UC de Projeto

Proposta 2: A minha Árvore Genealógica

Relatório de Desenvolvimento

Orientador: José Carlos Ramalho (DI)

André Pereira (A79196)

Filipe Miranda (A78992)

3 de Julho de 2018

Resumo

Este projeto consistiu no desenvolvimento de uma ontologia capaz de suportar informação genealógica, juntamente com o seu armazenamento num sistema de gestão de bases de dados de grafos e construção de uma interface Web para exploração da mesma.

Conteúdo

1 Introdução **2**

2 Análise e Especificação **3**
2.1 Ontologia . . . 3
2.1.1 Elementos de uma Ontologia . . . 3
2.1.2 RDF . . . 3
2.1.3 OWL . . . 3
2.2 Protégé . . . 4
2.3 GraphDB . . . 4
2.4 SPARQL . . . 5
2.4.1 Exemplo . . . 5
2.5 Programação Web . . . 5
2.5.1 Pug . . . 5
2.5.2 Node.js . . . 6
2.5.3 JavaScript . . . 6
2.6 Especificação dos Requisitos . . . 6

3 Modelação e Construção da Ontologia **7**
3.1 Indivíduos . . . 7
3.2 Relações . . . 8
3.3 Classes . . . 9

4 Implementação **10**
4.1 Arquitetura Geral . . . 10
4.2 Importar Ontologia no GraphDB . . . 10
4.3 Comunicação entre a Aplicação Web e o GraphDB . . . 11
4.4 Aplicação Web . . . 11

5 Aplicação - Exemplos **12**
5.1 Interrogação da Base de Dados através de Queries . . . 12
5.2 Listar todos os Indivíduos . . . 12
5.3 Listar Indivíduo Específico . . . 12
5.4 Inserir Indivíduo na Árvore Genealógica . . . 14

6 Conclusão **15**

Capítulo 1

Introdução

Este projeto enquadra-se no 3º ano da Licenciatura em Ciências da Computação da Universidade do Minho.
Uma árvore genealógica é um histórico dos antepassados totais/parciais de um indivíduo ou família. Mais especificamente, trata-se de uma representação gráfica genealógica para mostrar as conexões familiares entre indivíduos, apresentandos os nomes e, algumas vezes, datas e lugares de nascimento, casamento, etc. O uso destas é comum para provas de ancestralidade e também é usada na medicina, para estudos de doenças de origem genética, por exemplo. A árvore genealógica de uma pessoa é um domínio bem definido para a criação de uma ontologia que servirá de base para uma aplicação web. Os objetivos deste projeto são:

- Desenvolver uma ontologia em OWL capaz de suportar a descrição da árvore genealógica de um conjunto de indivíduos: dados pessoais, relações de parentesco, eventos como nascimento, etc;
- Importar a ontologia para uma base de dados orientada a grafos (modelo não relacional);
- Desenvolver uma aplicação Web que permita a exploração da mesma, cujo resultado é obtido através da invocação de queries SPARQL.

Neste relatório, para além da concepção da proposta que elaboramos para o projeto, também haverá um breve resumo das tecnologias que foram usadas no desenvolvimento de todos as componentes do sistema pretendido. Por fim serão apresentados todos os procedimentos desenvolvidos ao longo destes meses de trabalho e alguns exemplos da aplicação funcional.

Capítulo 2

Análise e Especificação

2.1 Ontologia

Uma ontologia consiste num artefacto de engenharia constituído por um vocabulário e um conjunto de restrições. O vocabulário corresponde a um conjunto de conceitos (ou classes) usados para classificar os elementos de um domínio e o conjunto de restrições permite obter conhecimento adicional sobre o domínio de conhecimento, de maneira a explorar e estender o vocabulário existente. Uma ontologia representa um modelo abstrato e cognitivo de um domínio de conhecimento, identificando os conceitos e as características desse mesmo domínio, sendo explícita no sentido em que as entidades da ontologia estão claramente definidas, distintas e inter-relacionadas entre si.

2.1.1 Elementos de uma Ontologia

- **Classes**: Representam algum tipo de interação da ontologia com um determinado domínio;
- **Atributos**: Cada atributo é utilizado para armazenar informação que é específica para o objeto associado a ele;
- **Relações**: Representam o tipo de interação entre os elementos do domínio (classes) e os indivíduos;
- **Indivíduos/Objetos**: São utilizados para representar elementos específicos, i.e, os próprios dados da ontologia.

2.1.2 RDF

RDF (Resource Description Framework) corresponde a um modelo de dados abstrato para representar informação na Web. Contrariamente aos modelos de dados mais tradicionais, como é o caso do modelo aplicado nas base de dados relacionais, a informação representada em RDF constitui um grafo. O RDF distingue-se dos outros modelos devido à:

- simplicidade e flexibilidade inata para representar novos dados;
- capacidade para representar de forma igual tanto os dados como o esquema subjacente a esses dados.

Em RDF, um triplo (statement) é constituído por um sujeito, predicado e objeto, como exemplifica a seguinte expressão:

```
'José Carlos Leite Ramalho 1967' - 'hasBirthYear' - '1967' (sujeito - predicado - objeto)
```

2.1.3 OWL

O OWL (Web Ontology Language) é usado por aplicativos para processar o conteúdo da informação em vez de apenas apresentar informações.

- É mais expressivo, na medida em que permite expressar ideias muito complexas e subtis sobre os dados e estabelecer uma hierarquia de forma dinâmica;

- É mais flexível, no sentido em que a especificação de um esquema em OWL corresponde à adição de novos triplos RDF, encarando alterações ao esquema de uma forma mais simples e natural do que as aplicações assentes nas linguagens de esquema mais tradicionais;
- Permite minimizar os dados que são explicitamente armazenados e a complexidade das consultas necessárias para recuperar esses mesmos dados.

