INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

HOMI

Hybrid Ontology Mapping Interface

Ana Carolina Baptista
41487@alunos.isel.ipl.pt
960314580

Eliane Almeida
41467@alunos.isel.ipl.pt
960271968

Projeto e Seminário

Licenciatura em Engenharia Informática e Computadores

Relatório Final

Orientadores:
Cátia Vaz, ISEL, cvaz@cc.isel.ipl.pt
José Simão, ISEL, jsimao@cc.isel.ipl.pt
Alexandre P. Francisco, IST, aplf@ist.utl.pt

Julho de 2018

Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores

Hybrid Ontology Mapping Interface

Ana Carolina Grangeiro Pinto Baptista, 41487
Eliane Socorro de Almeida, 41467

Orientadores: Cátia Vaz (ISEL)
José Simão (ISEL)
Alexandre P. Francisco (IST)

Relatório final realizado no âmbito de Projeto e Seminário,
Do curso de licenciatura em Engenharia Informática e de Computadores
Semestre de Verão 2017/2018

14 de Julho de 2018

Resumo

Na área da bioinformática, existem recursos científicos que necessitam de ser partilhados entre a comunidade científica por meio de ontologias. Sendo as ontologias normalmente definidas através de OWL (*Web Ontology Language*), em várias situações poderá não ser uma tarefa simples para os biólogos representar o seu conhecimento do domínio através das ontologias ou anotar recursos já existentes.

O projeto *Hybrid Ontology Mapping Interface*, daqui em diante designada de H.O.M.I., foi um projeto desenvolvido para auxiliar os biólogos a mapear ficheiros previamente descritos noutras linguagens, nomeadamente XML e JSON, para OWL. Este projeto é composto por uma vertente *Web* e uma vertente local, e funciona em modo mapeamento ou em modo editor, sendo que este último apenas anota conceitos descritos em OWL.

Deste modo, um dos grandes objetivos é que os utilizadores consigam de forma simples e intuitiva obter, a partir da descrição de uma ontologia em OWL, um caso concreto da mesma. Para alcançar este objetivo foi nos apresentada a ferramenta *Chaos Pop*, na qual permite mapear ficheiros de dados semiestruturados (XML ou JSON) para OWL.

Palavras-Chave: H.O.M.I.; Ontologias; OWL; Mapeamento de Ontologias; Anotação de conceitos; Chaos Pop.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Listagens

1. Introdução

Atualmente, com o grande crescimento e propagação de dados na internet, surge a necessidade de que a informação seja descrita e transmitida por meio de uma linguagem *standard*, sendo esta de fácil entendimento tanto para computadores quanto para humanos.

Uma das técnicas de descrição de informação que se está a tornar muito popular é baseada em ontologias [1]. Esta permite especificar explicitamente uma conceptualização ou um conjunto de termos de conhecimento para um domínio particular. A semântica web, que utiliza ontologias, é uma visão para o futuro Web na qual a informação é dada com um significado explícito, facilitando o seu processamento e integração automático por máquinas. A Semântica Web é baseada na capacidade do XML definir esquemas *markup* personalizados e na abordagem flexível do RDF para a representação de dados [2].

Na área da bioinformática, existem recursos científicos que necessitam de ser partilhados entre a comunidade científica por meio de ontologias. Sendo as ontologias normalmente definidas através de OWL [3] (*Web Ontology Language*), em várias situações poderá não ser uma tarefa simples para os biólogos representarem o seu conhecimento do domínio através das ontologias ou anotar recursos já existentes.

Apesar da popularidade das ontologias, existe informação que se encontra especificada noutro formato ou noutra linguagem que não a semântica web e que beneficiariam de ficar anotada através de uma ontologia.

Atualmente existem algumas ferramentas de edição de ontologias que permitem ao utilizador inserir um ficheiro referente a uma ontologia e criar novos dados de acordo com este ficheiro, como por exemplo *Protégé* [4]. Existem também algumas bibliotecas para *Java* que fazem o mapeamento de XML para OWL, que podem ser utilizadas por *developers.* Mais em detalhe, existe a ferramenta *Chaos Pop* que permite mapear ficheiros em JSON e XML para instâncias de ontologias. Contudo, não temos conhecimento da existência de uma ferramenta que combine estes dois aspetos: a criação de novas instâncias através de uma ontologia bem como o mapeamento de um caso concreto escrito noutra linguagem numa instância de uma ontologia.

Desta forma, de modo a ajudar os utilizadores – como por exemplo, os biólogos - o objetivo deste projecto é desenvolver uma aplicação que tenha uma interface intuitiva que permita esta transformação de dados semiestruturados em dados anotados com ontologias definidas em OWL. Esta interface também teria a possibilidade de anotar valores aos vários conceitos da ontologia ou apenas editar os existentes.

1.1 Descrição do problema e ferramentas já existente

Como referido anteriormente, com o avançar da tecnologia e, consequentemente, da propagação de dados, leva a que esta informação seja descrita em mais que uma linguagem /estrutura, sendo as mais comuns JSON, XML e OWL. Com isto, nesta área, existe a necessidade de transformar dados de um tipo para outro (por exemplo, de XML para OWL ou de JSON para OWL). Desta forma, com o crescimento da popularidade de OWL para a descrição de dados, seria de esperar que já existam alguns *softwares* para a edição e manipulação de OWL [5] de forma a auxiliar a transição de dados escritos noutras linguagens para OWL. Nesta secção iremos falar sobre alguns destes programas e sobre as diferenças entre eles [6]. Iremos começar por comparar 3 programas que têm um maior número de utilizadores dentro da área da semântica *web*: Apollo [7], Protégé e Swoop [8].

1.1.1 Apollo

Apollo é um modelador *user-friendly*, de utilização gratuita e de instalação local. Nesta aplicação, um utilizador pode modelar ontologias com noções básicas de ontologias (classes, relações, instâncias, etc). Também é possível criar novas instâncias a partir das classes presentes nessas ontologias. Alguns dos pontos fortes deste software é o seu validador de tipos que mantém a consistência de tipos durante o processo, bem como o armazenamento das ontologias (em ficheiros). Contudo, este programa é relativamente antigo e carece de uma visualização de dados em grafo, ao contrário dos seus concorrentes. Apollo carece da possibilidade de dar uma experiência multiutilizador aos seus utilizadores, para trabalhos em colaboração com mais do que um utilizador.

1.1.2 Protégé

Protégé é um editor e modelador de ontologias, de utilização gratuita e tem uma vertente local bem como online (*WebProtégé*) [9]. Tem uma arquitetura baseada em *plug-ins* o que deu origem ao desenvolvimento de inúmeras ferramentas relacionadas com semântica web. Implementa um conjunto de estruturas modeladoras de conhecimento e ações que suportam a criação, modelação e manipulação de ontologias, complementadas com inúmeras formas de visualização desses dados. A customização proporcionada aos seus utilizadores é uma das características que torna esta aplicação numa das mais populares na área.

1.1.3 Swoop

Swoop é um browser e também um editor *open-source*, de utilização gratuita e local. Esta ferramenta contém validadores de OWL e oferece várias formas de visualização gráfica de ficheiros em OWL. É composto também por um ambiente de edição, comparação e fusão entre múltiplas ontologias. As suas capacidades de

hyperlinks proporciona uma interface de navegação fácil aos seus utilizadores. Um utilizador pode também reutilizar dados ontológicos externos ao colocar os links para a entidade externa ou importando a ontologia completa, pois não é possível importar ontologias parciais, mas é possível realizar pesquisas por múltiplas ontologias.

Durante a nossa pesquisa sobre outras ferramentas que já existem nesta área, encontrámos também, para além de outros editores com características diferentes dos apresentados acima, algumas bibliotecas que mapeiam XML para OWL - Ontmalizer[1] e JXML2OWL[2]. Estas bibliotecas foram desenvolvidas em Java e estão disponíveis para *developers* usarem também em Java.

Foi nos sugerido utilizar a ferramenta *Chaos Pop* para o nosso projeto. Ao estudarmos esta ferramenta, observámos que esta apresentava vantagens quando comparada com os programas anteriormente mencionados. O *Chaos Pop* permite instanciar ontologias a partir de um ficheiro descritor de uma ontologia em OWL e de um ficheiro de dados em XML, JSON entre outros. Este também está disponível a partir de um servidor remoto com uma API utilizável, o que vai de encontro ao objetivo deste projeto. Apesar destas vantagens, esta ferramenta não continha uma interface gráfica.

Na Tabela 1.1, equiparámos todas as ferramentas descritas acima e comparámos os seus pontos fortes bem como pontos fracos. Podemos observar que, apesar da nossa pesquisa, não encontramos nenhuma aplicação que oferecesse todos as características descritas na tabela, sendo as mais relevantes: versão online como local, mapear XML para OWL[3] e criação de novas instâncias.

Característica - Programa	Criação de novas instâncias	Fácil Utilização	Armazenamento	Interface Intuitiva	Versão Online	Mapeia XML para OWL
Apollo	Sim	Sim	Sim, em ficheiros	Não	Não	Não
Protégé	Sim	Sim	Sim	Sim	Sim	Não
Swoop	Sim	Sim	Sim, em modelos HTML	Sim	Não	Não
Ontomalizer & JXML2OWL	Não	Sim	Não	Não contém interface	Não	Sim
Chaos Pop	Sim	Não	Sim	Não contém interface	Sim	Sim

Tabela 1.1 - Comparação entre várias ferramentas

[1] https://github.com/srdc/ontmalizer

[2] http://jxml2owl.projects.semwebcentral.org/index.html

[3] Mapear XML para OWL refere-se ao processo de realizar uma transformação de uma representação em XML para um documento OWL válido, através da associação de *tags* presentes em XML com conceitos de OWL

1.2 Como a H.O.M.I. se compara com estas ferramentas

A nossa ferramenta visa juntar os pontos fortes destas aplicações numa só aplicação. Na nossa ferramenta (H.O.M.I.) temos disponível uma versão local bem como uma versão *online*, fazendo dela uma aplicação de utilização Híbrida. Todas as funcionalidades estão presentes ambas versões. Tentámos criar uma Interface simples, fácil e intuitiva, ou seja, que funcione da forma que o utilizador espera que funcione. Através da H.O.M.I. um utilizador tem a opção de apenas criar uma nova instância a partir de uma dada ontologia ou então, através de ficheiro semiestruturado descrito em XML, realizar o Mapeamento de conceitos presentes na Ontologia em questão.

1.3 Sinopse

Este relatório está dividido em 7 capítulos.

No Capítulo 2 iremos explicar sucintamente o que é uma ontologia e OWL.

No Capítulo 3 iremos descrever o sistema *Chaos Pop* que fornecedora de uma API na qual utilizamos no nosso programa.

No Capítulo 4 iremos descrever a nossa arquitetura, bem como cada componente e como se relacionam entre si.

No Capítulo 5 iremos resumir a implementação do nosso projeto.

No Capítulo 6 iremos apresentar os nossos testes de escalabilidade realizados a alguns módulos.

No Capítulo 7 iremos realizar uma breve reflexão sobre este projeto, bem como notar algumas melhorias a este.

2. Ontologias e OWL

Neste capítulo apresentamos uma breve descrição sobre o que é uma ontologia e OWL, fazendo a ponte entre ambos e complementando com um exemplo comum entre eles.

2.1 Ontologia

Uma ontologia, na área da ciência da computação, é um modelo de dados que define os termos utilizados para descrever e representa um grupo de ideias ou conceitos dentro de um determinado domínio, bem como descrever as relações que existem entre esses conceitos ou ideias. Estas são usadas em várias áreas da ciência da computação, tais como inteligência artificial e semântica *web*, como uma forma de representar conhecimento sobre essa área, ou sobre um subconjunto dessa área. Uma ontologia descreve: entidades (conjuntos, coleções ou tipo de objetos), indivíduos (instâncias das entidades), atributos (propriedades, características ou parâmetros de entidades) e relacionamentos (entre indivíduos).

Existem vários formatos que podem ser usados para representar uma ontologia: RDF/XML *Syntax*, *Turtle Syntax*, OWL/XML *Syntax*, OWL *Functional Syntax*, entre outros [10].

Na Figura 2.2 podemos observar a representação em grafo de um exemplo de uma ontologia, composta por entidades, identificadas pelos retângulos amarelos "Person", "Father", "Son", "Mother"; e por relações, identificadas pelas setas a tracejado ("hasFather", "hasMother", "hasSon"). Na Figura 2.1 podemos observar uma instância da ontologia ilustrada pela Figura 2.2, onde o indivíduo "João" tem uma relação com o indivíduo "José", denominada por "hasFather" e com o individuo "Maria" denominada por "hasMother". Todos estes indivíduos são instâncias da entidade "Person".

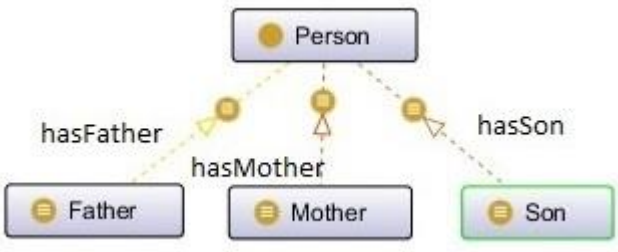

Figura 2.2 - Exemplo de uma ontologia em grafo

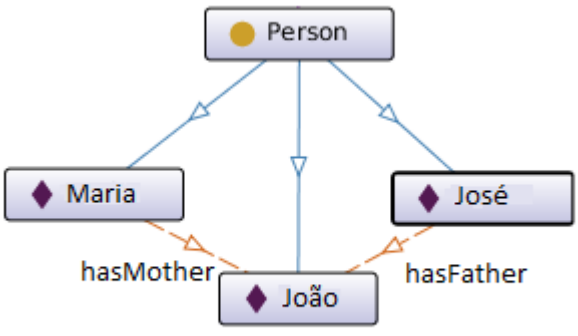

Figura 2.1 - Exemplo de uma instância de uma ontologia

2.2 OWL

OWL (*Ontology Web Language*) é uma linguagem utilizada na criação de ontologias em *Web* e é a linguagem recomendada da W3C (*World Wide Web Consortium*) para a criação de ontologias em Web. Esta linguagem foi desenhada para descrever informação, facilitando assim a sua interpretação por máquinas. Quando comparado com outras linguagens (XML, RDF, etc.), OWL destaca-se por fornecer um vocabulário adicional com uma semântica formal.

