



UNIVERSIDADE DO MINHO
MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA

PROCESSAMENTO E REPRESENTAÇÃO DE
CONHECIMENTO

MusicDB - Consulta de informação musical

Manuel Gouveia Carneiro de Sousa - **A78869**

13 de Junho de 2019

Resumo

Este relatório tem como principal objetivo exibir quais os passos tomados na criação de uma ontologia com o intuito de suportar uma aplicação de pesquisa tendo em conta um domínio fechado. Nesta aplicação, é possível consultar informações relativas a diversos artistas musicais, através de diversas filtragens aos dados disponíveis.

Conteúdo

1	Introdução	3
2	Ontologia	3
2.1	Estrutura	3
2.1.1	Classes	4
2.1.2	Relações	4
2.1.3	Atributos	5
2.2	Dataset & Parser	5
3	Aplicação Web	6
3.1	Interface e forma de pesquisa	6
4	Conclusões	8

1 Introdução

Uma Ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um domínio fechado. Esta, pretende descrever indivíduos, classes, atributos e relações de forma a caracterizar cada objeto presente num domínio. Através de uma ontologia é possível inferir sobre os objetos presentes num domínio, permitindo-nos assim obter mais conhecimento.

Neste presente trabalho, o conceito de ontologia foi usado por forma de manter uma conexão lógica entre os diversos objetos que representariam um domínio no contexto musical. O objetivo era o de ser possível a relação entre diversas classes deste domínio, como um artista e todas as suas obras, e posteriormente usar todo esse conhecimento numa aplicação que permitisse uma consulta flexível e responsiva de todos os dados disponíveis, criados ou inferidos.

Para o desenvolvimento deste projeto, foram necessárias diversas ferramentas para suportar as diferentes camadas que constituiriam a aplicação. Para o auxílio na criação da ontologia foi usado o *Protégé*, o qual permitia a inferência de conhecimento e ainda a geração de diversos formatos para representação da ontologia (para este presente trabalho o formato escolhido foi *Turtle*). Para suportar todo conhecimento e informação, foi usado o *GraphDB*. Na criação da aplicação, foi usado *Node.js* como *backend* em conjunto com *Vue.js* para a criação da interface.

Por último, o relatório apresenta os resultados obtidos e o comportamento da aplicação.

2 Ontologia

Como referido anteriormente, uma ontologia tem como função descrever indivíduos, classes, atributos e relações de forma a caracterizar diversos objetos pertencentes a um domínio. Visto isto, a estrutura criada parte de um âmbito musical, em que existem artistas, cada um tendo as suas obras, e cada uma das obras terá diversos atributos que a classificam.

2.1 Estrutura

Nesta secção, serão então enumeradas as classes que compõem a ontologia criada, todas as relações que permitem a interligação entre os diversos objetos

e, por fim, os atributos que permitem classificar cada objeto.

2.1.1 Classes

1. **Artist** - classe principal que representa um artista musical;
2. **Group** - subclasse de **Artist**, e que representa um grupo musical;
3. **Solo** - subclasse de **Artist**, e que representa um artista solo;
4. **Person** - classe que representa uma pessoa;
5. **Member** - subclasse de **Person**, a qual representa um membro de um grupo;
6. **Album** - classe que representa um álbum de música;
7. **Song** - classe que representa uma música;
8. **Genre** - classe que representa um género musical;
9. **Style** - classe que representa um estilo musical.

2.1.2 Relações

1. **hasAlbum** - representa a relação entre **Artist** e **Album**, indicando que um artista (solo ou grupo) tem álbuns;
2. **hasMember** - representa a relação entre **Artist** e **Member**, a qual significa que um artista (solo ou grupo) é composto por membros;
3. **hasGroup** - relação inversa de **hasMember**, indicando assim que um membro pertence a um artista;
4. **hasSong** - representa a relação entre **Album** e **Song**, indicando que um álbum é composto por músicas;
5. **hasGenre** - representa a relação entre **Album** e **Genre**, indicando que um álbum pertence a um determinado género musical;
6. **hasStyle** - representa a relação entre **Album** e **Style**, indicando que um álbum tem um determinado estilo musical;

7. **classifiesAlbum** - relação inversa de **hasGenre**, a qual indica que um determinado género musical classifica um álbum;
8. **includesAlbum** - relação inversa de **hasStyle**, a qual indica que um determinado álbum é incluído por um estilo musical.

2.1.3 Atributos

É importante referir que alguns dos atributos usados eram comuns a várias classes, pelo que são usados por diversos objetos com classificações diferentes.

