistema gratuito de gestão de conhecimento. Com a exportação da nossa ontologia para um ficheiro .ttl, esta fica preparada para acolher os dados exportados, e que serão abordados no próximo tópico. Sendo assim, foram criadas as três classes base da ontologia: **Artista**, **Álbum** e **Música**, que irão ser detalhadas já de seguida. Nesta ontologia, todas as classes estão ligadas entre si através das *Object Properties*.

2.1 Artista

O Artista é composto pelo seu ID único, nome, popularidade (valor de 0 a 100, sendo 100 uma popularidade máxima), número de seguidores, gênero(s), *link* para a sua página Spotify e uma imagem. Um Artista produz um Álbum e toca uma Música.

2.2 Álbum

O Álbum é composto pelo seu ID único, título, número total de músicas, data de lançamento, *link* para a sua página Spotify e imagem de capa. Um Álbum é produzido por um Artista e contém Músicas.

2.3 Música

A Música é composta pelo seu ID único, título, duração, explicita (ou não) e um *link* para a sua página Spotify. Uma Música é tocada por um Artista e está inserida num Álbum.

3 Extração dos dados

Por esta altura, a base da ontologia está completa, e portanto, é necessário povoá-la. Tratando-se de uma ontologia baseada em música, o grupo pesquisou as melhores alternativas para extrair este tipo de informação, uma vez que a hipótese de inserir uma grande quantidade de dados à mão sempre foi excluída desde a escolha do tema.

Para auxílio na implementação dos *scripts* Python, foram instaladas as bibliotecas 'csv', para escrita e leitura de ficheiros .csv, e 'pandas', para remover duplicados e ordenação dos artistas por popularidade. De acrescentar que esta ordenação dos artistas por popularidade não era necessária de ser realizada nesta altura, uma vez que esta pode ser efetuada com uma *query* SPARQL.

3.1 Spotipy

Após esta breve pesquisa, o grupo encontrou a libraria Spotipy, modulada para Python, que garante acesso aos dados presentes na plataforma online Spotify. Para isto, foi necessário registar esta aplicação na API oficial do Spotify e utilizar as credenciais dadas (ID e segredo) para poder realizar as *calls* pretendidas.

A API do Spotify não oferece a possibilidade de extrair todos os dados de uma só vez, pelo que tivemos que encontrar caminhos alternativos para a extração dos mesmos. Assim, tivemos de estudar todas as *calls* possíveis à API, disponíveis no seu *website* oficial.

Em primeiro lugar, decidimos utilizar uma conta Spotify de um dos elementos do grupo e retirar os artistas mais ouvidos no último ano. Com isto, conseguimos extrair da API cerca de 80 artistas. No entanto, 80 artistas estava muito longe do número pretendido e definido pelo grupo numa fase inicial. Assim, percorremos cada um destes artistas e extraímos os seus artistas semelhantes, removendo os duplicados, chegando a um número bastante aceitável de 1036 artistas na ontologia.

Depois disto, o processo tornou-se mais simples. Por cada artista extraído, retira-se da API toda a informação dos seus álbuns publicados. No fim, por cada álbum, são extraídas as informações das músicas lá inseridas. De notar que uma música pode estar presente em mais do que um álbum do artista, mas, obviamente, com IDs diferentes.

Todos estes dados referidos anteriormente foram guardados em três ficheiros .csv.

3.2 CSV para TTL

Para a inclusão dos dados na ontologia, foi necessário transformar os ficheiros .csv em .ttl. Esta transformação foi realizada recorrendo a *scrpits* Python, especificamente implementados para cada um dos três ficheiros .csv.

Como resultado, obtivemos três ficheiros em formato .ttl que, logo de seguida, foram copiados para dentro da ontologia previamente desenhada.