2.2 Protégé

Protégé é um editor genérico de ontologias gratuito e open source e um sistema de gestão de conhecimento que fornece uma interface gráfica para as definir. Inclui classificadores dedutivos para validar a consistência dos modelos e inferir novas informações com base na análise de uma ontologia e dispõe de muitas extensões (plug-ins) que permitem estender as capacidades básicas do editor.

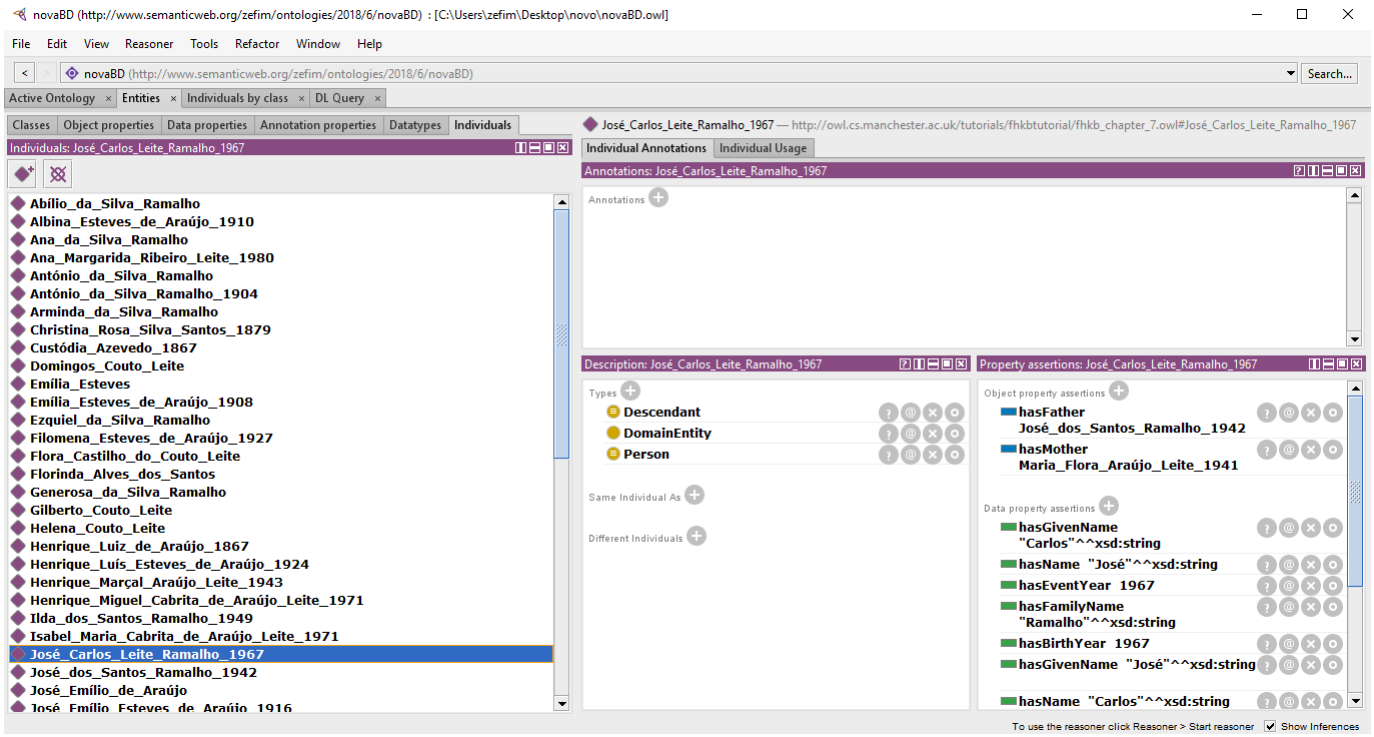

Figura 2.1: Interface do Protégé.

2.3 GraphDB

As redes complexas são redes de dados onde a relação entre os elementos é importante tal como os próprios elementos. Através de grafos podemos fazer a sua representação sendo que a relação é descrita pelas arestas e os elementos são descritos pelos vértices. Este tipo de modelo de base de dados permite uma modelação simples dos dados e uma consulta facilitada, devido à estrutura dos grafos. Trabalhando diretamente com grafos e operações relacionadas, o nível de abstração aumenta, facilitando a manipulação dos dados. O GraphDB é uma base de dados de grafos, compatível com os padrões W3C e tem capacidade para lidar com grandes quantidades de dados, consultas e inferência em tempo real. Bases de dados de grafos fornecem a infra-estrutura principal para soluções onde a agilidade de modelação, integração de dados, exploração de relacionamento e publicação e consumo de dados se tornam importantes.