1. **name** - atribui um nome a um artista solo ou a um grupo musical, ou um nome a um determinado género ou estilo musical;
2. **realname** - representa o nome verdadeiro de um artista solo;
3. **discogs_id** - número que identifica o artista na plataforma *Discogs*;
4. **title** - atribui um título a um álbum ou a uma música;
5. **year** - número que representa o ano de lançamento de um determinado álbum;
6. **country** - representa o país em que o álbum foi produzido;
7. **duration** - representa a duração de uma determinada música;
8. **position** - posição de uma música num determinado álbum.

2.2 Dataset & Parser

Depois de criada a ontologia, era necessária a criação de indivíduos que fizessem o uso da representação do conhecimento fornecida pela mesma. O grande desafio era o de arranjar dados relativamente fiáveis que fossem ao encontro daquilo que era pretendido. Depois de uma intensa pesquisa, a plataforma *Discogs* era aquela que oferecia uma vasta biblioteca musical, bem como *data dumps* cujo formato e organização favoreciam o seu processamento. Estes dados, compostos em 3 ficheiros *XML*, continham uma panóplia de artistas e suas obras, bem como informações relativas a cada um destes. Como estes ficheiros eram de uma dimensão elevada (cerca de 50 GB

cada um), foram apenas considerados cerca de 150 MB do ficheiro principal, o qual continha a informação necessária para a criação de indivíduos.

Uma vez obtido o *dataset* o qual continha os dados pretendidos, era necessária a sua análise e tratamento. De facto, era obrigatório o processamento deste ficheiro visto que a ontologia se encontra no formato *Turtle*, pelo que foi criado um script *Python* capaz de encontrar a informação pretendida, e posteriormente gerar os diversos indivíduos que iriam ser acrescentados ao ficheiro *ttl* gerado pelo *Protégé*. Por fim, e corrigidos eventuais erros ao longo do ficheiro gerado, era feita a concatenação do ficheiro inicial com este novo ficheiro, obtendo assim uma ontologia final com inúmeros indivíduos devidamente classificados.

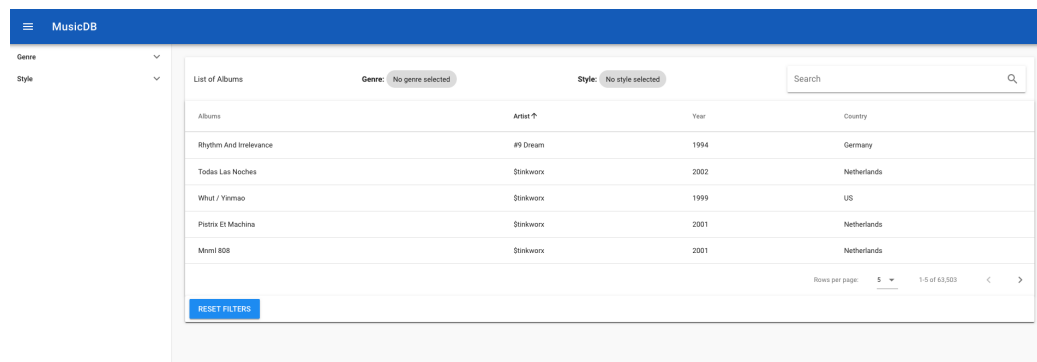
3 Aplicação Web

O principal objetivo da aplicação a ser desenvolvida era o de proporcionar uma fácil e intuitiva consulta a todos os dados presentes na ontologia. De forma a ir de encontro com este requisito, uma das soluções foi então construir uma interface usando a framework de *Javascript*, *Vue.js*, aplicando ainda uma componente denominada de *Vuetify*, que faz o uso do *Material Design* desenvolvido pela *Google*, dando acesso a diversos componentes úteis e bastante responsivos. Para armazenar os dados, foi então utilizada uma base de dados em *GraphDB* a qual teria a capacidade de suportar e interpretar o ficheiro *ttl* previamente criado, permitindo assim, através da linguagem para *queries RDF* denominada de *SPARQL*, interrogar e filtrar o conhecimento contido na nossa base de dados. Visto isto, era necessária uma forma de comunicação entre o servidor de *frontend* e a base de dados, pelo que foi construído um servidor em *Node.js*, que tratava de lidar com todos os pedidos por parte do utilizador, e interrogar a base de dados com diversas *queries* predefinidas.