O OWL está agrupado em 3 sub-linguagens, sendo cada uma desenhada para um tipo de utilização diferente. OWL *Lite* dá suporte aos utilizadores que necessitam de uma hierarquia de classificação e de restrições de conteúdo simples. OWL DL (*Description Logic*) foi desenhada para suportar a lógica descritiva já existente no segmento de negócio e possui propriedades computacionais necessárias para sistemas de raciocínio. OWL *Full* dá suporte a utilizadores que necessitem do máximo de expressividade e da liberdade sintática do RDF[4] sem garantias computacionais.

Um documento de OWL consiste num cabeçalho da ontologia (ou seja, um *hyperlink* para o documento que contém informação sobre essa ontologia; este cabeçalho é opcional) bem como inúmeras definições de **classe**, **indivíduos** e **propriedades**:

- Uma **classe** fornece um mecanismo de abstração para o agrupamento de recursos com características similares;
- Um **indivíduo** de cada classe é denominado de instância dessa classe, uma representação de um caso concreto de uma classe;
- Existem dois tipos de **propriedades**:
  - ***Object Properties*** refere-se a relações entre instâncias de classes;
  - ***Datatype Properties*** refere-se uma propriedade com valores literais.

  As **propriedades** podem ter relações com outras **propriedades** dos tipos *equivalentProperty* e *inverseOf,* assim como pode conter restrições globais de cardinalidades que podem ser *FunctionalProperty* ou *InverseFunctionalProperty* [3].

Tendo como base as **entidades** presentes na Figura 2.2, a listagem abaixo apresenta a representação em classe destas entidades em OWL:

```
<owl:Class rdf:ID="Person" />
<owl:Class rdf:ID="Mother" />
<owl:Class rdf:ID="Son" />
```

Listagem 2.1 - Exemplo de classes em OWL, utilizando o exemplo da Figura 2.1

[4] https://www.w3.org/RDF/

Apresentamos agora uma representação da relação "hasMother" entre as **classes** "Person" e "Mother", através de uma ***Object Properties***:

```
<owl:ObjectProperty rdf:ID="hasMother">
      <rdfs:domain rdf:resource="#Person"/>
      <rdfs:range rdf:resource="#Mother"/>
<owl:ObjectProperty>
```

Listagem 2.2 - Exemplo de uma object property em OWL

Caso quiséssemos adicionar uma ***Data property*** *"hasFamilyName"* do tipo *String* à classe "Person" de forma a representar o apelido de uma pessoa, a mesma seria definida do seguinte modo:

```
<owl:DatatypeProperty rdf:about="hasFamilyName">
      <rdfs:domain rdf:resource="#Person"/>
      <rdfs:range
      rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty
```

Listagem 2.3 Exemplo de uma data property em OWL

Usando agora o individuo "João" da Figura 2.1, um exemplo da representação deste individuo é a seguinte:

```
<owl:NamedIndividual rdf:ID="João">
     <rdf:type rdf:resource="#Person"/>
     <family:hasMother rdf:resource="#Maria"/>
     <family:hasFather rdf:resource="#José"/>
     <family:hasFamilyName
      rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">
              Silva
    </family:hasFamilyName>
</owl:NamedIndividual>
```

Listagem 2.4 - Exemplo de um indivíduo em OWL

Neste exemplo podemos observar que o individuo "João" é uma instância do tipo *Person*, com uma *data property "hasFamilyName"* do tipo *String* que representa o apelido "Silva" e com duas ***Object Properties****:*

- *"hasMother"* com referência para o individuo "Maria"
- "*hasFather*" com referência para o individuo "José"

3. Chaos Pop

O nosso projeto teve como requisito a integração com a ferramenta Chaos Pop, que permite mapear de forma semiautomática ficheiros de dados semiestruturados para ontologias, com base nos seus termos e conceitos, respetivamente. Foi nos deste modo, apresentada a ferramenta *Chaos Pop* para que pudéssemos utilizá-la.

O *Chaos Pop* fornece um serviço que permite mapear ficheiros semiestruturados de acordo com a descrição de uma ontologia com base em pelo menos dois ficheiros: um OWL referente à uma ontologia (`OntologyFile`) e outro que representa um caso concreto da ontologia em questão (`DataFile`). Neste momento, os ficheiros que são suportados como `DataFile` são de extensões JSON e XML.

Uma vez que não usufruímos de todas as funcionalidades disponíveis no serviço fornecido pelo *Chaos Pop,* neste capítulo apenas realizamos uma breve descrição de algumas das estruturas de dados e dos *endpoints* existentes, assim como um pequeno exemplo de utilização.

3.1 Estruturas de dados

O *Chaos Pop* armazena a informação numa base de dados remota que contém algumas estruturas de dados. Entre as estruturas de dados, existem algumas que é essencial ter conhecimento sobre quais dados armazenam e o que representam para que possa usufruir das funcionalidades do serviço fornecido. São elas:

- *Node:* quando se trata de um ficheiro no formato JSON cada *node* representa os objetos presentes neste. Por outro lado, caso o formato do ficheiro for XML, cada *node* representa um *child element* do *parent.*
- *DataFile:* representa um ficheiro de dados semiestruturados e armazena a informação referente a este na forma de num *array* de *nodes*.
- *OntologyFile:* representa um ficheiro de ontologia, guardando a informação sobre todas as *classes,* ***data properties*** e *object properties* que definem a ontologia em questão.
- *IndividualMapping*: representa o mapeamento de um determinado termo presente em `DataFile` para uma classe especifica presente em `OntologyFile.`
- *Mapping*: representa o mapeamento de um ou mais `DataFile's` para um `OntologyFile.` Este é responsável pela criação do ficheiro de *output* e por isso aglomera as informações necessárias para tal, tais como o nome do ficheiro de *output* bem como os identificadores dos *IndividualMapping's* que irão ser utilizados para realizar este mapeamento.

- *Batch*: utilizado quando se deseja popular uma ontologia com base nos mapeamentos realizados. Este é responsável por gerar um ficheiro de *output* por cada *Mapping* e armazena os *Mapping's* que se desejam executar a fim de popular a ontologia.

3.2 API

De modo a usufruir das funcionalidades existentes no *Chaos Pop*, seja para submeter ficheiros, obter dados referentes aos ficheiros submetidos ou mapear os vários conceitos, é necessário ter conhecimento dos *endpoints* existentes. Todos os *endpoints* apresentados na Listagem 3.1 iniciam com a URL do servidor do *Chaos Pop* http://chaospop.sysresearch.org/chaos/wsapi [11].

METHOD	PATH	BODY
POST	/createIndividualMapping	individualMappingTO: Object
POST	/replaceIndividualMapping	individualMappingTO: Object
POST	/removeIndividualMapping	ids: String
GET	/getAllIndividualMappings	
POST	/addFile	file: InputStream
POST	/getFile	id: String
POST	/removeFile	ids: String
GET	/listDataFiles	
POST	/createMapping	mappingTO: Object
POST	/removeMapping	ids: String
POST	/getAllNodesFromDataFile	id: String
POST	/getNode	id: String
POST	/getOntologyFile	id: String
POST	/removeOntologyFiles	ontologyIds: String
POST	/getOWLClasses	ontologyId: String
POST	/getObjectProperties	ontologyId: String
POST	/getDataProperties	ontologyId: String
GET	/listOntologyFiles	
POST	/createBatch	jsonBatch: String
POST	/removeBatch	batchIds: String

Listagem 3.1 Alguns dos endpoints disponíveis no ChaosPop

3.3 Exemplo de utilização

Para realizar um mapeamento, primeiro, são necessários os ficheiros envolvidos nesse, um ficheiro de dados `DataFile` e outro de ontologia `OntologyFile`. O ficheiro OWL usado contém as informações que são apresentadas abaixo no formato *Turtle* [10] (Listagem 3.2) e o ficheiro de dados semiestruturados é o da Listagem 3.3

```
@prefix : <http://chaospop.sysresearch.org/ontologies/family.owl#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://chaospop.sysresearch.org/ontologies/family.owl> .

<http://chaospop.sysresearch.org/ontologies/family.owl> rdf:type owl:Ontology ;

#    Object Properties

###  http://chaospop.sysresearch.org/ontologies/family.owl#hasPartner
:hasPartner rdf:type owl:ObjectProperty ;
         owl:inverseOf :isPartnerIn .

###  http://chaospop.sysresearch.org/ontologies/family.owl#isPartnerIn
:isPartnerIn rdf:type owl:ObjectProperty .

#    Data properties

###  http://chaospop.sysresearch.org/ontologies/family.owl#hasBirthYear
:hasBirthYear rdf:type owl:DatatypeProperty ,
                  owl:FunctionalProperty ;
           rdfs:domain :Person ;
           rdfs:range xsd:integer .

###  http://chaospop.sysresearch.org/ontologies/family.owl#hasFirstGivenName
:hasFirstGivenName rdf:type owl:DatatypeProperty ;
                rdfs:subPropertyOf :hasName ;
                rdfs:domain :Person ;
                rdfs:range xsd:string .

###  http://chaospop.sysresearch.org/ontologies/family.owl#hasName
:hasName rdf:type owl:DatatypeProperty ;
       rdfs:domain :Person ;
       rdfs:range xsd:string .

#    Classes

###  http://chaospop.sysresearch.org/ontologies/family.owl#Person
:Person rdf:type owl:Class .
```

Listagem 3.2 - Exemplo da descrição de uma ontologia em turtle

```
<family>
  <member>
    <name>
      <given>Maria Goretti</given>
      <surname>Fernandes</surname>
    </name>
    <birth_year>1960</birth_year>
   <partner_name>João Antero Cardoso</partner_name>
  </member>
  <member>
    <name>
      <given>João Antero</given>
      <surname>Cardoso</surname>
    </name>
    <birth_year>1958</birth_year>
    <partner_name>Maria Goretti Fernandes</partner_name>
  </member>
</family>
```

Listagem 3.3 - Exemplo de um ficheiro de dados semiestruturado descrito na linguagem XML

Para obter um ficheiro correspondente ao da ontologia mostrada acima, mas que corresponda a um caso concreto da mesma, o utilizador deve realizar as correspondências dos vários termos existentes no ficheiro semiestruturado, aos conceitos da ontologia. Os mapeamentos a serem realizados são:

Semiestruturado termos	Ontologia conceitos
member	Person
given	hasFirstGivenName
given e surname	hasName
partner_name	hasPartner
birth_year	hasBirthYear

Tabela 3.1 - Alguns mapeamentos do ficheiro semiestruturado para ontologia

Após realizar todos os mapeamentos do ficheiro semiestruturado, irá ser gerado o ficheiro do caso concreto da ontologia. Na Listagem 3.4 mostrada abaixo está apresentado o ficheiro gerado no formato *turtle*.

```
@prefix : <http://chaospop.sysresearch.org/ontologies/cardosofamily.owl##> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix family:  <http://chaospop.sysresearch.org/ontologies/family.owl#>
@base <http://chaospop.sysresearch.org/ontologies/cardosofamily.owl#> .

<http://chaospop.sysresearch.org/ontologies/minifamily.owl#>
rdf:type owl:Ontology ;
owl:imports <http://chaospop.sysresearch.org/ontologies/family.owl#> .

#    Individuals

###  http://chaospop.sysresearch.org/ontologies/cardosofamily.owl#JoãoAnteroCardoso
:JoãoAnteroCardoso a owl:NamedIndividual,
<http://chaospop.sysresearch.org/ontologies/family.owl#Person> ;
family:hasPartner:MariaGorettiFernandes ;
family:hasBirthYear:1958 ;
family:hasFirstGivenName:"João Antero".

###
http://chaospop.sysresearch.org/ontologies/cardosofamily.owl#MariaGorettiFernandes
:MariaGorettiFernandes a owl:NamedIndividual,
<http://chaospop.sysresearch.org/ontologies/family.owl#Person>
family:hasPartner:JoãoAnteroCardoso ;
family:hasBirthYear:1960 ;
family:hasFirstGivenName:"Maria Goretti".
```

Listagem 3.4 - Caso concreto de uma ontologia após ser mapeada utilizando o Chaos Pop

4. Arquitetura do H.O.M.I

Neste capítulo apresentamos a arquitetura da aplicação *web*, descrevemos qual a funcionalidade de cada módulo presente nesta, assim como a forma como estes interagem entre si e as tecnologias utilizadas na implementação da aplicação no geral.