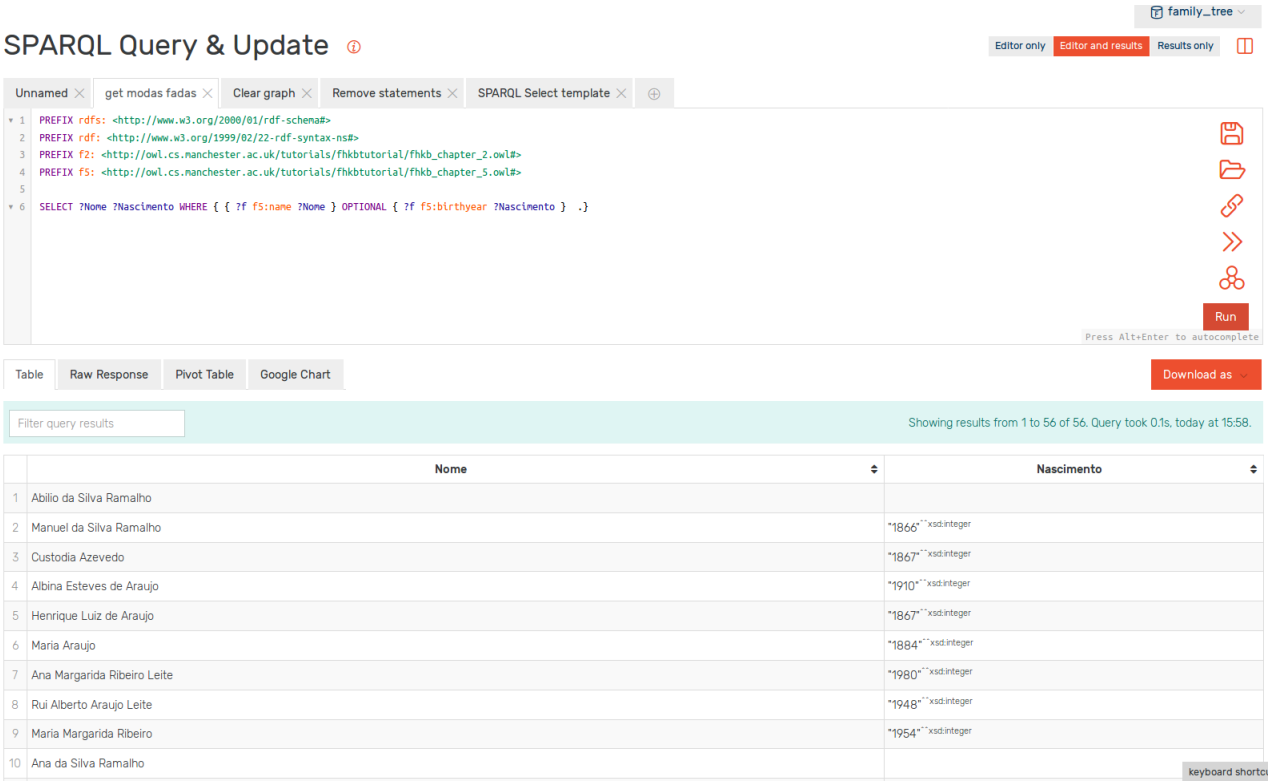

Figura 2.2: Interface do GraphDB.

2.4 SPARQL

O SPARQL é simultaneamente um protocolo e uma linguagem de consulta de dados representados em RDF. Como linguagem de consulta, o SPARQL é usado para realizar queries a dados representados em RDF da mesma forma que se faz no SQL para consultar dados em bases de dados relacionais. Como protocolo, o SPARQL assenta em serviços HTTP para transmitir queries SPARQL e os resultados entre uma aplicação cliente e um SPARQL Endpoint (servidor HTTP que expõe dados).

2.4.1 Exemplo

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX f2: <http://owl.cs.manchester.ac.uk/tutorials/fhkbtutorial/fhkb_chapter_2.owl#>
PREFIX f5: <http://owl.cs.manchester.ac.uk/tutorials/fhkbtutorial/fhkb_chapter_5.owl#>