3.1 Interface e forma de pesquisa

O grande objetivo ao construir uma interface neste contexto era o de permitir, de forma simples e flexível, uma consulta a todos os dados que foram carregados para o *GraphDB*. Desta forma, a aplicação é centrada numa página principal a qual consiste num menu lateral que permite a filtragem por género/estilo musical, e numa tabela centrada na qual aparecem os diver-

sof álbuns e respetivos artistas, onde é possível efetuar uma procura manual através de uma *Search Box*.



The screenshot shows the MusicDB application interface. At the top is a blue header with the MusicDB logo. On the left is a sidebar with 'Genre' and 'Style' filters. The main area displays a table of albums with columns for Album, Artist, Year, and Country. A search bar is located at the top right of the main area. Below the table, there is a 'Rows per page' selector set to 5, showing '1-5 of 63,303' items, and a 'RESET FILTERS' button.

Albums	Artist ↑	Year	Country
Rhythm And Inrelevance	#9 Dream	1994	Germany
Todas Las Noches	Stinkwore	2002	Netherlands
What / Yimnap	Stinkwore	1999	US
Pistons Et Machina	Stinkwore	2001	Netherlands
Mom! BOB	Stinkwore	2001	Netherlands

Figura 1: Menu Principal

Clicando numa entrada dessa tabela, o utilizador é levado para o menu do artista, onde aí são encontradas as informações relativas ao mesmo, como nome, nome verdadeiro/membros, e toda a sua obra. É de notar que, as imagens que fazem parte do *slideshow* em cada página do artista, são obtidas através de um pedido GET aos *endpoints* da plataforma *Discogs*, usando o atributo *discogs_id*, previamente mencionado.

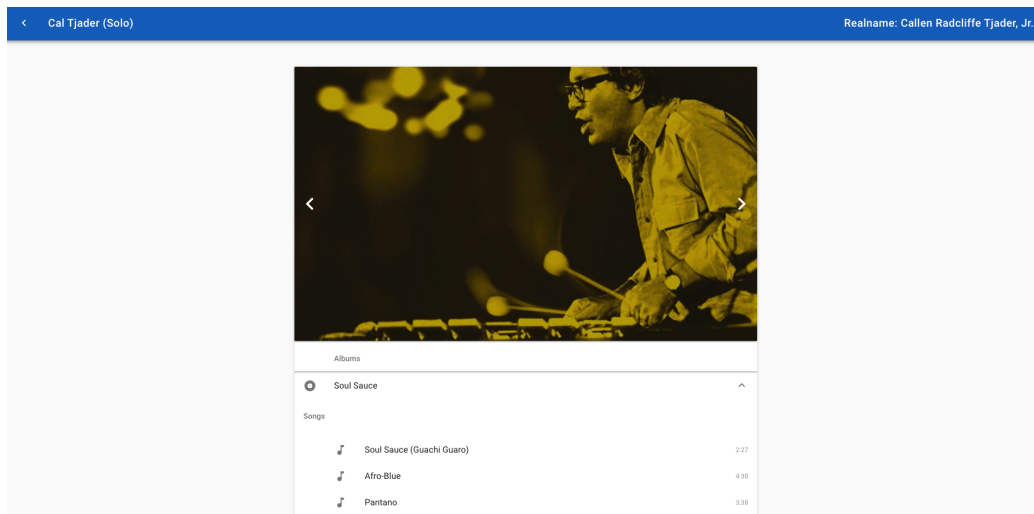


Figura 2: Menu do Artista

4 Conclusões

É importante referir que, apesar da forma de consulta ser bastante intuitiva, o facto é que a tabela é “bombeada” com dados cada vez que entramos na página principal. De facto, mesmo após retrocedermos de uma página de um artista, a tabela é novamente carregada, o que causa uma certa latência no que toca ao preenchimento da tabela. Neste caso, não foi possível a implementação de um certo tipo de *cache* capaz de guardar o estado anterior, sem ter que voltar a fazer um pedido ao servidor. Outra solução seria, por exemplo, “partir” os dados antes de estes serem inseridos na tabela, inserindo assim apenas o necessário. Por último, realçar que uma boa estruturação da ontologia permitiu com fosse possível conjugar diversos tipos de informação, simplificando desta forma as interrogações necessárias para obter os dados pretendidos, facilitando assim cada pedido feito pelo utilizador.

Referências

- [1] <https://vuetifyjs.com/en/getting-started/quick-start> - **Vuetify.js**
- [2] <https://www.discogs.com/> - **Discogs Music Database and Marketplace**
- [3] <https://data.discogs.com/?prefix=data/2019/> - **Discogs Data**