4.1 Descrição

A Figura 4.1 mostra a arquitetura da aplicação *web*, na qual está representada a interação entre todas as camadas existentes. Existem quatro camadas de acesso:

1. Views
2. Routes
3. Services
4. Data access

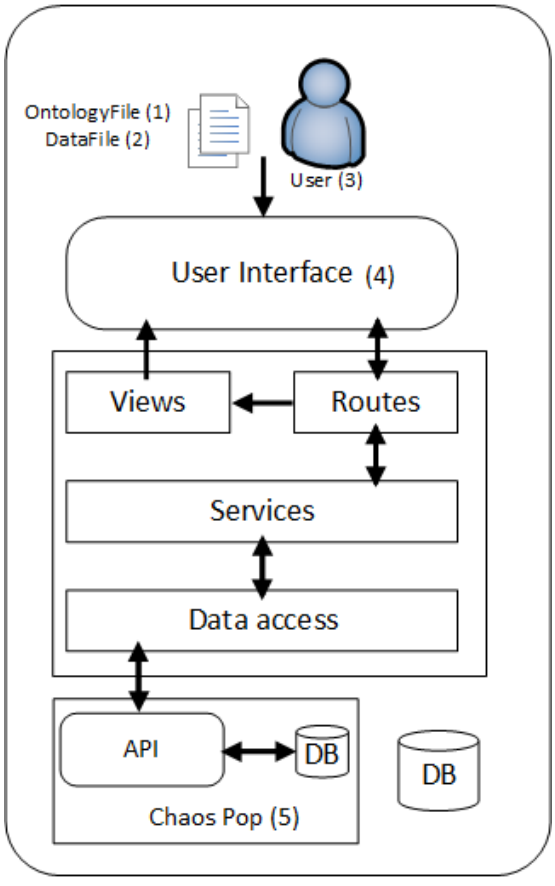

Figura 4.1 - Arquitetura da aplicação

A interação inicia-se quando o utilizador (3) insere um ou mais ficheiros com a definição de uma ontologia (1) e, opcionalmente, um ou mais ficheiros de dados semiestruturados (2). Estes ficheiros irão ser submetidos à API do *Chaos Pop* (5). De seguida irá ser gerada uma interface gráfica (4) onde o utilizador poderá anotar novos dados aos vários conceitos presentes nos ficheiros que descrevem uma ontologia (1) ou mapear os termos dos ficheiros semiestruturados (2) com os conceitos dos ficheiros da ontologia (1). No final deste processo, é gerado um novo ficheiro OWL que contém um caso concreto dos dados descritos nos `OntologyFile's`.

4.2 Componentes

A aplicação foi implementada por camadas (Figura 4.1), na qual cada camada apenas comunica com a superior ou inferior. Esta implementação proporciona uma maior facilidade na manipulação das funcionalidades. As camadas existentes são:

- ***controllers:*** contém todos os *endpoints* possíveis de serem acedidos através da *user interface* e comunica com *views* e *services*;
- ***services:*** camada intermédia entre *controllers* e *data access*, contém toda a lógica necessária para a execução das operações disponíveis;
- ***data access:*** engloba todo o acesso aos dados. Seja este acesso na API *Chaos Pop* ou na nossa própria base de dados;
- ***views:*** inclui todas as representações visuais utilizadas em *user interface.*

4.3 Tecnologias

Em cada uma das camadas da arquitetura apresentada anteriormente existem diferentes módulos. Nestes estão implementadas funções, usufruindo de algumas tecnologias. Todas as tecnologias utilizadas foram previamente escolhidas tendo em conta as funcionalidades que cada uma delas disponibilizam, assim como o objetivo final do projeto.

4.3.1 Node.js

Uma plataforma de software que é usada para construir aplicativos de redes escaláveis. O *Node.js* usa o *JavaScript* como linguagem de *script* e alcança rendimentos altos através de um *loop* de eventos, no qual interpreta uma única *thread*, as requisições de forma assíncrona em vez de sequenciais, não permitindo bloqueios [11].

Já tivemos experiências em outras unidades curriculares com esta tecnologia. Concluímos que fornece uma maneira fácil de construir uma aplicação *web*, bem como é suportada por outras tecnologias que a utilizam para criar aplicações *desktop*, o que nos facilitou no desenvolvimento do projeto para a versão desktop.

4.3.2 Express

Equivale a um *framework* rápido, flexível e minimalista para Node.js. *Express* fornece um conjunto robusto de recursos para desenvolver aplicações *web*, assim como um grande número de métodos utilitários HTTP e *middleware* para criar uma API rápido e fácil [12].

Por já termos experiências com esta tecnologia e pelo facto da sua utilização ser fácil, decidimos implementar a nossa aplicação *web* e todos os *endpoints* disponíveis com base nesta tecnologia.

4.3.3 D3.js

Consiste numa biblioteca JavaScript que permite vincular dados arbitrários ao *Document Object Model (DOM)* e em seguida aplicar transformações nos dados usando diferentes padrões *web* como HTML, SVG e CSS [13].

Decidimos optar por tal pelo facto de existirem vários exemplos de utilização na documentação da mesma que podiam ser adaptados para a nossa necessidade de obter de uma forma simples e prática uma representação em árvore dos dados referentes ao ficheiro de entrada `DataFile`.

4.3.4 Electron

É uma biblioteca *open source* utilizada para criar aplicações *desktop* multiplataformas com tecnologias *web* (HTML, JavaScript e CSS). *Electron* combina *Chromium* e *Node.js* em um único fio de execução e as aplicações podem ser empacotadas para Mac, Windows e Linux [14].

Uma vez que a nossa aplicação foi projetada para ter duas vertentes, uma *web* e outra *desktop*, tivemos de optar por uma tecnologia que nos permitisse aproveitar o máximo possível do código desenvolvido para a aplicação *web* na criação da versão *desktop.* O que o *Electron* nos proporciona é exatamente isso, reutilizar o que já estava implementado em JavaScript sem a necessidade de aprender a usar uma nova tecnologia no desenvolvimento da aplicação *desktop*.

4.3.5 MongoDb

MongoDB é um banco de dados que armazena dados em documentos flexíveis semelhantes a JSON, o que significa que os campos de cada documento podem variar de um para outro e a estrutura de dados pode ser alterada ao longo do tempo [15].

Apesar de podermos desfrutar da base de dados do *Chaos Pop,* todos os acessos a essa teriam um grande custo em tempo de execução decorrente das várias operações que são necessárias realizar. Por isso decidimos criar a nossa própria base de dados

usando o MongoDb. Optámos por uma base de dados documental pois já tínhamos experiência de unidades curriculares anteriores a utilizar base de dados deste tipo com aplicações desenvolvidas em *javaScript*. Por outro lado, ao existir um modelo entidade relação, não nos comprometemos aos tipos já pré-definido nesse mesmo modelo, dando-nos a liberdade de ir modelando, alterando e adaptando as nossas estruturas de dados consoante as necessidades do nosso projeto, ao logo do seu desenvolvimento.

Escolhemos o MongoDb por ser uma tecnologia bastante utilizada no mercado de trabalho e pelo facto de permitir ser flexível na criação dos documentos e a execução de *queries* para obter esses documentos.

5. Implementação

Neste capítulo descrevemos todas estruturas de dados existentes na base de dados documental e o seu respetivo intuito, os *endpoints* presentes na API que permite a comunicação entre *client-side* e *server-side,* a organização do código em vários módulos e a interação com o utilizador na aplicação *web.*

5.1 Estruturas de dados

A nossa base de dados é documental e está dividida em quatro *collections*, onde cada uma armazena dados estruturados de forma distinta. São as *collections:*

1. DataFiles
2. OntologyFiles
3. IndividualMappings
4. Populates

Cada documento presente nestas *collections* contém o campo `_id` que é o identificador único gerado pelo *MongoDb*, para além de outros campos que são descritos abaixo.

5.1.1 Data Files

Esta *collection* armazena dados referentes aos ficheiros de *input* `DataFile`. A constituição de um ficheiro presente nesta é:

```
{
      _id: ObjectId,
      name: String,
      chaosid: String,
      nodes: Array
}
```

- `name:` nome do ficheiro;
- `chaosid`: identificador no ficheiro do servidor do *Chaos Pop*;
- `nodes`: contém todos os nodes que constituem a árvore representativa dos dados.

5.1.2 Ontology Files

Esta *collection* armazena dados referentes aos `OntologyFile's`. A constituição de um ficheiro presente nesta é:

```
{
      _id: ObjectId,
      name: String,
      chaosid: String,
      classes: Array,
      dataProperties: Array,
      objectProperties: Array
}
```

- `name:` nome do ficheiro;
- `chaosid`: identificador no ficheiro do servidor do *Chaos Pop*;
- `classes`: classes que pertencem ficheiro de ontologia;
- `dataProperties`: propriedades do tipo *data* que pertencem ao ficheiro de ontologia;
- `objectProperties`: propriedades do tipo *object* que pertencem ao ficheiro de ontologia.

5.1.3 Individual Mappings

Esta *collection* contém dados que representam o mapeamento de um determinado indivíduo e de suas propriedades, entre um `DataFile` e um `OntologyFile`, que são denominados de *IndividualMapping*. Também é utilizada para armazenar a criação de indivíduos apenas a partir de uma `OntologyFile`. O nosso *IndividualMapping* é semelhante ao utilizado pelo *Chaos Pop* e tem a seguinte estrutura:

```
{
      _id: ObjectId,
      tag: String,
      nodeId: String,
      owlClassIRI: String,
      ontologyFileId: String,
      dataFileId: String,
      specification: Boolean,
      objectProperties: Array,
      dataProperties: Array,
      annotationProperties: Array
}
```

- `tag`: *tag* do *node* presente no `DataFile` que estamos a mapear;
- `nodeId`: id do *Chaos Pop* correspondente ao *node* do `DataFile` que estamos a mapear;
- `owlClassIRI`: IRI da classe que estamos a mapear, contida no `OntologyFile`;

- `ontologyFileId`: id presente na *collection OntologyFiles* correspondente ao `OntologyFile` que queremos mapear;
- `dataFileId`: id presente na *collection DataFiles* correspondente ao `DataFile` que queremos mapear,
- `specification`: uma *flag* que indica se este *IndividualMapping* irá alterar algum outro já existente;
- `individualName`: metadados que indicam qual a propriedade de dados que irá ser utilizada para o nome deste indivíduo;
- `objectProperties`: cada elemento deste *array* é composto por: IRI da propriedade no `OntologyFile` e metadados dos *nodes* (no `DataFile`) envolvidos nesta propriedade;
- `dataProperties`: cada elemento deste *array* é composto por: IRI da propriedade no `OntologyFile`, um *pair* composto pelos metadados dos *nodes* (no `DataFile`) envolvidos nesta propriedade, bem como o tipo da mesma (*string, integer, etc*);
- `annotationProperties`: cada elemento deste *array* é composto por: tipo de anotação (*label*, *comment*, etc) e metadados dos *nodes* (no `DataFile`) envolvidos nesta propriedade.

No geral, a constituição das estruturas presentes nesta *collection* é a mencionada acima. Porém quando o *IndividualMapping* se diz respeito a um mapeamento e não a criação de um individuo apenas, este contém mais um campo denominado `indTree`, que representa a subárvore extraído do `DataFile` correspondente ao *IndividualMapping* em questão.

5.1.4 Populates

Nesta *collection* estão presentes estruturas que podem ser consideradas como uma sessão, uma vez que em cada uma delas está presente um aglomerado de identificadores indispensáveis em toda a interação com o utilizador. As estruturas podem ser usadas numa população de ontologia com (*populate with data*) ou sem dados semiestruturados (*populate without data*). Os campos em comum nos dois casos são:

```
{
      _id: ObjectId,
      ontologyFiles: Array,
      indMappings: Array,
      chaosid: String,
      batchId: String,
      outputFileId: String,
      outputFileName: String
}
```

- `ontologyFiles`: *array* com dados sobre todos os ficheiros de ontologia que deseja popular;

- `indMappings`: *array* de *IndividualMappings* pertencentes ao mapeamento;
- `chaosid`: identificador do servidor do *Chaos Pop* referente ao *Mapping;*
- `batchId`: identificador do servidor do *Chaos Pop* referente ao *Batch*;
- `outputFileId`: identificador do servidor do *Chaos Pop* referente ao ficheiro de *output,* quando esse é gerado;
- `outputFileName`: nome escolhido pelo utilizador para ser atribuído ao ficheiro de *output*.

Quando se trata de um *populate with data*, além da descrição acima também contém o campo `dataFiles` que representa todos os ficheiros de dados semiestruturados envolvidos no mapeamento.

5.2 Endpoints

A comunicação entre o *client-side* e o *server-side* é feita através de pedidos HTTP. Desta forma, o servidor disponibiliza uma lista de *endpoints* para facilitar esta comunicação. Estes *endpoints* estão definidos nos *routes*, agrupados pela sua finalidade. Por exemplo, todos os *endpoints* relacionados com *Populate's* poderão ser encontrados no populates-routes.

METHOD	**PATH**	**BODY**
POST	/dataFile	File
POST	/ontologyFile	File
GET	/dataFile	
GET	/ontologyFile	
DELETE	/dataFile/:id	
DELETE	/ontologyFile/:id	
POST	/map/individual	individualMapping: Object
PUT	/map/individual/:id	individualMapping: Object
PUT	/map/individual/:id/name	individualName: Array
PUT	/map/individual/:id/properties/annotation	annotationProps: Array
PUT	/map/individual/:id/properties/data	dataProps: Array
PUT	/map/individual/:id/properties/object	objectProps: Array
GET	/map/individual/:id/properties/annotation/view	
GET	/map/individual/:id/properties/data/view	
GET	/map/individual/:id/properties/object/view	
DELETE	/map/individual/:id	
DELETE	/map/individual/:id/properties/:pid	

POST	/individual	individual: Object
PUT	/individual/:id/properties/annotation	annotationProps: Array
PUT	/individual/:id/properties/data	dataProps: Array
PUT	/individual/:id/properties/object	objectProps: Array
GET	/individual/:id/properties/annotation/view	
GET	/individual/:id/properties/data/view	
GET	/individual/:id/properties/object/view	
DELETE	/individual/:id	
DELETE	/individual/:id/properties/:pid	
POST	/map	mapping: Object
GET	/map/:id	
DELETE	/map/:id	
POST	/populate	populate: Object
PUT	/populate/:id/output	
GET	/populate	
DELETE	/populate/:id	
GET	/populate/data/:id	
GET	/populate/data/:id/tree	
GET	/populate/data/:id/mapping	
GET	/populate/data/:id/individual/:ind	
GET	/populate/data/:id/individual/:ind/tree	
PUT	/populate/nondata/:id/finalize	indMappings: Array
GET	/populate/nondata/:id	
GET	/populate/nondata/:id/mapping	
GET	/populate/nondata/:id/individual/:ind	

Listagem 5.1 - Endpoints disponíveis na H.O.M.I.