SELECT ?s WHERE { { ?s f2:hasFather ?o }UNION { ?s f2:hasMother ?o }. }
```

2.5 Programação Web

2.5.1 Pug

Um navegador lê HTML e CSS e, em seguida, exibe imagens formatadas e texto para o cliente com base no que esse HTML e CSS diz para fazer. Pug acaba por ser o intermediário/ um template do Node.js que permite injetar dados e produzir HTML. O Pug (e outros mecanismos de modelo) substituem variáveis dos ficheiros por valores reais e, em seguida, enviam a sequência HTML resultante para o cliente.

2.5.2 Node.js

Node.js é um interpretador de código JavaScript open source, focado em migrar o Javascript do lado do cliente para servidores. Seu objetivo é ajudar programadores na criação de aplicações de alta escalabilidade, com códigos capazes de manipular dezenas de milhares de conexões simultâneas, numa única máquina física. O Node.js é baseado no interpretador V8 JavaScript Engine e o desenvolvimento é mantido pela fundação Node.js em parceria com a Linux Foundation.

2.5.3 JavaScript

JavaScript é uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o utilizador sem a necessidade deste script passar pelo servidor. Atualmente, é a principal linguagem para programação client-side em navegadores web. Começa também a ser bastante utilizada do lado do servidor através de ambientes como o node.js. Foi concebida para ser uma linguagem script com orientação a objetos baseada em protótipos, tipagem fraca e dinâmica e funções de primeira classe.

2.6 Especificação dos Requisitos

A ontologia em questão terá que ser capaz de suportar a descrição da genealogia de um conjunto de indivíduos, desde dados pessoais passando pelas relações de parentesco e pelos eventos significativos da vida de uma pessoa, havendo a possibilidade de associar fotografias ou registos documentais que sejam de interesse. No fim, a ontologia deverá ser armazenada num sistema de gestão de bases de dados de grafos, o GraphDB ou o BlazeGraph, e deverá ser construída uma interface Web para exploração da mesma.

Requisitos e conceitos que foram estudados no decurso do projeto:

- Ontologias;
- OWL, RDF;
- Bases de dados de grafos;
- Programação Web: HTML, CSS e Javascript.

Capítulo 3

Modelação e Construção da Ontologia

3.1 Indivíduos

Indivíduos representam objetos no domínio em que estamos interessados. Uma diferença importante entre Protégé e OWL é que o OWL não usa a suposição de nome exclusivo. Isto significa que dois nomes diferentes podem se referir ao mesmo indivíduo. Por exemplo, 'Professor Ramalho' e 'José Carlos Ramalho' podem se referir ao mesmo indivíduo. Em OWL, tem que se afirmar explicitamente que os indivíduos são iguais ou diferentes entre si.

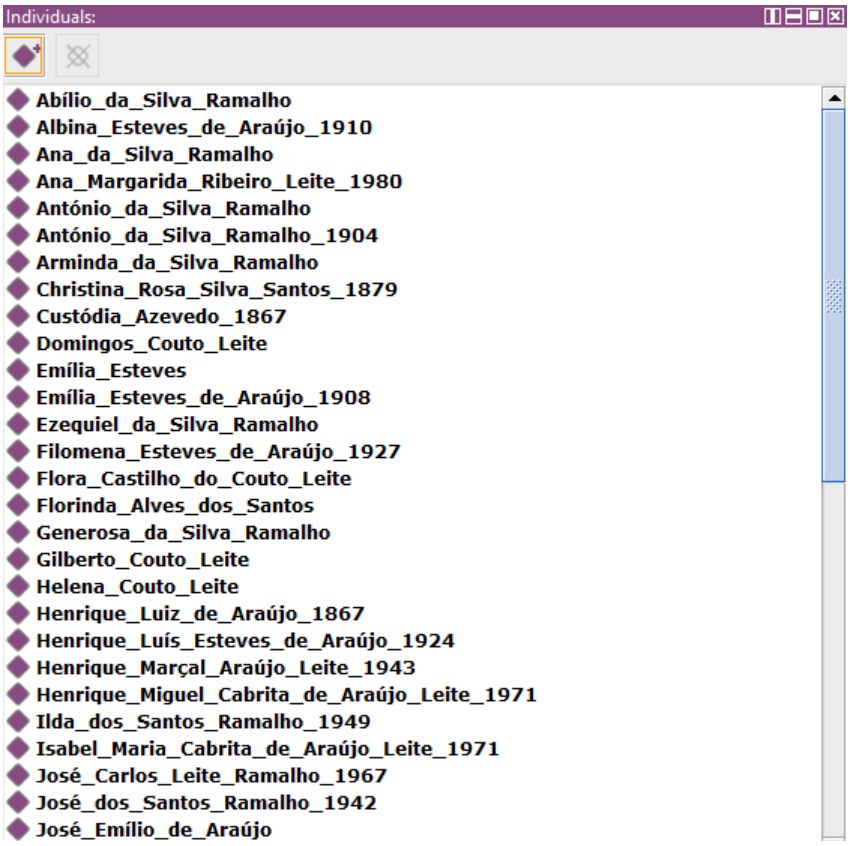

Figura 3.1: Exemplo dos indivíduos presentes na nossa ontologia.

3.2 Relações

As relações correspondem a relações binárias entre indivíduos, ou seja, uma relação liga dois indivíduos. Por exemplo, a relação hasFather pode ligar o indivíduo Abílio ao indivíduo António, ou a relação hasGrandmother pode ligar o indivíduo Helena ao indivíduo Maria. As relações podem ter inversos. Por exemplo, o inverso de hasFather é isFatherOf. As relações podem ser limitadas a ter um único valor, isto é, serem funcionais. Também podem ser transitivas ou simétricas e ter certas restrições implementadas. A Figura 3.2 mostra uma representação das relações entre dois indivíduos.

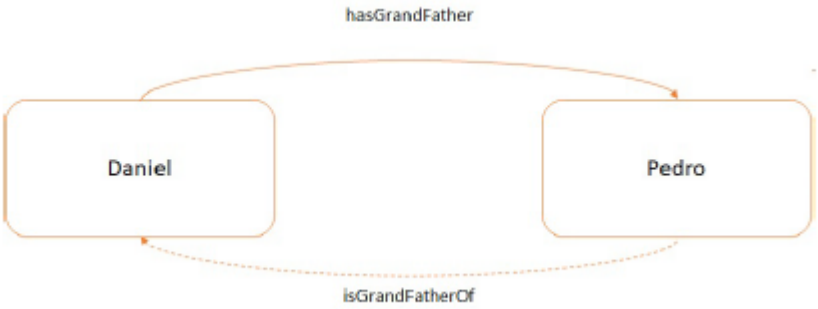

Figura 3.2: Exemplo de uma relação entre avô e neto e da sua inversa.

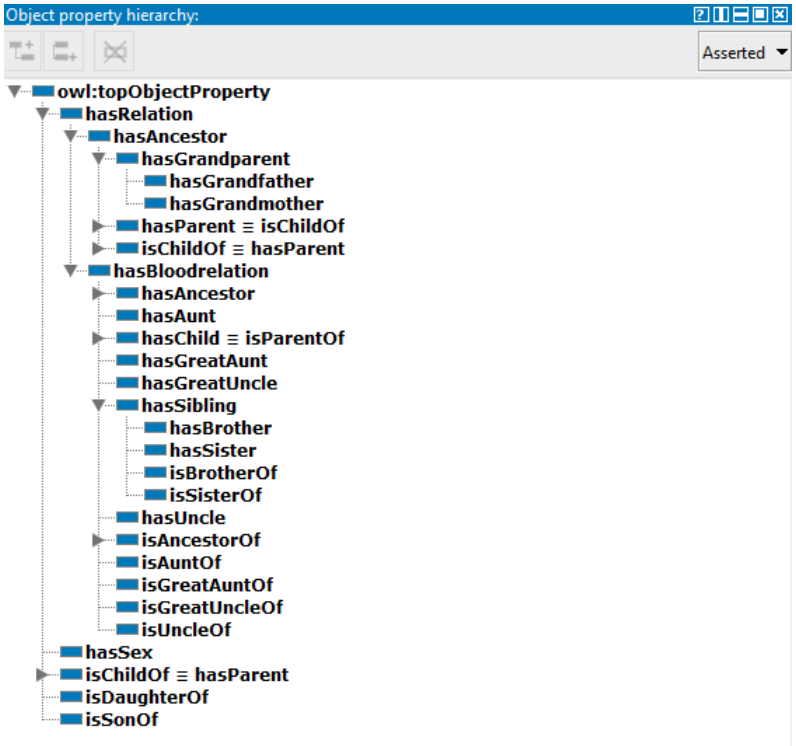

Figura 3.3: Listagem das relações presentes na nossa ontologia.

3.3 Classes

