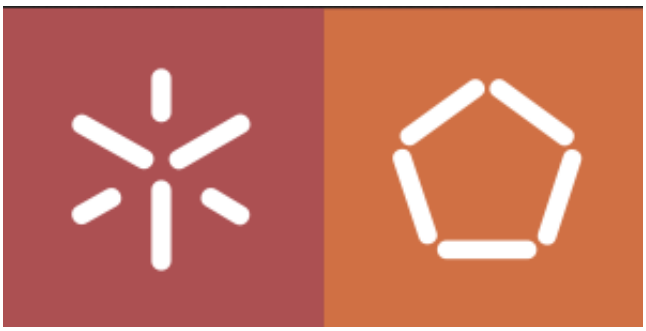

Universidade do Minho

Mestrado em Engenharia Informática

Processamento e Representação de Conhecimento

Game Store

Membros do Grupo:

João Vieira	A76516
Manuel Monteiro	A74036
Miguel Dias	PG41089

5 de Julho de 2020

Conteúdo

1 Introdução

Este relatório é referente a uma aplicação Web desenvolvida no âmbito do perfil de Processamento de Linguagens e Conhecimento. Com a temática livre, o grupo decidiu implementar uma plataforma com informações sobre vídeo jogos.

O objetivo do trabalho passou por desenvolver uma aplicação, que dentro da temática do projeto, estaria dividida por um servidor com uma API de dados, e uma interface que permitisse fornecer as funcionalidades sobre essa API. Sendo assim, a resolução do nosso grupo passa por implementar em *NodeJS* o servidor *backend*, com o auxílio de *scripts* em *Python* de forma a ir actualizando a informação. Quanto ao *frontend*, o seu desenvolvimento passou por utilizar uma *framework* de interfaces reativas com *Vue.js*.

As próximas alíneas explicam toda a arquitetura e lógica pensada para o funcionamento da aplicação.

2 Ontologia

Para representar o conhecimento neste domínio foi então criada uma ontologia que traduz toda a informação sobre os vídeo jogos e os possíveis utilizadores da aplicação. A persistência de dados desta aplicação foi implementada em *GraphDB*, uma base de dados orientada a grafos. Primeiramente, foi feita uma extração de informação para povoar a base de dados com informações de jogos, com recurso a *webscrapping* e diversas APIs disponíveis acerca deste tema.

2.1 Caracterização da ontologia

2.1.1 Classes

Após a extração da informação, podemos inicializar a definição da ontologia com base na informação que está inserida no domínio de dados e no objetivo da aplicação. Numa primeira fase, foram identificadas várias classes:

- **User** - Representa os utilizadores da aplicação.
- **Game** - Representa os jogos existentes da aplicação.
- **Company** - Representa as companhias desenvolvedoras e editoras de vídeo jogos.
- **Category** - Classe que representa a categorização dos jogos pela sua forma de jogar.
- **Genres** - Classe que representa os géneros dos jogos pelo seu estilo de jogo.
- **Plataform** - Representa as plataformas nas quais o jogo pode ser jogado.
- **Sale** - Representa a promoção que pode haver num dado jogo.

Foram ainda criadas duas subclasses de companhias, pois estas podem ser editoras e/ou desenvolvedoras de jogos. Daí termos ainda as classes **Publisher** e **Developer** para sub-categorizar as empresas presentes na ontologia.