5.3 Organização do código

Neste subcapítulo explicaremos como funciona as camadas de acesso presentes em *routes*, *services* e *data-access* assim como outros requesitos referentes a implementação que achamos relevantes.

5.3.1 Módulos routes, services e data-access

Como mencionado anteriormente, o servidor está dividido em 3 camadas: *routes*, *services* e *data-access*. Apresentamos agora um esquema mais completo que indica as relações entre estas camadas:

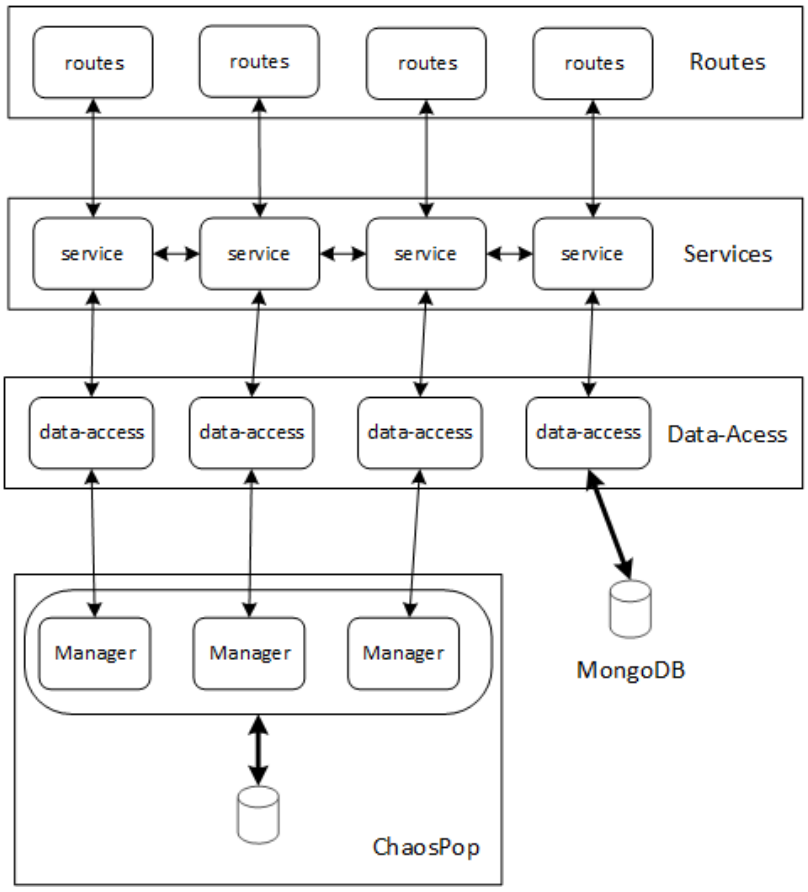

Figura 5.1 - Camadas de acesso e seus respetivos módulos

Através da arquitetura em camadas garantimos que toda a lógica da aplicação está nos *services*. Cada *routes* apenas comunica com o seu serviço e os serviços comunicam entre si bem como com vários *data-access,* dependendo da ação a ser realizada. Os *routes* apenas respondem a pedidos HTTP, retiram a informação necessária dos mesmos (caso se trate de um POST, PUT ou DELETE) e de seguida essa informação é passada ao *service* correspondente para processar aquele pedido que, por sua vez, pode precisar de outros *services* e de um ou vários *data-access*.

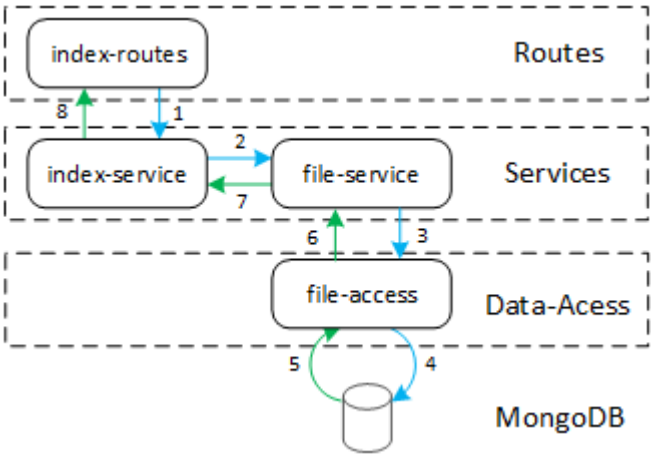

Figura 5.2 - Exemplo de acesso às várias camadas

Utilizando o exemplo da Figura 5.2, quando é chamado o *endpoint* '/, este está localizado no *index-routes*. Para realizar o *render* da página inicial, é necessário ir buscar os ficheiros já existentes para o utilizador ter a opção de selecionar um ficheiro que já submeteu anteriormente. Então o *index-router* irá comunicar com o *index-service* (1), que por sua vez irá precisar do *file-service* (2). O *file-service* (3) é responsável por processar toda a informação relacionada com ficheiros, então será utilizado o *file-access* (4) para ir buscar os ficheiros já existentes. Após a obtenção dos ficheiros (5), o *file-access* irá devolver os ficheiros ao *file-service* (6) e este ao *index-service* (7). Após algumas validações e lógica correspondente, este *service* irá devolver os resultados obtidos ao *index-routes* (8) e assim este já pode realizar o *render* da página inicial com toda a informação necessária.

A comunicação com a nossa base de dados é feita apenas na camada *data-access*, no *mongodb-access*. Já a comunicação com o *Chaos Pop* é feita através dos restantes módulos presentes em *data-access*, sendo que cada um destes *data-access* comunica com o seu *Manager* correspondente no *Chaos Pop*, com exceção do *file-access* que comunica com *FileManager* e *OntologyManager*, pois achamos que não valia a pena separar esse acesso do nosso lado.

5.3.2 Módulo utils

Neste módulo implementámos algumas funcionalidades que seriam necessárias para a lógica da aplicação, mas não se encaixava em nenhum dos *services* criados. Como por exemplo, o *parser* de propriedades de acordo com a linguagem que o *Chaos Pop* consiga interpretar.

A seguir damos uma breve explicação sobre as funções implementadas neste módulo, assim como alguns pequenos exemplos.

Properties Parser

Criámos um *parser* para transformar as propriedades numa metalinguagem definida pelo *Chaos Pop* e que o mesmo consiga interpretar. Esta metalinguagem define um determinado comando e o caminho desde o nó do *individual mapping* até ao nó ou nós que contêm os valores que queremos definir nessa propriedade. O comando que utilizamos é sempre o comando "In Specific Child" que indica que o valor que queremos está num nó *child* do nó associado ao *individual mapping*.

De seguida apresentamos um ficheiro de dados (Listagem 5.2) que iremos utilizar para uma exemplificação do *parser*, em conjunto com uma propriedade de cada tipo que irá representar o *input* deste *parser* (Listagem 5.3, Listagem 5.4 e Listagem 5.5).

```
<family>
  <member>                              //nodeId: 1
    <name>                              //nodeId: 2
      <given> João </given>             //nodeId: 3
      <surname> Silva </surname>        //nodeId: 4
    </name>
    <mother > Maria </mother >          //nodeId: 5
    <nickname> Jonhy </nickname>        //nodeId: 6
  </member>
</family>
```

Listagem 5.2 - Exemplo de um ficheiro de dados em XML

```
AnnotationProperty: {
      annotation: 'label'
      toMapNodeIds: ['6'],
      id: 'property1'
}
```

Listagem 5.3 - Annotation property antes de ser submetida ao parser

```
DataProperty: {
      owlIRI: '#hasName'
      toMapNodeIds: ['3', '4'],
      id: 'property2'
}
```

Listagem 5.4 - Data property antes de ser submetida ao parser

```
ObjectProperty: {
      owlIRI: '#hasMother'
      toMapNodeIds: ['5'],
      id: 'property3'
}
```

Listagem 5.5 - Object property antes de ser submetida ao parser

Para exemplificar os dados apresentados nas listagens anteriores, utilizamos o *individual mapping* abaixo:

```
IndividualMapping: {
      tag: ‘member’
      dataFileIds: [Listagem 5.2],
      owlClassIRI: ‘#Person’,
      nodeId: ‘1’,
      annotationProperties: [‘property1’],
      dataProperties: [‘property2’],
      objectProperties: [‘property3’]
}
```

Listagem 5.6 - Exemplo de um individual mapping definido pelo ficheiro de dados presente em Listagem 5.2 antes de passar pelo parser

Ao receber as propriedades contidas no *individual mapping* apresentado acima, o *parser* irá realizar os seguintes passos:

1. Começar a sua pesquisa pelos *child nodes* do nó ao qual corresponde o *nodeId* presente no *individual mapping*, para este caso é o nó com a *tag member* (*nodeId* 1).
2. Para cada propriedade, irá pesquisar no *child nodes* do *nodeId* até que encontre pelos identificadores presentes no campo `toMapNodeIds` da propriedade em questão.
3. Ao encontrar o caminho pretendido, irá construir a metalinguagem que o *ChaosPop* sabe interpretar.

Por exemplo, para *a object property* (Listagem 5.5) o nó com o identificador 5 é um *child node* direto de *nodeId*, então a metalinguagem é a seguinte: ***.inspecificchild-mother***.

No fim, o *IndividualMapping* terá sofrido as alterações aplicadas pelo *parser* e ficará definido do seguinte modo (os dados a *bold* são os *metadados* gerados pelo *parser*):

```
Individual Mapping: {
    tag: 'member',
     dataFileIds: '169',
     owlClassIRI: '#Person',
     annotationProperties: {
       label: '.inspecificchild-nickname'
     },
     objectProperties: {
        #hasMother: '.inspecificchild-mother '
     },
     dataProperties: {
        #hasName: ['.inspecificchild-name-given;
                    .inspecificchild-name-surname', 'String']
     }
}
```

Listagem 5.7 - Exemplo de um individual mapping, depois de passar pelo parser

List To Tree

Uma das funcionalidades do *Chaos Pop*, como mencionada no Capítulo 4, é a habilidade desta ferramenta em transformar `DataFile` em *nodes*. Conseguimos aceder a estes *nodes* através do *endpoint /getAllNodesFromDataFile*. Este *endpoint* retorna todos os *nodes* existentes no ficheiro em questão e seus respetivos *children*. Para facilitar a visualização para o utilizador, criamos um *parser* (*list-to-tree*) que trasnforma esta lista de *nodes* numa árvore n-ária.

Fake File Maker

De forma permitir que um utilizador anote novos dados aos conceitos de uma ontologia, inserindo diretamente os dados em vez de mapear de um ficheiro para outro, damos a opção ao utilizador de realizar um mapeamento sem dados (*Populate without data*).

Para continuarmos a poder utilizar as funcionalidades do *Chaos Pop*, torna-se necessário a existência de um ficheiro de dados semiestruturados, uma vez que este está implementado para funcionar apenas se tiver os dois ficheiros: um de ontologia e outro de dados. Desta forma, a solução que implementamos deixa que um utilizador vá inserindo os valores que pretende anotar e no fim, quando este criar um *Mapping* com os *Individual Mappings* criados, esta função gera um ficheiro XML 'falso' com os dados que o utilizador inseriu e submete-o ao *Chaos Pop*. Para tal, utilizamos o módulo *XMLWriter* do npm para nos auxiliar a gerar este documento.

Para os dois Individual Mappings *presentes em Listagem 5.8 e Listagem 5.9, o ficheiro gerado pelo* fake file maker *é o ficheiro presente na*

Listagem 5.10.

```
Individual Mapping: {
    _id: '5b1fef788b45ea1c34a34db2',
    individualName: 'João'
    owlClassIRI: '#Person',
    annotationProperties: {
       id: '_BJs5JO6gX',
       label: 'Johny'
    },
    objectProperties: {
       id: '_rJXXeualX,
       #hasMother: 'Maria'
    },
    dataProperties: {
       id: '_ Hgxlj1tQo',
       #hasName: ['João Silva','String']
    }
 }
```

Listagem 5.8 – Indivíduo João criado pela inserção de dados do utilizador

```
Individual Mapping: {
    _id: ‘5b46a35732724f1aac75fcc1’,
    individualName: 'Maria'
}
```

Listagem 5.9 - Indivíduo Maria criado pela inserção de dados do utilizador

```
<XML>
  <root>
    <_5b1fef788b45ea1c34a34db2>
      <individualName>João</individualName>
      <_BJs5JO6gX>Johny </_BJs5JO6gX>
      <_rJXXeualX>Maria</_rJXXeualX>
      <_Hgxlj1tQo>João Silva </_Hgxlj1tQo>
    </_5b1fef788b45ea1c34a34db2>
    <_5b1fef788b45ea1c34a34db2>
      <individualName>Maria</individualName>
    </_5b1fef788b45ea1c34a34db2>
  </root>
</XML>
```

Listagem 5.10 - Ficheiro de dados gerado pelo fake file maker

5.3.3 Módulo public e views

Como utilizámos a tecnologia *Express.js*, por convenção dessa tecnologia, todos os ficheiros estáticos que serão usados na aplicação estão presentes em *public* (imagens, *scripts* e *styles*). Os *scripts* presentes em *public* servem, em sua maioria, para manipular os objetos presentes nos documentos contidos em *views*.

Apesar de termos cogitados outras tecnologias na implementação do *front-end,* como por exemplo o *React*, optamos pelo *Handlebars* [18] na criação dos *templates* e pelo DOM (*Document Object Model*) convencional na manipulação dos objetos presentes nesses. A nossa decisão deveu-se ao facto de termos demorado algum tempo até compreender todas as funcionalidades necessárias do *Chaos Pop* e também porque não queríamos gastar mais tempo ao aprender uma tecnologia nova para que fosse utilizada no projeto.