As classes são interpretadas como conjuntos que contêm indivíduos. São descritas usando formúlas lógicas e matemáticas que descrevem com precisão os requisitos para a adesão à classe. Por exemplo, a classe Male conteria todos os indivíduos que são pessoas do sexo masculino no nosso domínio de interesse. As classes podem ser organizada em uma hierarquia de superclasse-subclasse, que também é conhecida como taxonomia. Por exemplo, consideremos as classes Pessoa e Homem (sendo Homem uma subclasse de Pessoa e, consequentemente, Homem é uma superclasse de Pessoa). Isto diz que "Todos os homens são pessoas", "Ser Homem implica ser Pessoa". Uma das principais características do OWL-DL é que as relações subclasses-superclasses podem ser computados automaticamente por um reasoner. Nas classes são construídas descrições que especificam as condições que devem ser satisfeitas por um indivíduo para que seja um membro da classe.

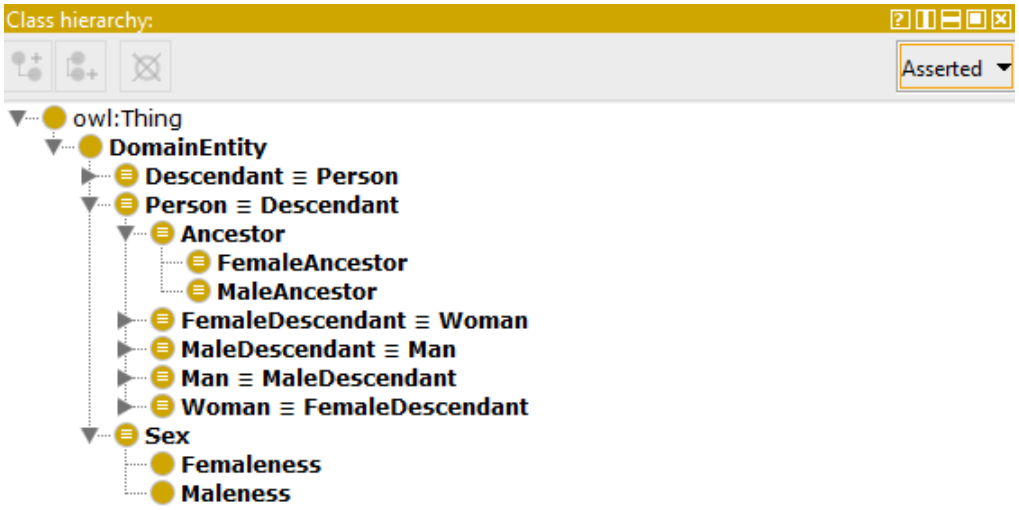

Figura 3.4: Listagem das classes presentes na nossa ontologia.

Capítulo 4

Implementação

4.1 Arquitetura Geral

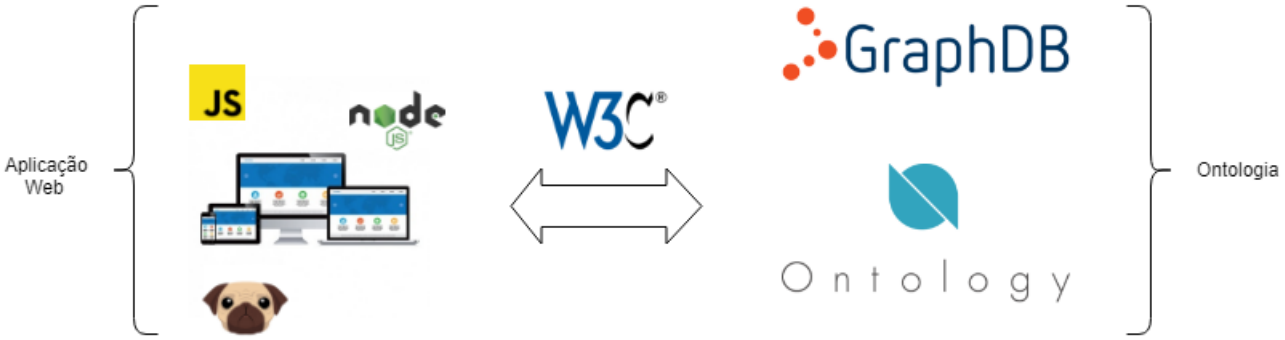

Figura 4.1: Arquitetura.

Como se pode ver no esquema, este projeto consiste num sistema com duas componentes distintas que comunicam entre si. A primeira componente corresponde à ontologia e o respetivo repositório e a segunda consiste na aplicação web que é alimentada pela ontologia. No esquema estão representadas as tecnologias e linguagens usadas, de maneira a que se possa ter uma visão geral de como foi concebido o projeto.

4.2 Importar Ontologia no GraphDB

Posteriormente ao que foi exposto no capítulo de modelação, a ontologia está pronta a ser colocada numa base de dados de grafos, que como foi referido antes, será o GraphDB. De modo a que fosse possível alimentar a aplicação Web foi necessário ter uma base de dados de grafos onde se colocou a ontologia criada de forma a realizar-se as inferências semânticas necessárias, tirando assim todo o proveito da mesma.

O primeiro passo correspondeu a instalar o GraphDB Free. Após instalar e definir a porta que será usada, podemos abrir o GraphDB que automaticamente abre no browser um workbench para que seja feita a gestão dos repositórios e dos grafos que lá existam. Antes de importar no GraphDB, a ontologia do Protégé foi exportada com formato RDF e posteriormente foi criado um novo repositório com o nome desejado.

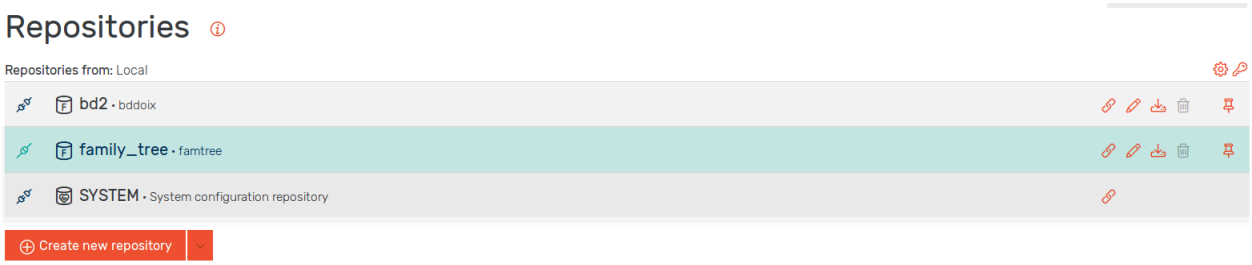

Figura 4.2: Interface do GraphDB - Repositório.

Além do que já foi mencionado, o GraphDB possui também um separador no workbench onde se pode fazer queries ao grafo, sendo útil para a construção das queries usadas na comunicação com a aplicação Web. Esta ferramenta possibilitou testar e visualizar os resultados da query e possíveis erros que a mesma poderia ter, antes de as incorporar nas páginas da aplicação (Ver Figura 2.2).

4.3 Comunicação entre a Aplicação Web e o GraphDB

Identificadas estas duas partes é preciso faze-las comunicar de forma a que a informação existente na ontologia passe para as páginas da aplicação Web. Esta ponte é feita através do SPARQL, uma linguagem de consulta de dados em RDF, que também funciona como protocolo de serviços HTTP, de maneira a que seja possível transmitir os resultados para uma aplicação cliente, que neste caso é a aplicação web que foi criada.

4.4 Aplicação Web

Foi criada uma pequena aplicação web usando a framework Express (para Nodejs), Javascript (maioritariamente em backEnd), usufruindo dos serviços do SPARQL para a gestão do programa, Pug e Ajax para a geração das páginas em HTML e CSS para a criação de uma interface entre o utilizador e a base de dados previamente povoada com todos os dados relativos à árvore genealógica.
Desta forma, recorrendo ao motor do Nodejs é possível obter informação mediante o URL introduzido na barra de procura do browser. É necessário fazer a conexão entre a framework Express e o GraphDB de modo a que a comunicação entre ambos seja feita. Pata tal, após o servidor da base de dados ser invocado e termos obtido um identificador que nos redireciona para o repositório existente, utilizando a API fornecida pelo SPARQL, fazemos a conexão no NodeJS com o repositório ativo, abrindo assim um canal entre estas duas ferramentas, conectando o cliente (que usará o node) ao servidor (da base de dados GraphDB).