4 Aplicação Web

Deste modo, passamos para a apresentação da segunda metade do projeto, que foi implementar uma aplicação Web que leia os dados e os apresente ao utilizador através de uma interface limpa e intuitiva. De referir que a ontologia completa foi carregada para a ferramenta GraphDB, também aconselhada pelo docente da cadeira. No GraphDB, podemos reparar que estão carregadas 947.772 declarações, sendo que cerca de 89% (843.394 decl.) estão explícitas na ontologia e 11% (104.378 decl.) são inferidas pelo sistema. **Nota:** Por vezes, aparecem dois álbuns aparentemente idênticos a olho nú. No entanto, tratam-se de um álbum com linguagem explícita e outro com essa linguagem censurada, ambos presentes na base de dados oficial do Spotify. Sendo assim, não se tratam de erros de extração de dados ou de *queries* SPARQL mal formuladas.

4.1 API

Primeiro, o grupo implementou a API de dados da aplicação, que permite a comunicação entre a Interface e o GraphDB, que contém todos dados da ontologia.

Os métodos implementados foram:

- **getMusicas()** - Retorna a lista de todas as músicas na ontologia.
- **getMusica(id)** - Retorna as informações da música dada como argumento.
- **getArtistas()** - Retorna a lista de todos os artistas na ontologia.
- **getArtista(id)** - Retorna as informações do artista dado como argumento.
- **getAlbums()** - Retorna a lista de todos os álbuns na ontologia.
- **getAlbum(id)** - Retorna as informações do álbum dado como argumento.
- **getAlbumsPorArtista(id)** - Retorna a lista de álbuns do artista dado como argumento.
- **getMusicasPorAlbum(id)** - Retorna a lista de músicas do álbum dado como argumento.

À medida que a implementação da Interface ia avançando, este servidor de dados foi sendo atualizado, de modo a satisfazer os pedidos provenientes do servidor da interface, que irá ser detalhado de seguida.

4.2 Interface

A aplicação Web pode ser dividida em quatro secções importantes: a página inicial e as três tabelas de artistas, álbuns e músicas disponíveis. Quando o utilizador se encontra numa destas três tabelas, o mesmo pode clicar numa entidade artista ou álbum (consoante a secção onde estiver) e ser redireccionado para a sua página individual. Também é possível ser redireccionado para a página Spotify da entidade.

Em termos de librarias, utilizamos o Vue Router, Vuex para o *store*, o Vue Resource para aceder à API, e o Vuetify para o embelezamento da aplicação.

Apresentamos agora alguns *prints* do sistema desenvolvido:

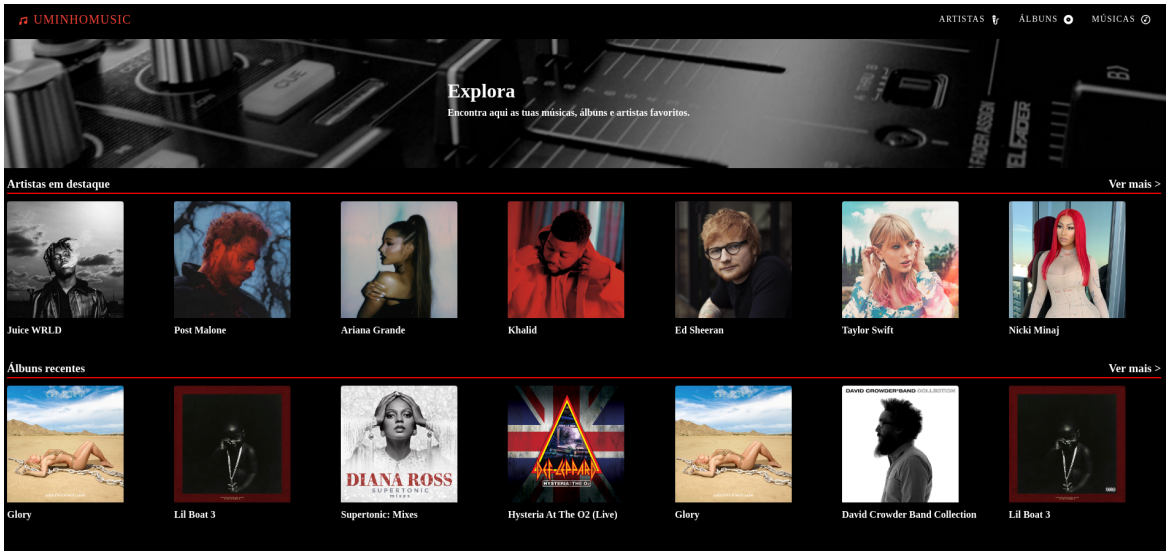

Figura 1: Homepage da plataforma UMinhoMusic

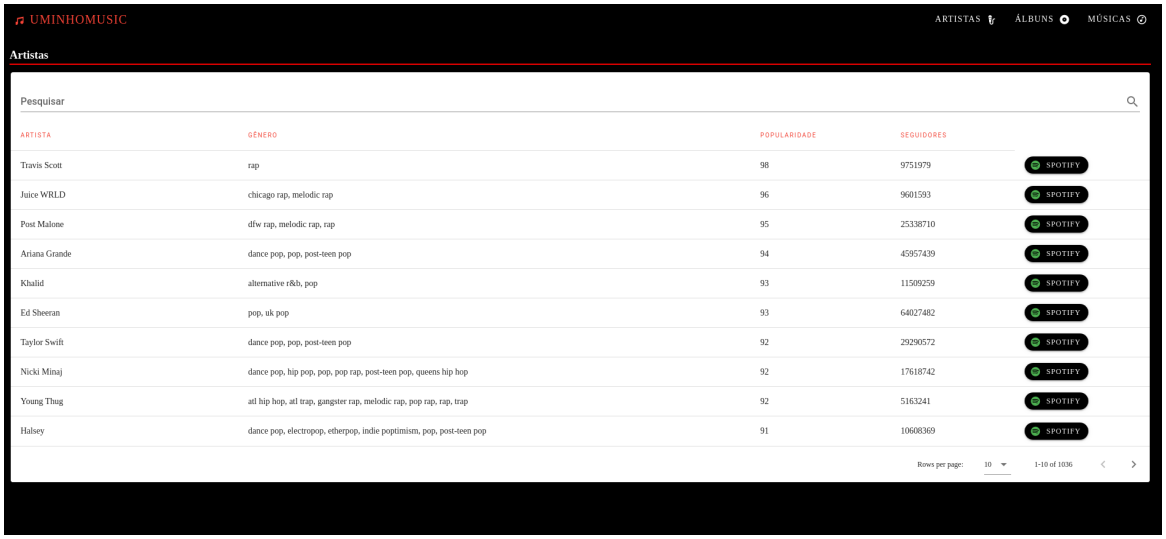

Figura 2: Secção com a tabela dos artistas

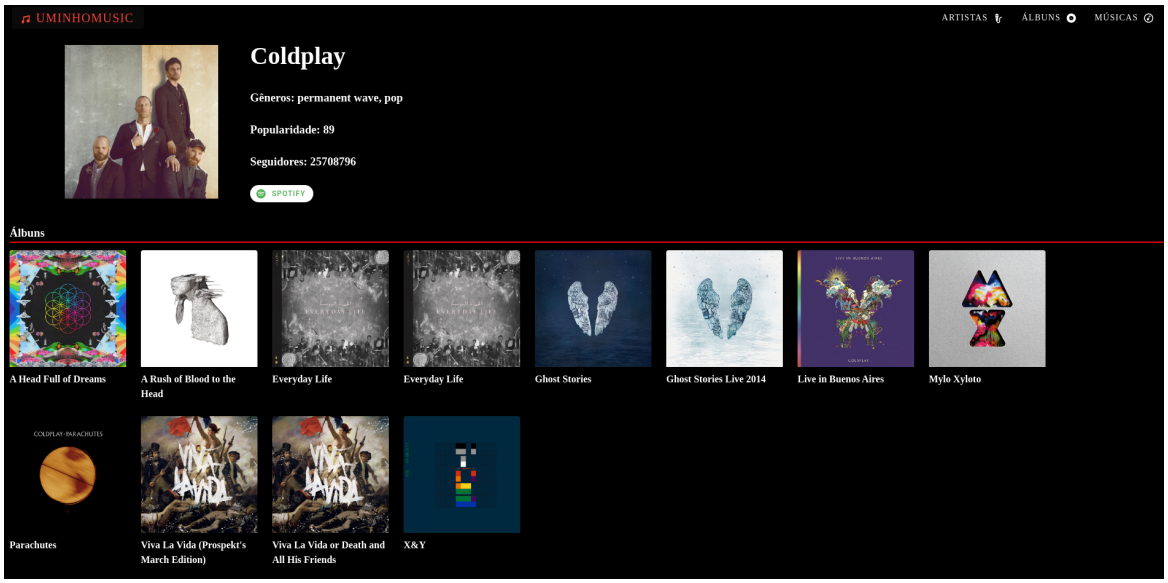

Figura 3: Página individual de um artista

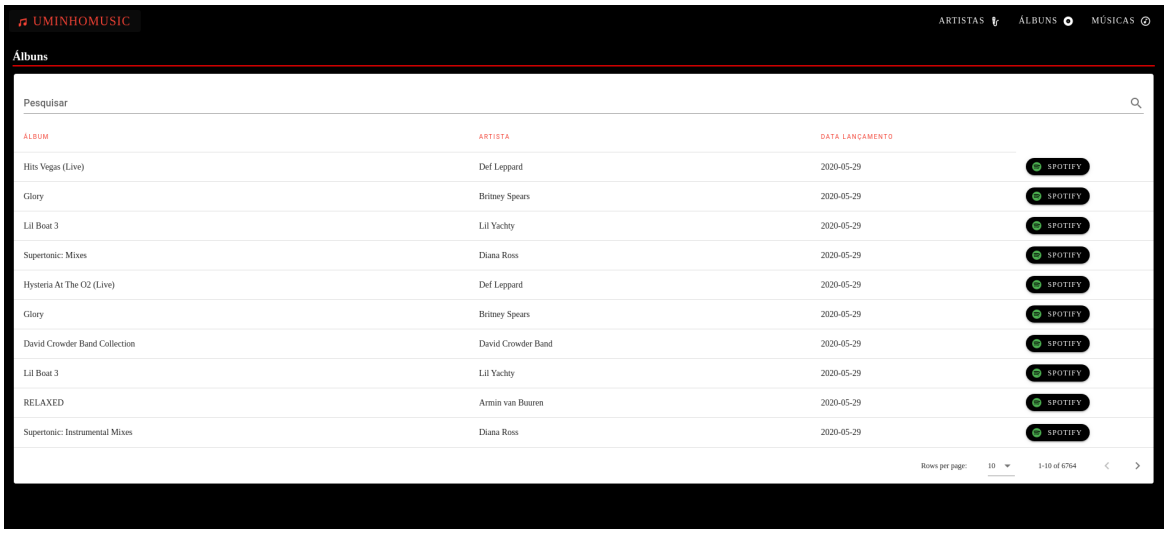

Figura 4: Secção com a tabela dos álbuns

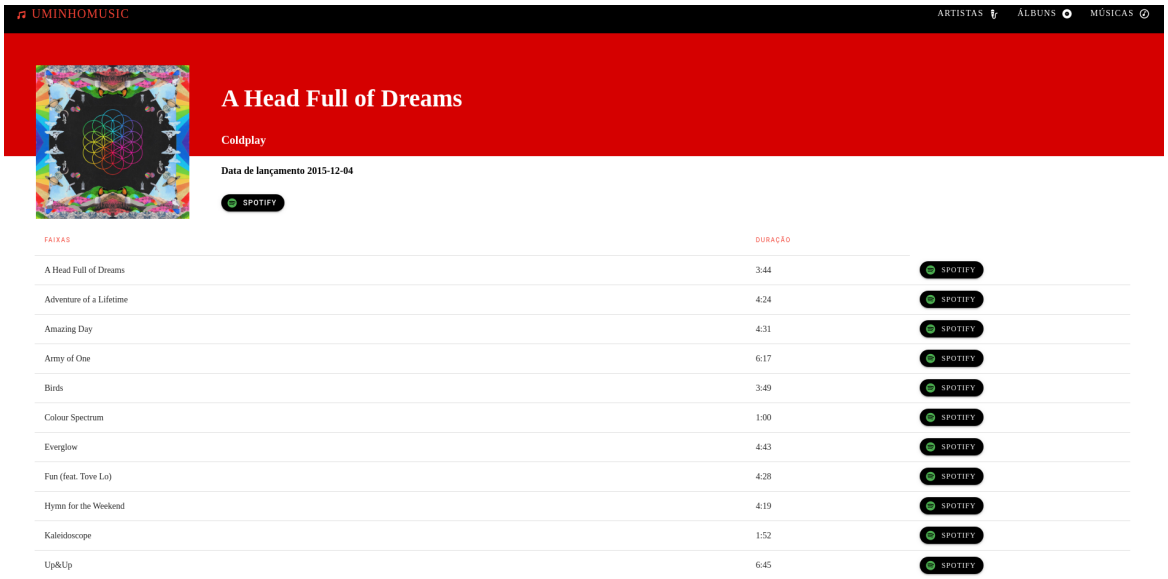

Figura 5: Página individual de um álbum

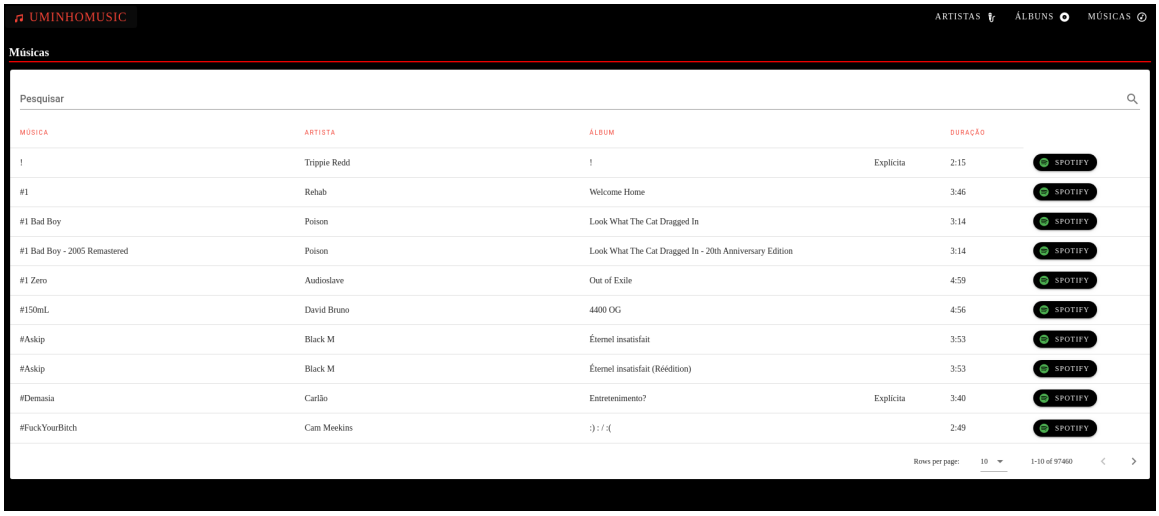

Figura 6: Secção com a tabela das musicas

5 Conclusões

Para concluir este documento, iremos abordar as dificuldades encontradas ao longo do caminho e uma apreciação final ao trabalho desenvolvido.

Um dos grandes desafios do trabalho começou na extração de dados, quando tivemos que achar alguma solução para extrair uma série de entradas a partir da API do Spotify. Isto porque, como já foi referido no documento, não existe nenhuma *query* que permita extrair, por exemplo, uma série de artistas aleatórios ou os artistas mais ouvidos na plataforma; pelo que tivemos de encontrar caminhos alternativos.

Já numa fase adiantada da implementação da interface, começamos a notar umas falhas de memória na aplicação. Isto provocava um tempo de espera bastante grande quando navegávamos pelas diferentes páginas do sistema, nomeadamente, quando se abria uma página individual de um artista. Notamos então que isto se devia à grande quantidade de pedidos e dados a circular entre a interface, a API e o GraphDB. Felizmente, conseguimos resolver este problema separando os diferentes *states*. Assim, ficamos com o *state* inicial, que é carregado para a *homepage*, e os restantes, que são atualizados quando é feito um *request* com ID.

No final de contas, podemos dizer que o grupo se encontra bastante satisfeito com o resultado obtido, uma vez que cumpriu grande parte dos requisitos delineados numa fase inicial. Este resultado engloba uma ontologia com milhares de entradas distintas e uma aplicação Web simples e organizada, com um *design* minimalista.