5.5 Interação com o utilizador

A interface do H.O.M.I é bastante intuitiva na sua interação e fornece algumas funcionalidades. Na página inicial (Figura 5.3) podemos adicionar ficheiros de dados (1) e de ontologia (2) assim como selecionar os existentes para criar *populates* (3) ou apenas navegar pelos ficheiros e *populates* existentes (4).

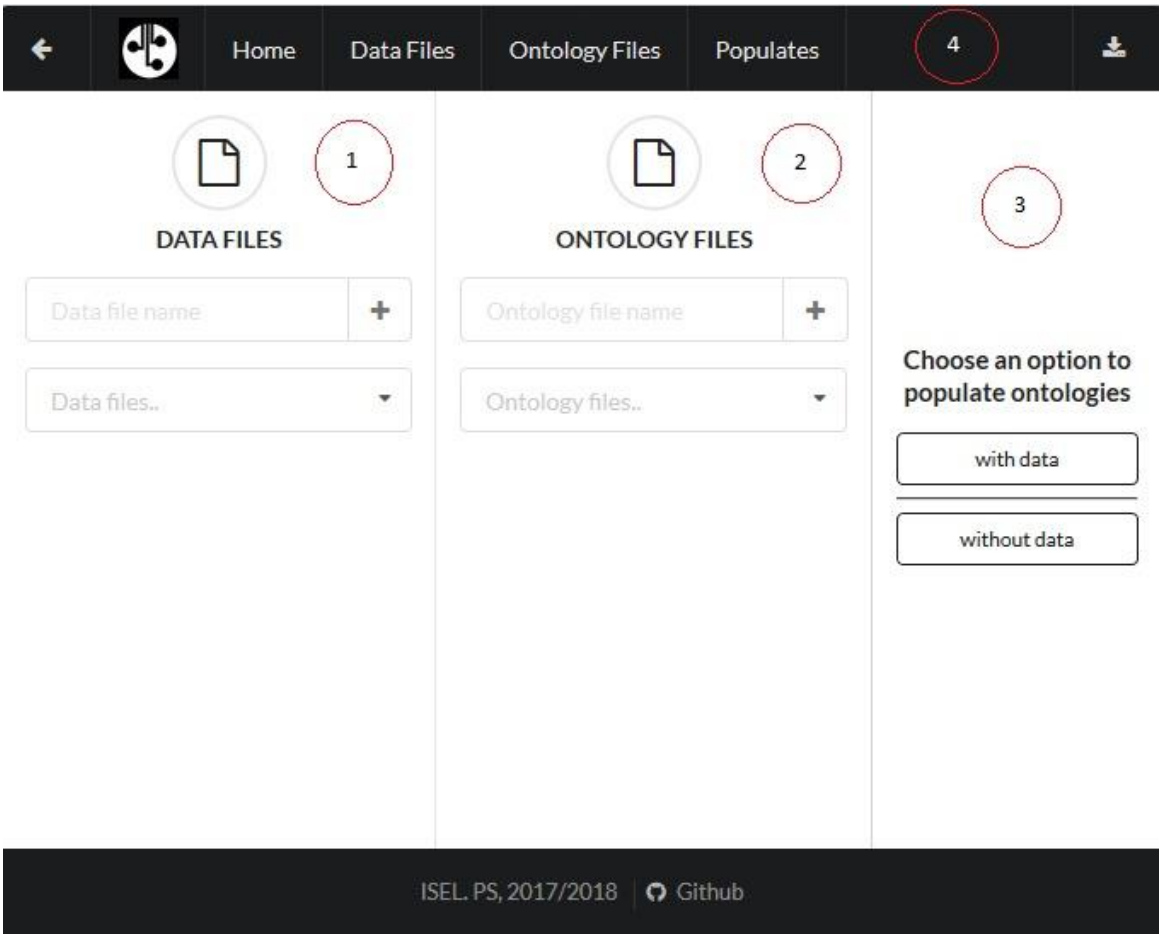

Figura 5.3 - Página inicial da aplicação

Populate with data: nesta página (Figura 5.4) é possível observar a árvore gerada por um ficheiro de dados semiestruturados (ao pressionar o botão) (5), criar indivíduos (6) ou apenas observar os existentes (7).

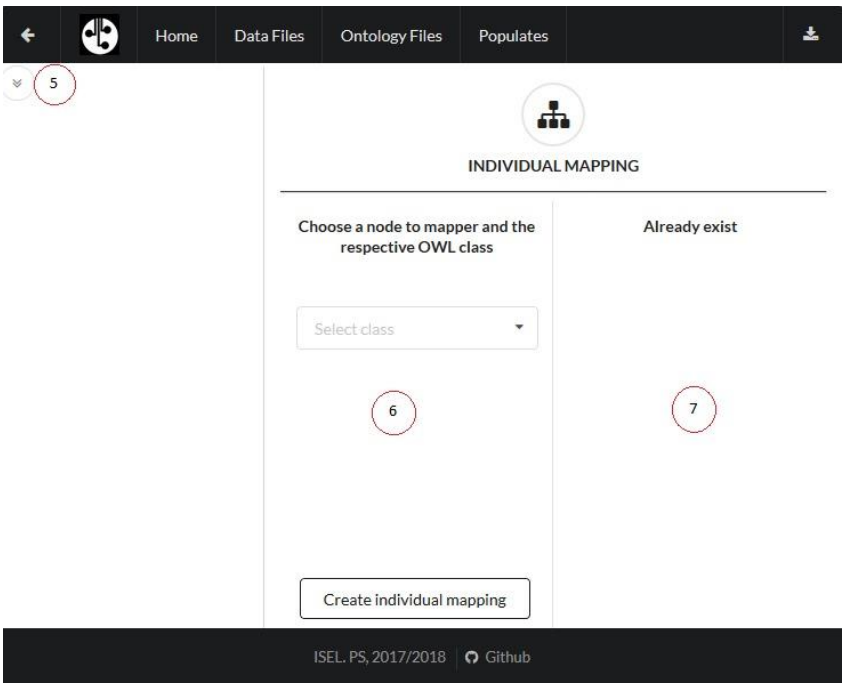

Figura 5.4 - Página de um populate with data

Quando se cria um *individual mapping* por meio de um *populate with data* em sua página (Figura 5.5) realizamos a visualização da sub-árvore referente a este (8), adicionamos propriedades (9) e salvamos este no *Chaos Pop* (10).

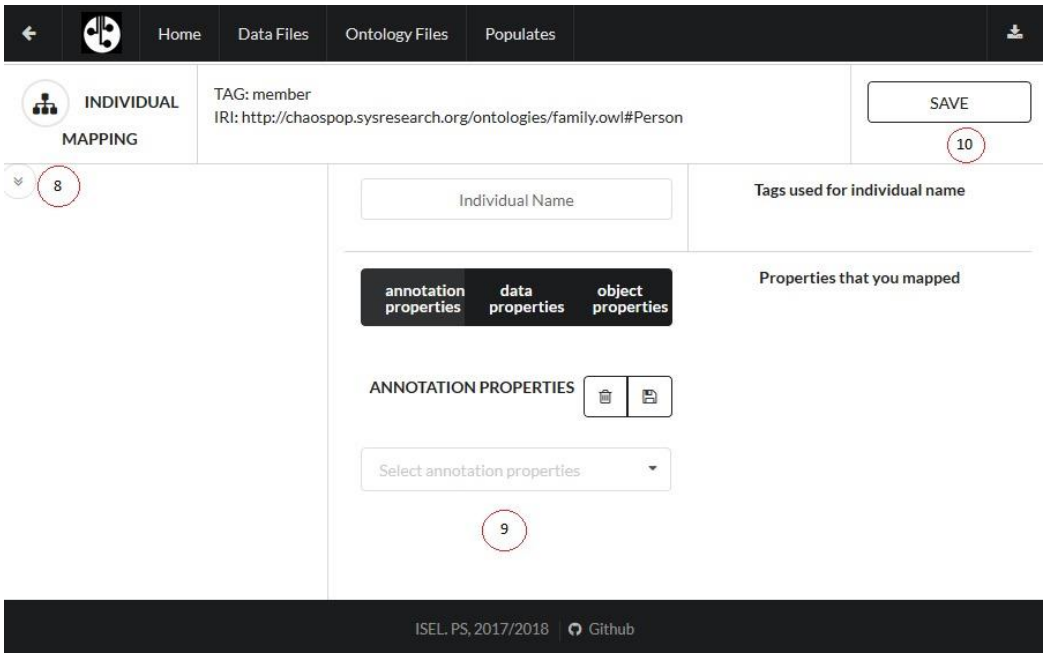

Figura 5.5 - Página de um individual mapping criado por meio de um populate with data

Populate without data: na página de um *populate* deste tipo também é possível a criação de indivíduos (11) e observar os existentes.

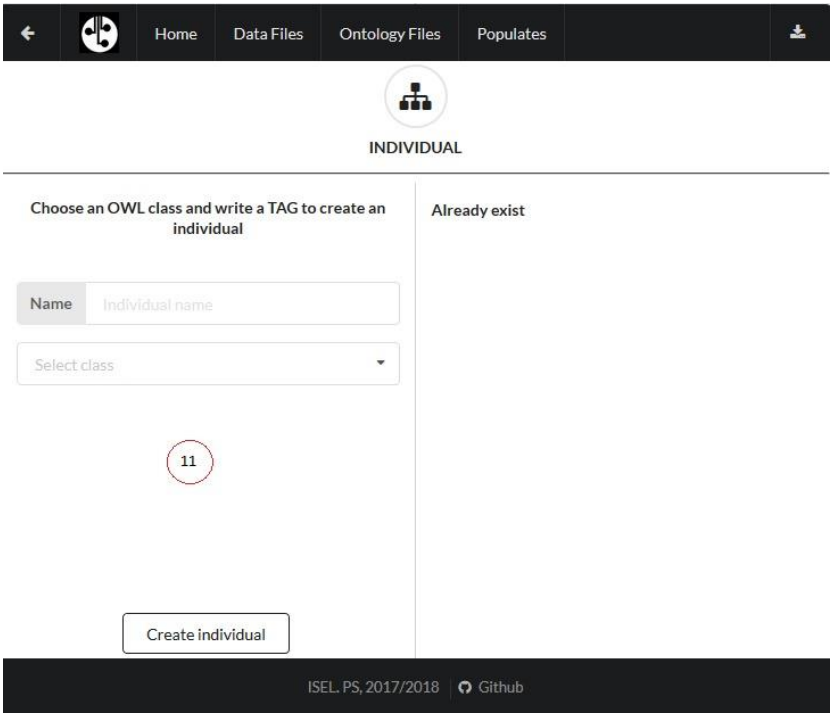

Figura 5.6 - Página de um populate without data

E na página gerada a partir do indivíduo criado (Figura 5.7) pode-se mapear as propriedades (12).

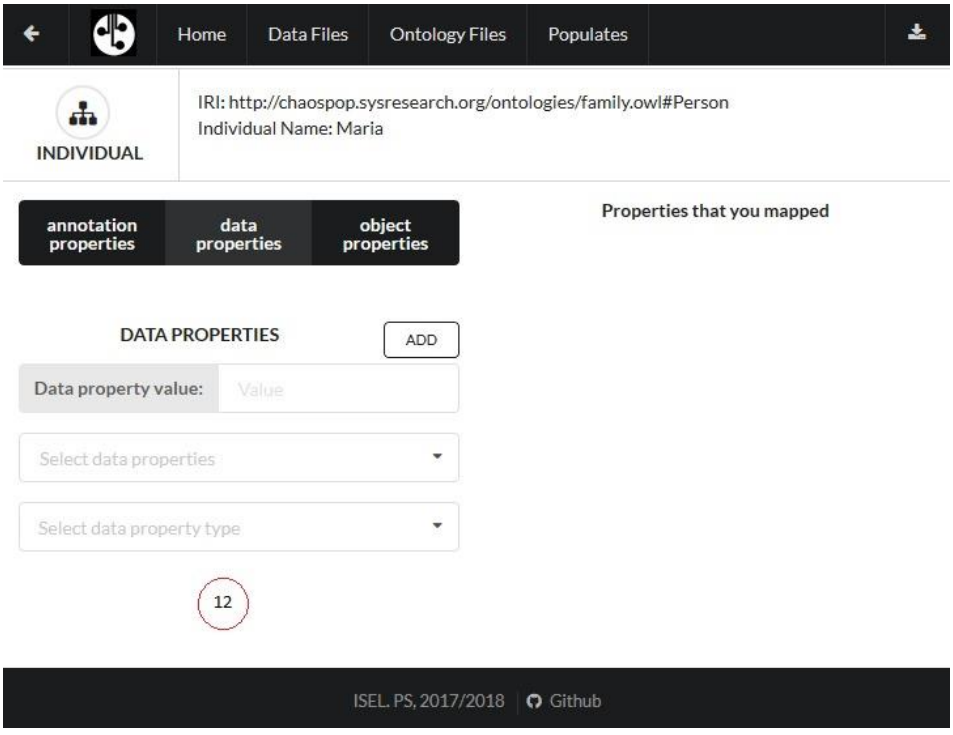

Figura 5.7 - Página de um indíviduo criado por meio de um populate without data

Criar *mapping* e obter ficheiro de *output*: após popular as ontologias desejadas, seja com ou sem o uso de ficheiro de dados semiestruturados, ao navegar até os *Populates* (Figura 5.8) podemos criar um *mapping* (14) para depois conseguir obter o ficheiro de output (13). A opção de obter o ficheiro de *output* apenas aparece quando já existe algum *mapping* criado para o *populate* em questão.