2.1.2 Propriedades de objetos

De forma a relacionar as várias classes presentes na ontologia, foram definidas várias propriedades de objetos:

- **hasCategory** (*domínio*: **Game**, *contra-domínio*: **Category**)
- **isCategoryOf** (*inversa de*: **hasCategory**)
- **hasDeveloper** (*domínio*: **Game**, *contra-domínio*: **Developer**)
- **isDeveloperOf** (*inversa de*: **hasDeveloper**)
- **hasGenre** (*domínio*: **Game**, *contra-domínio*: **Genre**)
- **isGenreOf** (*inversa de*: **hasGenre**)

- **hasPlatform** (*domínio*: **Game**, *contra-domínio*: **Platform**)
- **isPlatformOf** (*inversa de*: **hasPlatform**)
- **hasPublisher** (*domínio*: **Game**, *contra-domínio*: **Publisher**)
- **isPublisherOf** (*inversa de*: **hasPublisher**)
- **hasSale** (*domínio*: **Game**, *contra-domínio*: **Sale**)
- **isSaleOf** (*inversa de*: **hasSale**)
- **owns** (*domínio*: **User**, *contra-domínio*: **Game**)
- **ownedBy** (*inversa de*: **owns**)
- **wishes** (*domínio*: **User**, *contra-domínio*: **Game**)
- **wishedBy** (*inversa de*: **wishes**)

2.1.3 Propriedades de dados

De forma a representar informação sobre as classes apresentadas, foram definidas várias propriedades de dados:

- **achievments** (*domínio*: **Game**)
- **averagePlaytime** (*domínio*: **Game**)
- **description** (*domínio*: **Game**)
- **description** (*domínio*: **Game**)
- **discount** (*domínio*: **Sale**)
- **email** (*domínio*: **User**)
- **id** (*domínio*: **User**)
- **image** (*domínio*: **Game**)
- **name** (*domínio*: **Game**, **Company**, **Genre**, **Category**)
- **password** (*domínio*: **User**)
- **price** (*domínio*: **Game**)
- **rating** (*domínio*: **Game**)
- **releaseDate** (*domínio*: **Game**)
- **salePrice** (*domínio*: **Sale**)
- **username** (*domínio*: **User**)
- **website** (*domínio*: **Game**)

2.2 Grafo da ontologia

Para uma melhor percepção da ontologia, foi gerado um grafo da mesma utilizando o *plugin OntoGraf* disponibilizado na ferramenta *Protegé*.