Estando o node a executar, é possível usar o endereço IP local como host que será usado como um website. Usando o Express, o serviço foi preparado para aceitar os URL's "http://127.0.0.1:3000/query", "http://127.0.0.1:3000/individuos", "http://127.0.0.1:3000/individuos/<nome>"e "http://127.0.0.1:3000/inserir".
Portanto, caso um destes URL's seja invocado na barra de procura do browser, ele é apanhado pelo serviço do Nodejs e faz um GET ou POST consoante a operação REST a ser realizada. Por exemplo, se quisermos somente os indivíduos existentes na base de dados, é realizado um GET sobre o GraphDB de modo a extrair essa informação. Pelo contrário, se pretendermos editar a própria base de dados, ao introduzir um novo indivíduo, este será injetado no graphDB através do método POST, de maneira a adicionar novos dados no repositório.
Após a framework ter obtido todos os dados pretendidos pelo utilizador, estamos em condições de os formatar legivelmente numa página HTML para que as informações sejam lidas e mostradas pelo browser.
Após a invocação do método HTTP GET, há um reencaminhamento do request do utilizador, que invoca o respetivo ficheiro Pug que é responsável pela geração do HTML juntando a informação proveniente do GraphDB que foi obtida usando interrogações sobre a base de dados pela framework. Assim com o auxilio do javascript e do Pug (PugJS), o código de formatação é gerado e "enviado" para a página, mostrando a informação pretendida pelo utilizador (mediante a sua interrogação usando SPARQL no GraphDB). Há ainda um auxílio ao PugJS através de técnicas web usando Ajax, ativando assim eventos mediante objeto clicados no HTML, ou eventos decorrentes do mesmo.

Capítulo 5

Aplicação - Exemplos

5.1 Interrogação da Base de Dados através de Queries

Caso queiramos interrogar a base de dados utilizando a nossa aplicação, apenas será necessário introduzir o URL "http://127.0.0.1/query"na barra de pesquisa, que gera uma página HTML com capacidade de introduzir a respetiva query numa caixa de texto e após o clique no botão "Processar", a query em SPARQL é enviada para o GraphDB que, por si, retorna a informação em formato JSON para a framework, onde, posteriormente, é formatada para uma tabela em HTML que contém a informação requerida.
O ficheiro "sparql.js", responsável pela gestão de todos os routings relativos à base de dados, ja foi previamente conectado com o GraphDB (como foi mencionado no capítulo anterior). Ao receber o request/ query, este invoca o ficheiro "queryInput.pug"que será resposável por gerar a página de input para a query, contendo uma caixa de texto onde será introduzida a query e os botões responsáveis por limpar a página e por processar a query.
Ao clicar no botão "Processar", o evento relativo ao processamento da query é despoletado pelo Ajax, sendo acionado um pedido POST pelo routing do node. O texto introduzido é enviado para a base de dados sob a forma de query e este responde novamente com a informação formatada em JSON. No node, estes dados são usados pelo Ajax, que cria então uma tabela com esta informação, em formato HTML, obtendo resposta à interrogação (ver Figura 5.1).

5.2 Listar todos os Indivíduos

Ao introduzir "http://127.0.0.1/individuos", obtemos numa tabela o nome e a respetiva data de nascimento de todos os indivíduos da árvore genealógica. De um modo análogo ao exemplo anterior, uma query é enviada ao servidor, sendo a resposta enviada ao nosso programa Web que gerará o HTML contendo os indivíduos. A única diferença do exemplo anterior é que a query foi previamente preparada e é fixa:

```
SELECT ?Nome ?Nascimento WHERE { { ?f f5:name ?Nome } OPTIONAL { ?f f5:birthyear ?Nascimento } .}"
```

Assim quando o request da página é invocado, esta query é automaticamente enviada para o GraphDB e uma vez mais é recebido em JSON a resposta que será formatada com ajuda do PugJS e do Ajax (ver Figura 5.2).

5.3 Listar Indivíduo Específico

Cria uma página HTML contendo o IRI do indivíduo e o ano de nascimento. Para tal introduzimos /<nome> após "http://127.0.0.1/individuos", por exemplo, "http://127.0.0.1/individuos/Jose Carlos Ramalho". Este URL é filtrado pelo routing e o nome (que está seguido de /individuos) é guardado pelo node. Tendo o nome pretendido, de modo análogo aos exemplos anteriores, através de uma query definida e usando esta variável que é o nome do invidíduo a ser filtrado, obtemos o IRI do objeto encontrado e a base de dados responde com este IRI e com a data de nascimento. A página com o resultado é gerada usando o mesmo metodo citado anteriormente (ver Figura 5.4).

```
SELECT ?s ?ano WHERE { { ?s f5:name \"" + nome + "\"} OPTIONAL { ?s f5:birthyear ?ano}  .}
```

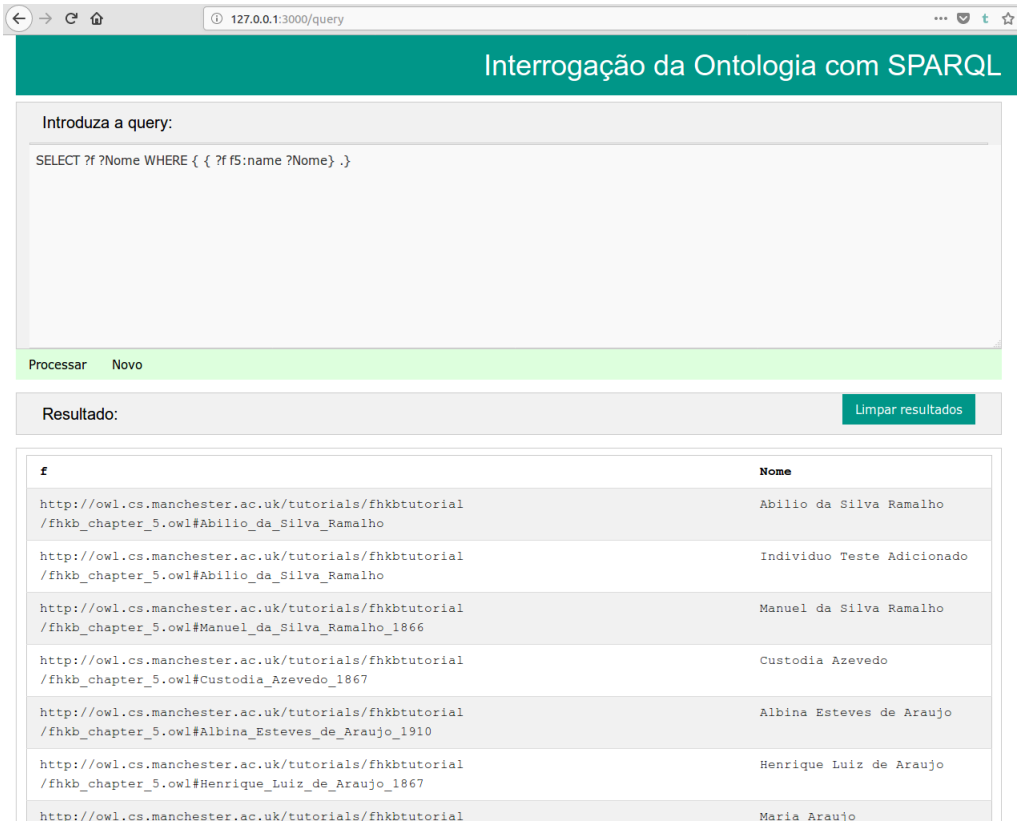

Figura 5.1: Exemplo de uma query.

Figura 5.2: Indivíduos existentes na base de dados.

Figura 5.3: Resultado de listar um indivíduo específico.

5.4 Inserir Indivíduo na Árvore Genealógica

Se quisermos adicionar um novo objeto na base de dados, é apenas necessário introduzir tais dados no formulário gerado no URL "http://127.0.0.1/inserir". Aqui existem dois campos, um relativo ao nome e outro à data de nascimento. Após estes campos estarem preenchidos, e após clicarmos no botão "Inserir", estes dados são enviados para o GraphDB e usados para criar uma nova instância de Person.
Este processo é realizado através o Ajax, que é invocado após o botão ser carregado. Este botão foi gerado pelo Pug que foi invocado devido ao URL existente na barra de procura. De tal modo, o Ajax, ao despoletar o evento, redireciona os dados dos campos de texto para o routing ("sparql.js") e este aciona o método POST. Estes dados são lidos das caixas de texto e após sabermos o nome do indivíduo e a sua data de nascimento, a query:

```
INSERT DATA { " + newIRI + " f5:name " + nome + " ; f5:birthyear " + nascimento + " }
```

é formada, onde newIRI, é um IRI gerado em runtime que será o identificador deste novo indivíduo. Estando a query preparada, esta é injetada na base de dados pela framework, e o SPARQL fica encarregue de povoar o GraphDB com esta informação.

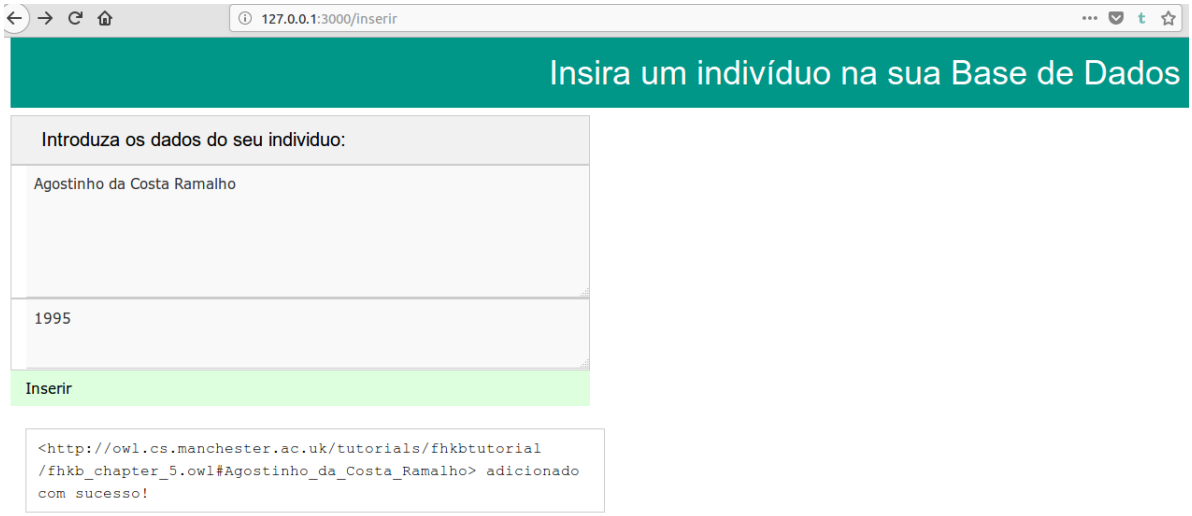

Figura 5.4: Inserção de um indivíduo na Árvore Genealógica.

Capítulo 6

Conclusão

No final deste projeto, o pretendido era obter uma aplicação web suportada por uma ontologia relativa à árvore genealógica de uma família, em que fosse possível fazer a exploração da mesma.

Inicialmente foram-nos propostas quais seriam as ferramentas que iriamos utilizar para o desenvolvimento do sistema por parte do nosso orientador, Prof. José Carlos Ramalho, e, posteriormente, foi feito um estudo das tecnologias através de tutoriais e hiperligações indicadas pelo mesmo.

A primeira parte consistiu em desenvolver a ontologia, que nos ocupou demasiado tempo que devia ter sido dedicado a outras fases do projeto, como a construção da aplicação Web. Tivemos alguns problemas de inferência em certos relacionamentos tais como os dos irmãos, da duplicação dos relacionamentos mãe e da não implementação das relações entre avós e netos, porém foi possível resolvê-los.

A segunda parte consistiu em entender e configurar o GraphDB para dar respostas às necessidades da ontologia e da aplicação Web que deveria ser criada. Acabou por ser um processo rápido, sendo a construção da aplicação Web a etapa mais complicada, devido à falta de bases em JavaScript (node.js). Na etapa final, idealizamos inserir um indivíduo na base de dados através da aplicação, porém de momento não está a ser impresso o HTML correspondente.

No geral, achamos que o nosso projeto está incompleto, havendo vários aspetos que poderiam ser melhorados (nomeadamente a aplicação Web, que não se encontra num estado user-friendly), porém achamos que o projeto contribuiu para termos alguma ideia de como nos devemos adaptar a tecnologias novas e enfrentar desafios aos quais não possuimos bases, mesmo quando temos outras UCs e trabalhos em curso.