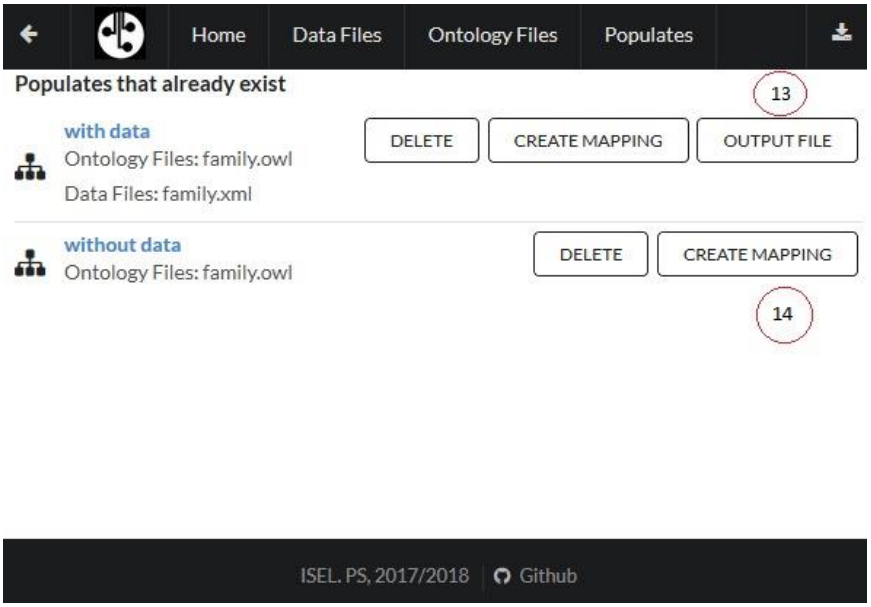

Figura 5.8 - Página de populates que permite criar mapping e obter o ficheiro de output

A ação de criar um *mapping* (Figura 5.9) consiste em selecionar os indivíduos nas quais deseja adicionar ao ficheiro de output. Quando estiver num *populate with data* apenas aparecem os indivíduos que foram salvos no *Chaos Pop* (ver Figura 5.5, ponto 10).

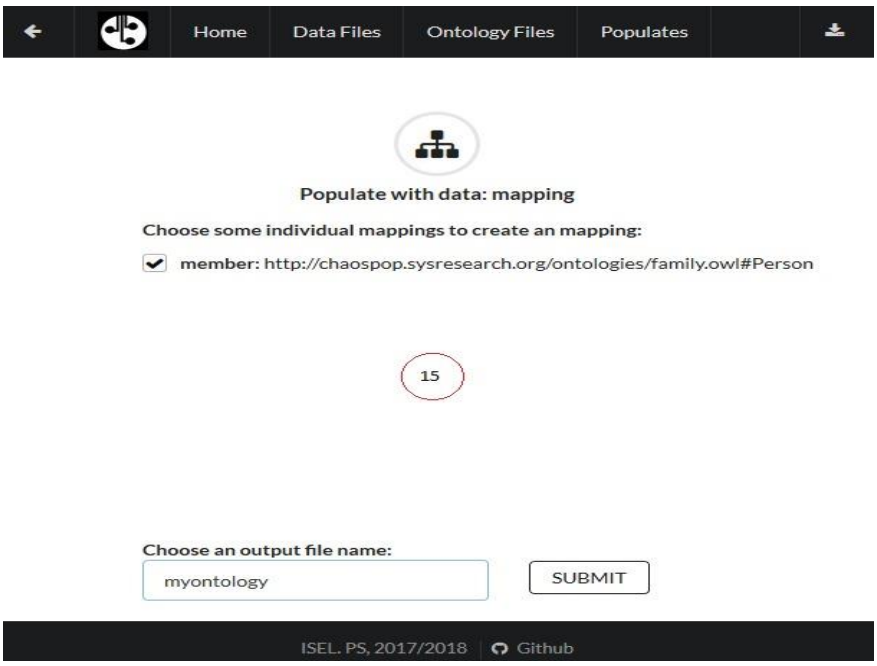

Figura 5.9 - Página da criação de um mapping

6. Avaliação Experimental

Para testar a eficiência de algumas funcionalidades da nossa aplicação, foram realizados alguns testes de escalabilidade sobre as mesmas que visam mostrar os tempos de execução. Escolhemos realizar estes testes às seguintes funcionalidades:

- Acesso à base de dados, através da escrita de documentos.
- *Properties Parser*, que é utilizado no mapeamento com um *DataFile.*
- *Fake File Maker,* que é utilizado no mapeamento sem dados, ou seja, em modo Editor.

Todos os testes foram executados numa máquina com as seguintes características:

- Marca e modelo: ASUS – X541UV
- CPU: Intel Core i7-6500U CPU @ 2.50GHz 2.50 GHz
- Memória (RAM): 16,0 GB (15.9 GB utilizáveis)

6.1 Teste ao acesso à base de dados

Como descrito em capítulos anteriores, utilizámos o *MongoDB* para a nossa base de dados documental. Desta forma, realizámos testes à escrita de novos documentos utilizando o módulo realizado para o acesso à base de dados, em particular, a função *sendDocToDb* (`/data-access/mongodb-access/sendDocToDb`).

Criámos um gerador de documentos que usa o módulo *async* do npm para enviar estes documentos assincronamente para a base de dados, e realizamos dois testes:

1. Variámos o número de documentos gerados entre 1 e 100.000 sempre com 5 linhas.
2. Geramos 5 documentos com linhas a variar entre 1 e 100.000.

Para cada teste foram realizadas 5 iterações. Os tempos de execução apresentados nas tabelas abaixo são uma média dessas 5 iterações.

Nº de documentos	Nº de linhas	Média de tempos em milissegundos
5	1	1029
5	10	1029
5	100	1031
5	1000	1034
5	5000	1063
5	10000	1101
5	100000	1557

Tabela 6.1 - Média de tempos de execução, mantendo o número de documentos e variando as linhas de cada documento

Nº de documentos	Nº de linhas	Média de tempos em milissegundos
1	5	206
10	5	1035
100	5	1138
1000	5	2236
5000	5	5849

Tabela 6.2 - Média de tempos de execução, variando o número de documentos e mantendo as linhas de cada documento

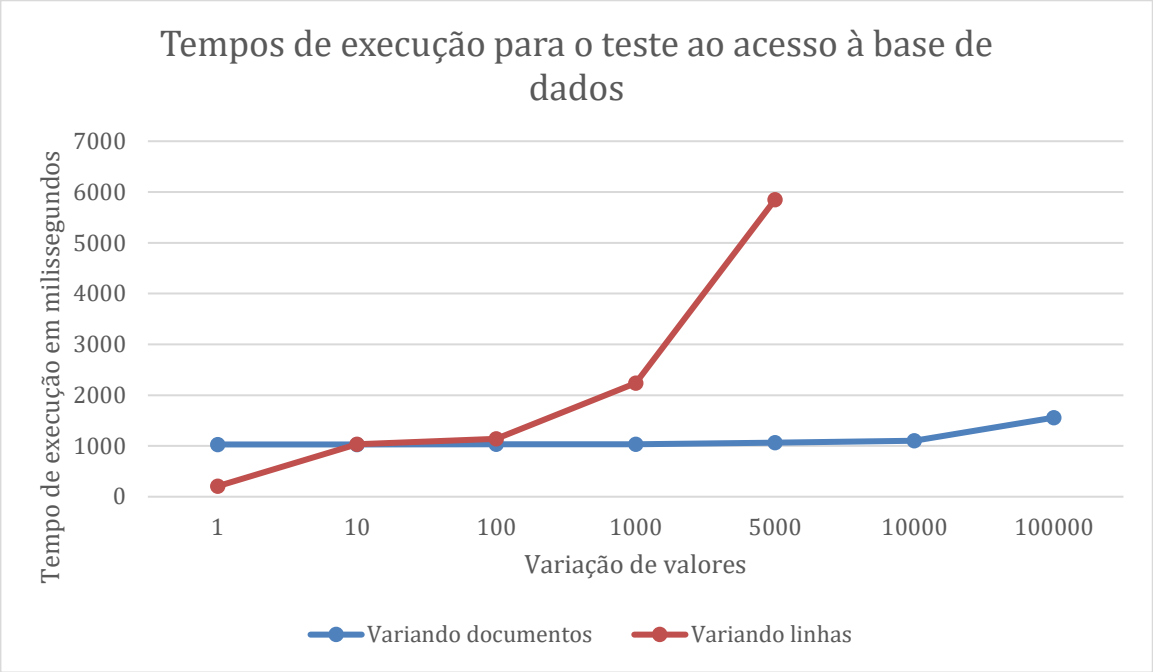

Figura 6.1 - Representação em gráfico das tabelas Tabela 6.1 e Tabela 6.2

Como podemos observar pelo gráfico da Figura 6.1, o que influencia o tempo de execução ao escrever ficheiros na base de dados é o número de ficheiros e não a dimensão dos mesmos. É de notar também que a não existência de valores a partir de 5.000 documentos pois o *MongoDB* encerrava a ligação à nossa aplicação devido a sua *pool size*. Notámos também que a partir das 100.000 linhas, o *MongoDb* também lançava exceção devido ao seu *buffer size*.

6.2 Teste ao *Properties Parser*

Como descrito em capítulos anteriores, criámos um *properties parser* para transformar as propriedades de cada *individual mapping* em metalinguagem que o *Chaos Pop* consiga interpretar. Desta forma, realizámos testes em que testamos a escalabilidade deste *parser* a cada tipo de propriedade individualmente, ou seja, às funções *parseAnnotationProperties*, *parseDataProperties* e ao *parseObjectProperties*. No final destes testes, notámos que a diferença de valores entre eles não era significativa o

suficiente para apresentarmos todos esses valores. Desta forma, apenas apresentamos os valores para *Data Properties,* pois a partir deste conseguimos inferir os restantes.

Criámos um gerador de *individual mappings* e utilizámos o módulo *async* do npm para que cada iteração fosse feita assincronamente e realizamos dois testes:

1. Variámos o número de propriedades por *individual mapping*.
2. Variámos o número de *individual mappings*, mantendo 5 propriedades para cada um.

Os tempos extraídos e apresentados são uma média das iterações.

Nº de Individual Mappings	Nº de Data Properties	Média de tempos em milissegundos
5	1	1035
5	10	1036
5	100	1046
5	1000	1138
5	10000	1988
5	100000	10446

Tabela 6.3 - Média de tempos de execução, mantendo o número de Individual Mappings e variando as Data Properties de cada Individual Mapping

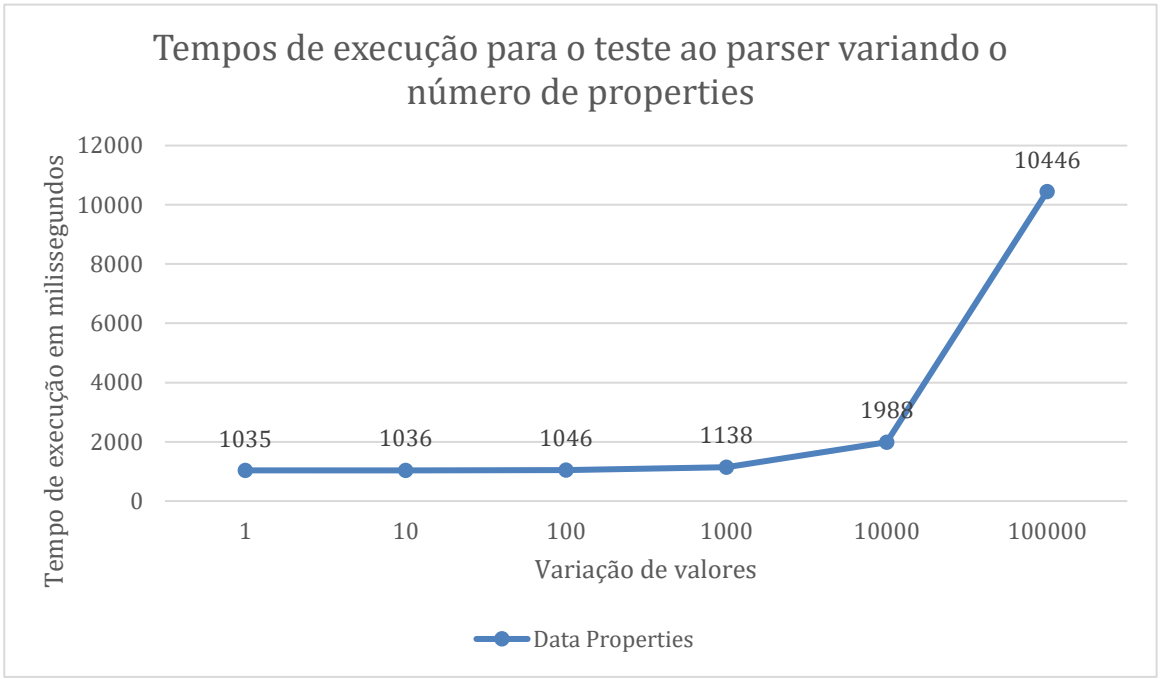

Figura 6.2 - Representação em gráfico da tabela Tabela 6.3

Nº de Individual Mappings	Nº de Data Properties	Média de tempos em milissegundos
1	5	1035
10	5	1040
100	5	1172
1000	5	2415

Tabela 6.4 - Média de tempos de execução, variando os Individual Mappings e mantendo as Data Properties de cada