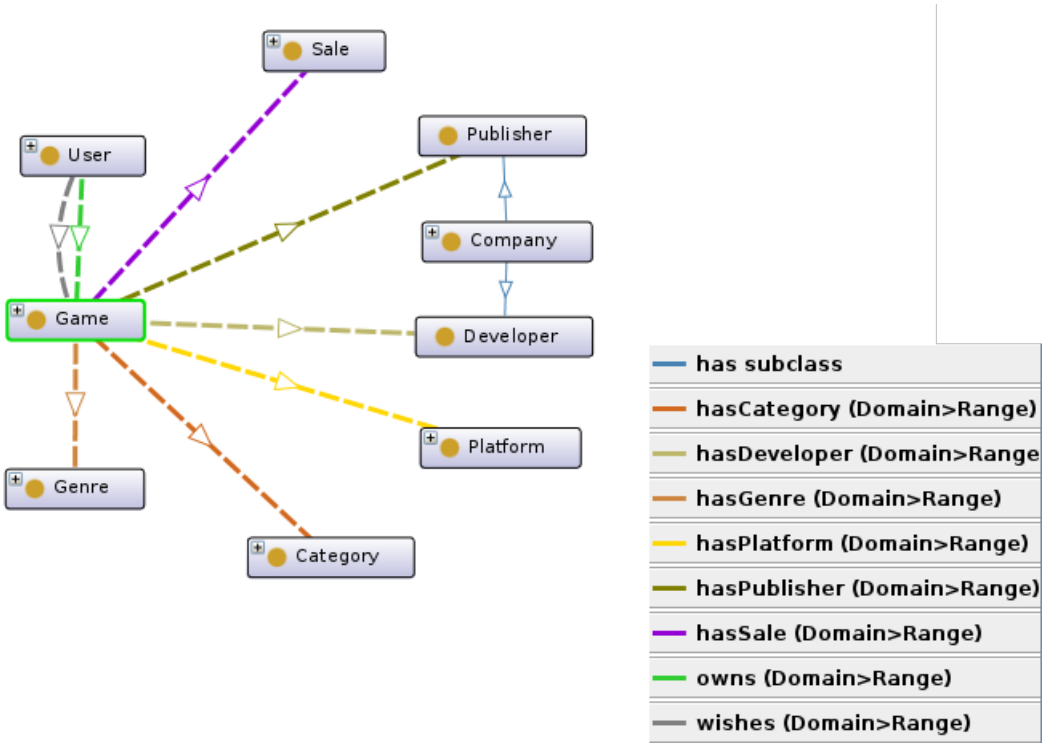

Figura 1: Grafo da ontologia

3 Desenvolvimento

Em termos de tecnologias utilizadas o grupo optou por aquelas que foram leccionadas nas aulas. Sendo assim, o *frontend* foi desenvolvido em *Vue.js* com recurso ao *plugin Vuetify*. Já o *backend* foi implementado em *Node.js + Express.js* com persistência de dados em *GraphDB*, como já foi mencionado.

3.1 Servidor *Backend*

Quanto às rotas, foram criados um roteador da API de dados por cada uma das classes referidas anteriormente, *users*, *games*, *categorys*, *sales*, *devs*, *pubs* e *genres*.

Também foi implementado no servidor um *script* em *Python*, que realiza *Web-Scraping* a uma página de promoções de jogos, de modo a termos a informação sobre as promoções dos jogos sempre atualizada na nossa aplicação.

Para assegurarmos a segurança e limitarmos o acesso à nossa API de dados, protegemos todas as rotas exceto a autenticação e o registo, através de uma função que verifica se o *token* enviado nos pedidos é válido.

3.2 Servidor *Frontend*

O *frontend* é constituído por várias *views*: *homepage*, *login*, loja, livraria de jogos, *wishlist* e página de jogo.

É utilizado uma biblioteca de gestão de estados chamada de *Vuex*, que não só permite ter uma melhor organização de dados e facilidade de os reutilizar como também serve de *cache* de informação entre mudanças de página reduzindo o número de chamadas ao servidor tornando a aplicação mais fluida. São guardadas na biblioteca todos os dados do utilizador bem como o seu *token* de acesso, é também guardado a lista de jogos pertencente à sua livraria e *wishlist*.

3.3 Autenticação

No que diz respeito à autenticação, utilizamos *Json Web Token* (JWT) com o algoritmo RSA256, que utiliza um par de chaves pública e privada. O servidor tem uma chave privada para gerar a assinatura do *token*, que após autenticação, o cliente recebe e este é sempre validado através de uma chave pública. Quaisquer futuros pedidos por parte do cliente, terão que incluir no *Authorization Header*, o *token* de modo a aceder aos recursos do servidor. O tempo de expiração de um *token* na nossa aplicação é de uma hora, em que após este tempo o cliente tem que voltar a realizar a autenticação.

4 Conclusão

Chegado o fim do desenvolvimento do projeto, podemos afirmar que as competências adquiridas durante as aulas da U.C. de PRC, foram melhoradas e permitiu-nos alargar algum deste conhecimento. O trabalho também permitiu complementar os conhecimentos da U.C. do 1º semestre e ter uma visão muito mais clara do que é o desenvolvimento e a *stack* de uma aplicação Web, visão essa que não era tão clara durante a licenciatura.

Relativamente aos requisitos propostos, achamos que foram implementados com sucesso tanto em termos de *backend*, como em termos da interface em *Vue.js*. O aspeto que talvez levantou maiores questões de nossa parte, foi a realização das interrogações **SPARQL** aos nossos dados, que estavam na forma de ontologia, formato este que nos foi introduzido este semestre na U.C.

Em suma, o projeto apesar de ter um grau de complexidade um pouco elevado, tornou-se um desafio o que nos fez aumentar bastante o conhecimento nesta área tão importante como é o desenvolvimento de aplicações *Web*.