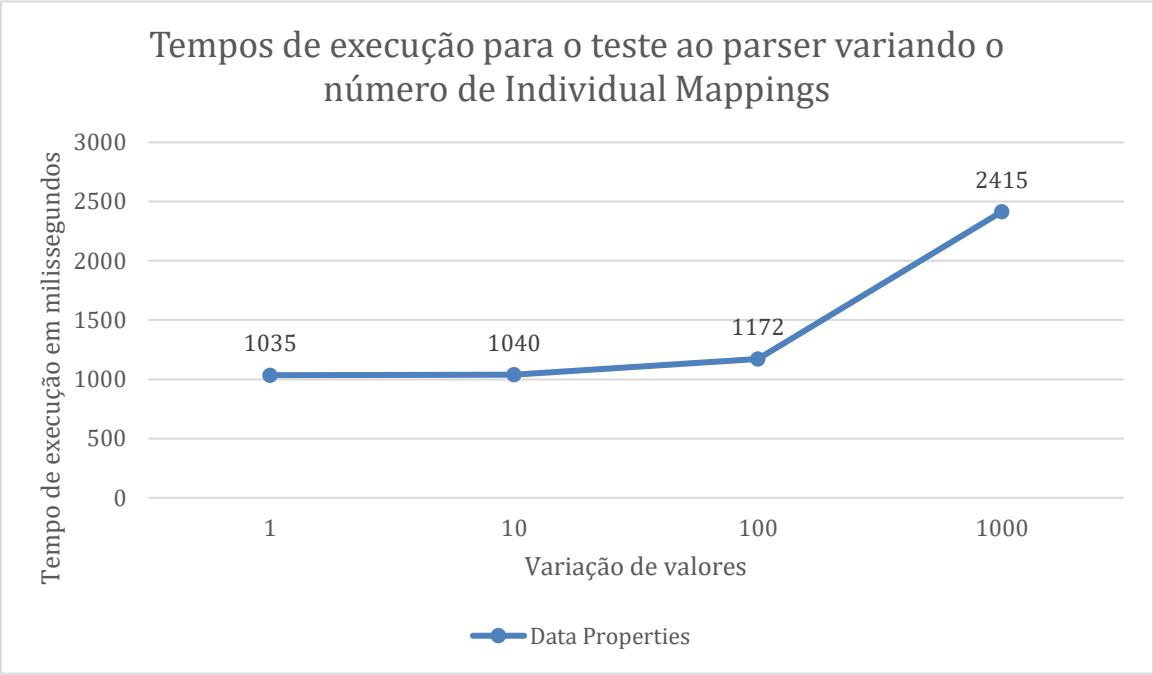

Figura 6.3 - Representação em gráfico da tabela Tabela 6.4

Como podemos observar pelos gráficos presentes nas Figura 6.2 e Figura 6.3, o que influencia o tempo de execução deste *parser* é o número de *Individual Mappings* e não o número de propriedades de cada um. A utilização do módulo *async* diminui drasticamente o tempo de execução pois desta forma todos os pedidos são iniciados ao mesmo tempo, não ficando dependentes da finalização do pedido anterior. É de notar que na Tabela 6.4 e na Figura 6.3, a variação de valores termina nos 1000 *individual mappings* pois este *parser* não conseguiu processar as metas seguintes: 10000 e 100000.

6.3 Teste ao Fake File Maker

Como descrito em capítulos anteriores, criámos um *fake file maker* que recebe uma lista de *individual mappings* para gerar *DataFiles*, usado no mapeamento sem dados. Desta forma, realizámos testes em que testamos a escalabilidade deste *Maker*, localizado no módulo */utils* da nossa aplicação.

Criámos um gerador de listas de *individual mappings* e utilizámos o módulo *async* do npm para que cada iteração fosse feita assincronamente.

Realizámos três testes:

1. Variámos o número de *individual mapping* por lista, mantendo o número de propriedades de cada um.
2. Variámos o número de propriedades, mantendo o número de listas e o número de *individual mappings*.
3. Variámos o número de listas, mantendo o número de *individual mappings* e o número de propriedades para cada um deles.

As variações ocorrem entre 1 e 100.000 e os tempos extraídos e apresentados são uma média das iterações.

Nº de Listas	Nº de Individual Mappings	Nº de propriedades por individual mapping	Média de tempos em milissegundos
1	5	5	4
10	5	5	9
100	5	5	52
1000	5	5	478
10000	5	5	5036

Tabela 6.5 - Média de tempos de execução, variando o número de listas e mantendo os Individual Mappings e as propriedades de cada um

Nº de Listas	Nº de Individual Mappings	Nº de propriedades por individual mapping	Média de tempos em milissegundos
5	1	5	4
5	10	5	7
5	100	5	34
5	1000	5	210
5	10000	5	2147

Tabela 6.6 - Média de tempos de execução, mantendo o número de listas, variando os Individual Mappings e mantendo as propriedades de cada um

Nº de Listas	Nº de Individual Mappings	Nº de propriedades por individual mapping	Média de tempos em milissegundos
5	5	1	5
5	5	10	8
5	5	100	24
5	5	1000	175
5	5	10000	1943

Tabela 6.7 - Média de tempos de execução, mantendo o número de listas e de Individual Mappings e variando as propriedades de cada um

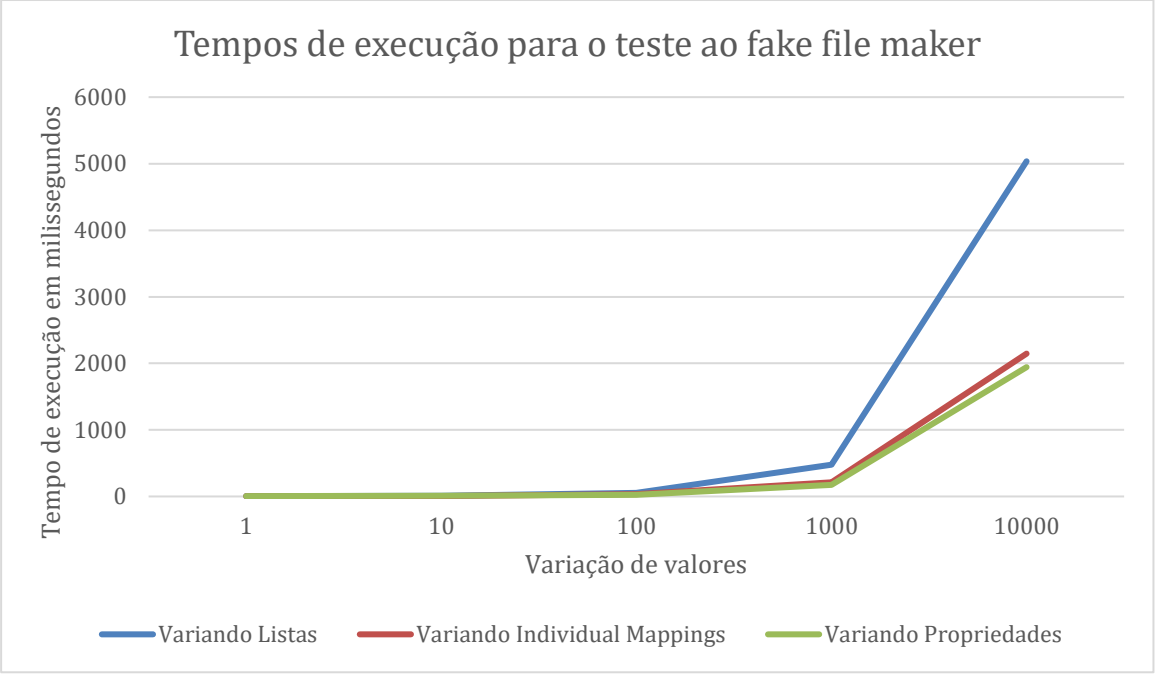

Figura 6.4 - Representação em gráfico das tabelas Tabela 6.5, Tabela 6.6 e Tabela 6.7

Através da Figura 6.4 conseguimos observar que, mesmo não utilizando assincronismo no *File Maker*, este consegue ter um tempo de execução bastante rápido, conseguindo gerar dez mil ficheiros em pouco mais de 5 segundos. O número de *individual mapping* e o número de propriedades de cada um obtiveram tempos bastaste semelhantes, o que nos leva a concluir que não é o tamanho dos objetos que influencia o tempo de execução, mas sim o número de ficheiros produzidos.

7. Notas Finais

Este projeto permitiu perceber o papel importante que as ontologias têm vindo a ganhar nos dias atuais, uma vez que permite a normalização de conceitos e estabelecer as relações semânticas existentes entre eles. Apesar de estar a ser cada vez mais predominante, ainda não existia uma ferramenta que permitisse realizar o mapeamento de dois ficheiros a fim de instanciar uma ontologia conhecida e o H.O.M.I tenta suprir essa necessidade.

Existem duas funcionalidades que, caso houvesse mais tempo, seriam úteis à sua implementação:

1. Na versão *web,* criar a noção de usuários para assim poder restringir o acesso aos ficheiros inseridos por cada um, pois todos os utilizadores neste momento podem aceder aos ficheiros que foram submetidos pelos outros.

2. Na versão *desktop,* tentamos fornecer aos utilizadores um pouco mais de privacidade ao utilizar a base de dados local do H.O.M.I, mas isto não muda em relação ao *Chaos Pop,* uma vez que os dados submetidos ficam sempre no servidor do mesmo. Então, um aspeto que seria acréscimo ao H.O.M.I nesta versão é poder estar vinculado a uma versão local do *Chaos Pop*.

Com a implementação dos dois pontos apresentados acima, transmitiríamos mais segurança aos utilizadores no que diz respeito a inserção de dados sensíveis, pois eles teriam a garantia de que esses apenas estariam presentes na sua máquina.

Esperamos com este projeto facilitar a todos os utilizadores da mesma, especialmente os bioinformáticos, que foram o publico alvo da H.O.M.I.

Referências

[1] "W3C Semantic Web - Ontology," [Online]. Available: https://www.w3.org/standards/semanticweb/ontology. [Acedido em 19 03 2018].

[2] "TechRepublic," [Online]. Available: https://www.techrepublic.com/article/the-benefits-of-the-web-ontology-language-in-web-applications/. [Acedido em 12 07 2018].

[3] "W3C Semantic Web - OWL reference," [Online]. Available: https://www.w3.org/TR/owl-ref/. [Acedido em 19 03 2018].

[4] "Protégé," [Online]. Available: https://protege.stanford.edu/. [Acedido em 15 03 2018].

[5] "W3C Semantic Web - Ontology Editors," [Online]. Available: https://www.w3.org/wiki/Ontology_editors. [Acedido em 21 04 2018].

[6] "Comparison Some of Ontology," pp. http://www.ef.uns.ac.rs/mis/archive-pdf/2013%20-%20No2/MIS2013-2-4.pdf, 21 04 2018.

[7] "Apollo," [Online]. Available: http://apollo.open.ac.uk/. [Acedido em 22 04 2018].

[8] "Github," [Online]. Available: https://github.com/ronwalf/swoop. [Acedido em 23 04 2018].

[9] "Protégé - Web," [Online]. Available: https://protege.stanford.edu/products.php#web-protege. [Acedido em 23 04 2018].

[10] "W3C Semantic Web - Turtle," [Online]. Available: https://www.w3.org/TR/turtle/. [Acedido em 26 05 2018].

[11] "Node.js," [Online]. Available: https://nodejs.org/en/about. [Acedido em 19 03 2018].

[12] "Express," [Online]. Available: https://expressjs.com/. [Acedido em 19 03 2018].

[13] "Data-Driven Documents," [Online]. Available: https://d3js.org/. [Acedido em 19 03 2018].

[14] "Electron," [Online]. Available: https://electronjs.org/. [Acedido em 19 03 2018].

[15] "MondoDB," [Online]. Available: https://www.mongodb.com/. [Acedido em 21 04 2018].

[16] C. E. T. S. Jamie Taylor, Programming the Semantic Web, 2009.

[17] J. R. Wilson, Node.js the Right Way: Practical, Server-side JavaScript that Scales, 2013.

[18] "W3C Semantic Web - Meta formats," [Online]. Available: https://www.w3.org/standards/webarch/metaformats. [Acedido em 26 04 2018].

[19] "W3C Semantic Web - Semantic Web Tools," [Online]. Available: https://www.w3.org/wiki/SemanticWebTools#Semantic_Web_Development_Tools:. [Acedido em 21 04 2018].

Logotipo feito por: Joana Antunes (joanasantosantunes@hotmail.com)

8 Anexo A - Ontologia descritora de ferramentas em OWL

Um dos pontos obrigatórios deste projeto era gerar uma nova ontologia para descrever ferramentas utilizadas em *workflows*. Desta forma, utilizando os conceitos de CWL, gerámos uma nova ontologia de representa uma parte da descrição de ferramentas, tendo em conta que CWL é uma linguagem bastante completa. Gerámos esta ontologia, designada por cwlOntology, através do auxilio da ferramenta Protégé.. Esta ontologia foi gerada nos seguintes formatos: RDF/XML Syntax; Turtle Syntax; OWL/XML Syntax; OWL Functional Syntax; JSON-LD. Estes ficheiros estão disponíveis no repositório da aplicação. Abaixo está o conteúdo do ficheiro da ontologia, em OWL/XML Syntax.

```
<?xml version="1.0"?>
<rdf:RDF xmlns="urn:absolute:cwlOntology"
    xml:base="urn:absolute:cwlOntology"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
   <owl:Ontology rdf:about="urn:absolute:cwlOntology"/>



   <!--

///////////////////////////////////////////////////////////////////////////
//////////
   //
   // Object Properties
   //

///////////////////////////////////////////////////////////////////////////
//////////
    -->



   <!-- urn:absolute:cwlOntology#hasCommandInputParameter -->
```

```
  <owl:ObjectProperty
rdf:about="urn:absolute:cwlOntology#hasCommandInputParameter">
    <owl:inverseOf
rdf:resource="urn:absolute:cwlOntology#isCommandInputParameterOf"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:range>
      <owl:Restriction>
        <owl:onProperty
rdf:resource="urn:absolute:cwlOntology#hasCommandInputParameter"/>
        <owl:minQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:m
inQualifiedCardinality>
        <owl:onClass
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
      </owl:Restriction>
    </rdfs:range>
  </owl:ObjectProperty>



  <!-- urn:absolute:cwlOntology#hasCommandLineBinding -->

  <owl:ObjectProperty
rdf:about="urn:absolute:cwlOntology#hasCommandLineBinding">
    <owl:inverseOf
rdf:resource="urn:absolute:cwlOntology#isCommandLineBindingOf"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
    <rdfs:range rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
  </owl:ObjectProperty>



  <!-- urn:absolute:cwlOntology#hasCommandOutputParameter -->

  <owl:ObjectProperty
rdf:about="urn:absolute:cwlOntology#hasCommandOutputParameter">
    <owl:inverseOf
rdf:resource="urn:absolute:cwlOntology#isCommandOutputParameterOf"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:range>
      <owl:Restriction>
        <owl:onProperty
rdf:resource="urn:absolute:cwlOntology#hasCommandLineBinding"/>
```

```
            <owl:minQualifiedCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:m
inQualifiedCardinality>
            <owl:onClass
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
          </owl:Restriction>
        </rdfs:range>
    </owl:ObjectProperty>



    <!-- urn:absolute:cwlOntology#hasOutputBinding -->

    <owl:ObjectProperty rdf:about="urn:absolute:cwlOntology#hasOutputBinding">
        <owl:inverseOf rdf:resource="urn:absolute:cwlOntology#isOutputBindingOf"/>
        <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
        <rdfs:range rdf:resource="urn:absolute:cwlOntology#CommandOuputBinding"/>
    </owl:ObjectProperty>



    <!-- urn:absolute:cwlOntology#isCommandInputParameterOf -->

    <owl:ObjectProperty
rdf:about="urn:absolute:cwlOntology#isCommandInputParameterOf">
        <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
        <rdfs:range rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    </owl:ObjectProperty>



    <!-- urn:absolute:cwlOntology#isCommandLineBindingOf -->

    <owl:ObjectProperty
rdf:about="urn:absolute:cwlOntology#isCommandLineBindingOf">
        <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
        <rdfs:range rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
    </owl:ObjectProperty>



    <!-- urn:absolute:cwlOntology#isCommandOutputParameterOf -->
```

```
  <owl:ObjectProperty
rdf:about="urn:absolute:cwlOntology#isCommandOutputParameterOf">
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
    <rdfs:range rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
  </owl:ObjectProperty>



  <!-- urn:absolute:cwlOntology#isOutputBindingOf -->

  <owl:ObjectProperty rdf:about="urn:absolute:cwlOntology#isOutputBindingOf">
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandOuputBinding"/>
    <rdfs:range
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
  </owl:ObjectProperty>



  <!--

////////////////////////////////////////////////////////////////////////
//////////
  //
  // Data properties
  //

////////////////////////////////////////////////////////////////////////
//////////
   -->



  <!-- urn:absolute:cwlOntology#CommandLineBinding -->

  <owl:DatatypeProperty
rdf:about="urn:absolute:cwlOntology#CommandLineBinding">
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#CommandLineToolProps -->
```

```
  <owl:DatatypeProperty
rdf:about="urn:absolute:cwlOntology#CommandLineToolProps">
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#CommandOutputBinding -->

  <owl:DatatypeProperty
rdf:about="urn:absolute:cwlOntology#CommandOutputBinding">
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandOuputBinding"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#CommandParameter -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#CommandParameter">
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#FileProperties -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#FileProperties"/>



  <!-- urn:absolute:cwlOntology#baseCommand -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#baseCommand">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineToolProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#basename -->
```

```
  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#basename">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#checksum -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#checksum">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#class -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#class">
    <rdfs:subPropertyOf
rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#Requirements"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:comment>always required</rdfs:comment>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#contents -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#contents">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#coresMax -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#coresMax">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
```

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#coresMin -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#coresMin">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#cwlVersion -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#cwlVersion">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineToolProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:range>
      <rdfs:Datatype>
        <owl:oneOf>
          <rdf:Description>
            <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#List"/>
            <rdf:first>draft-2</rdf:first>
            <rdf:rest>
              <rdf:Description>
                <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#List"/>
                <rdf:first>draft-3</rdf:first>
                <rdf:rest>
                  <rdf:Description>
                    <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#List"/>
                    <rdf:first>draft-3.dev1</rdf:first>
                    <rdf:rest>
                      <rdf:Description>
                        <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#List"/>
                        <rdf:first>draft-3.dev2</rdf:first>
```

```
                        <rdf:rest>
                          <rdf:Description>
                            <rdf:type rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#List"/>
                            <rdf:first>draft-3.dev3</rdf:first>
                            <rdf:rest>
                              <rdf:Description>
                                <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
                                <rdf:first>draft-3.dev4</rdf:first>
                                <rdf:rest>
                                  <rdf:Description>
                                    <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
                                    <rdf:first>draft-3.dev5</rdf:first>
                                    <rdf:rest>
                                      <rdf:Description>
                                        <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
                                        <rdf:first>draft-4.dev1</rdf:first>
                                        <rdf:rest>
                                          <rdf:Description>
                                            <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
                                            <rdf:first>draft-4.dev2</rdf:first>
                                            <rdf:rest>
                                              <rdf:Description>
                                                <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
                                                <rdf:first>draft-4.dev3</rdf:first>
                                                <rdf:rest>
                                                  <rdf:Description>
                                                    <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>
                                                    <rdf:first>v1.0</rdf:first>
                                                    <rdf:rest>
                                                      <rdf:Description>
                                                        <rdf:type
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#List"/>

<rdf:first>v1.0.dev4</rdf:first>
                                                        <rdf:rest
rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                                                      </rdf:Description>
                                                    </rdf:rest>
                                                  </rdf:Description>
```

```
                                                    </rdf:rest>
                                                  </rdf:Description>
                                                </rdf:rest>
                                              </rdf:Description>
                                            </rdf:rest>
                                          </rdf:Description>
                                        </rdf:rest>
                                      </rdf:Description>
                                    </rdf:rest>
                                  </rdf:Description>
                                </rdf:rest>
                              </rdf:Description>
                            </rdf:rest>
                          </rdf:Description>
                        </rdf:rest>
                      </rdf:Description>
                    </rdf:rest>
                  </rdf:Description>
                </owl:oneOf>
              </rdfs:Datatype>
            </rdfs:range>
          </owl:DatatypeProperty>



          <!-- urn:absolute:cwlOntology#dirname -->

          <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#dirname">
            <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
            <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
          </owl:DatatypeProperty>



          <!-- urn:absolute:cwlOntology#doc -->

          <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#doc">
            <rdfs:subPropertyOf
        rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
            <rdfs:domain
        rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
            <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
            <rdfs:domain
        rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
```

```
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>



<!-- urn:absolute:cwlOntology#dockerFile -->

<owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#dockerFile">
   <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#dockerProps"/>
   <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>



<!-- urn:absolute:cwlOntology#dockerImageId -->

<owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#dockerImageId">
   <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#dockerProps"/>
   <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>



<!-- urn:absolute:cwlOntology#dockerImport -->

<owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#dockerImport">
   <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#dockerProps"/>
   <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>



<!-- urn:absolute:cwlOntology#dockerLoad -->

<owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#dockerLoad">
   <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#dockerProps"/>
   <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>



<!-- urn:absolute:cwlOntology#dockerOutputDirectory -->
```

```
  <owl:DatatypeProperty
rdf:about="urn:absolute:cwlOntology#dockerOutputDirectory">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#dockerProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#dockerProps -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#dockerProps">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#requirementProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#dockerPull -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#dockerPull">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#dockerProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#DockerRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#fileClass -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#fileClass">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#format -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#format">
    <rdfs:subPropertyOf
rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
```

```
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#glob -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#glob">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandOutputBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandOuputBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#hint -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#hint">
    <rdfs:subPropertyOf
rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#double"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range>
      <rdfs:Datatype>
        <owl:oneOf>
          <rdf:Description>
            <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#List"/>
            <rdf:first>Directory</rdf:first>
            <rdf:rest>
              <rdf:Description>
                <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#List"/>
                <rdf:first>File</rdf:first>
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
              </rdf:Description>
            </rdf:rest>
          </rdf:Description>
        </owl:oneOf>
      </rdfs:Datatype>
```

```
      </rdfs:range>
   </owl:DatatypeProperty>



   <!-- urn:absolute:cwlOntology#id -->

   <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#id">
      <rdfs:subPropertyOf
rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
      <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
      <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
      <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
   </owl:DatatypeProperty>
   <owl:Axiom>
      <owl:annotatedSource rdf:resource="urn:absolute:cwlOntology#id"/>
      <owl:annotatedProperty rdf:resource="http://www.w3.org/2000/01/rdf-
schema#domain"/>
      <owl:annotatedTarget
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
      <rdfs:comment>required</rdfs:comment>
   </owl:Axiom>
   <owl:Axiom>
      <owl:annotatedSource rdf:resource="urn:absolute:cwlOntology#id"/>
      <owl:annotatedProperty rdf:resource="http://www.w3.org/2000/01/rdf-
schema#domain"/>
      <owl:annotatedTarget
rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
      <rdfs:comment>non required</rdfs:comment>
   </owl:Axiom>
   <owl:Axiom>
      <owl:annotatedSource rdf:resource="urn:absolute:cwlOntology#id"/>
      <owl:annotatedProperty rdf:resource="http://www.w3.org/2000/01/rdf-
schema#domain"/>
      <owl:annotatedTarget
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
      <rdfs:comment>required</rdfs:comment>
   </owl:Axiom>



   <!-- urn:absolute:cwlOntology#itemSeparator -->
```

```
  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#itemSeparator">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#label -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#label">
    <rdfs:subPropertyOf
rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#loadContents -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#loadContents">
    <rdfs:subPropertyOf
rdf:resource="http://www.w3.org/2002/07/owl#topDataProperty"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandOuputBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#location -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#location">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#nameext -->
```

```
  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#nameext">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#nameroot -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#nameroot">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#outdirMax -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#outdirMax">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#outdirMin -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#outdirMin">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#outputEval -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#outputEval">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandOutputBinding"/>
```

```
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandOuputBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#path -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#path">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#position -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#position">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#positiveInteger"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#prefix -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#prefix">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#ramMax -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#ramMax">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
```

```
  <!-- urn:absolute:cwlOntology#ramMin -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#ramMin">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#requirementProps -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#requirementProps">
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#Requirements"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#resourceRequiredProps -->

  <owl:DatatypeProperty
rdf:about="urn:absolute:cwlOntology#resourceRequiredProps">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#requirementProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#secondaryFiles -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#secondaryFiles">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandParameter"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
```

```
  <!-- urn:absolute:cwlOntology#separate -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#separate">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#shellQuote -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#shellQuote">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#size -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#size">
    <rdfs:subPropertyOf rdf:resource="urn:absolute:cwlOntology#FileProperties"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#stderr -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#stderr">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineToolProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#stdin -->
```

```
  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#stdin">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineToolProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#stdout -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#stdout">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineToolProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#streamable -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#streamable">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandParameter"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#tmpdirMax -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#tmpdirMax">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
```

```
  <!-- urn:absolute:cwlOntology#tmpdirMin -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#tmpdirMin">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#resourceRequiredProps"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#ResourceRequirement"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#long"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#type -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#type">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandParameter"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
    <rdfs:domain
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!-- urn:absolute:cwlOntology#valueFrom -->

  <owl:DatatypeProperty rdf:about="urn:absolute:cwlOntology#valueFrom">
    <rdfs:subPropertyOf
rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:domain rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>



  <!--

///////////////////////////////////////////////////////////////////////////
//////////
  //
  // Classes
  //
```

```
////////////////////////////////////////////////////////////////////////
//////////
   -->



   <!-- urn:absolute:cwlOntology#CommandInputParameter -->

   <owl:Class rdf:about="urn:absolute:cwlOntology#CommandInputParameter">
      <rdfs:subClassOf rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
      <owl:disjointWith
rdf:resource="urn:absolute:cwlOntology#CommandLineBinding"/>
   </owl:Class>



   <!-- urn:absolute:cwlOntology#CommandLineBinding -->

   <owl:Class rdf:about="urn:absolute:cwlOntology#CommandLineBinding">
      <rdfs:subClassOf
rdf:resource="urn:absolute:cwlOntology#CommandInputParameter"/>
   </owl:Class>



   <!-- urn:absolute:cwlOntology#CommandLineTool -->

   <owl:Class rdf:about="urn:absolute:cwlOntology#CommandLineTool"/>



   <!-- urn:absolute:cwlOntology#CommandOuputBinding -->

   <owl:Class rdf:about="urn:absolute:cwlOntology#CommandOuputBinding">
      <rdfs:subClassOf
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
      <owl:disjointWith
rdf:resource="urn:absolute:cwlOntology#CommandOutputParameter"/>
   </owl:Class>



   <!-- urn:absolute:cwlOntology#CommandOutputParameter -->
```

```
  <owl:Class rdf:about="urn:absolute:cwlOntology#CommandOutputParameter">
    <rdfs:subClassOf rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
  </owl:Class>


  <!-- urn:absolute:cwlOntology#DockerRequirement -->

  <owl:Class rdf:about="urn:absolute:cwlOntology#DockerRequirement">
    <rdfs:subClassOf rdf:resource="urn:absolute:cwlOntology#Requirements"/>
  </owl:Class>


  <!-- urn:absolute:cwlOntology#File -->

  <owl:Class rdf:about="urn:absolute:cwlOntology#File"/>


  <!-- urn:absolute:cwlOntology#Outputs -->

  <owl:Class rdf:about="urn:absolute:cwlOntology#Outputs">
    <rdfs:subClassOf rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
  </owl:Class>


  <!-- urn:absolute:cwlOntology#Requirements -->

  <owl:Class rdf:about="urn:absolute:cwlOntology#Requirements">
    <rdfs:subClassOf rdf:resource="urn:absolute:cwlOntology#CommandLineTool"/>
  </owl:Class>


  <!-- urn:absolute:cwlOntology#ResourceRequirement -->

  <owl:Class rdf:about="urn:absolute:cwlOntology#ResourceRequirement">
    <rdfs:subClassOf rdf:resource="urn:absolute:cwlOntology#Requirements"/>
  </owl:Class>
</rdf:RDF>
```

<!-- Generated by the OWL API (version 4.2.8.20170104-2310)
https://github.com/owlcs/owlapi -->