INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

HYBRID ONTOLOGY MAPPING INTERFACE

H.O.M.I.

Projeto Final de Curso

Licenciatura em Engenharia Informática e Computadores

Ana Carolina Baptista

41487@alunos.isel.ipl.pt

960314580

Eliane Almeida

41467@alunos.isel.ipl.pt

960271968

Relatório de Progresso

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

Abril de 2018

Índice

Lista de Figuras

Lista de Tabelas

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. Apesar da popularidade das ontologias, há em geral dificuldade em transformar o conhecimento pré-definido num caso concreto.

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 [2] (*Web Ontology Language*), em várias situações poderá não ser uma tarefa simples para os bioinformáticos representar o seu conhecimento do domínio através das ontologias.

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* [3]. Existem também algumas bibliotecas para Java que fazem o mapeamento de XML para OWL, que podem ser utilizadas por *developers*. Contudo, não temos conhecimento da existência de uma ferramenta que combine estes dois aspetos: a criação de novas instancias através de uma ontologia bem como o mapeamento de um caso concreto escrito noutra linguagem numa instancia de uma ontologia.

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

2 Descrição do problema

Uma ontologia, na área da ciência da computação, é um modelo de dados que representa um grupo de ideias ou conceitos dentro de um determinado domínio e as relações que existem entre eles. Estas (ontologias) 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 indivíduos (objetos básicos), classes (conjuntos, coleções ou tipo de objetos), atributos (propriedades, características ou parâmetros) e relacionamentos (entre objetos). [4]

OWL (Ontology Web Language) é uma linguagem utilizada para definir e instânciar ontologias em Web. Em OWL conseguimos definir as 4 propriedades de ontologias descritas acima (individuas, classes, atributos e relacionamentos). Esta linguagem foi desenhada para processar conteúdo (informação), facilitando assim sua interpretação por maáquinas. Quando comparado com outras linguagens (XML, RDF, etc.), OWL destaca-se por fornecer um vocabulário adicional com uma semântica formal. É a linguagem recomendada da W3C (World Wide Web Consortium).

Na figura 1 podemos a representação em grafo de um exemplo de uma ontologia, composta por indivíduos ("Faculdade", "Departamento", "Curso") bem como as relações entre eles, como por exemplo, "Faculdade" tem uma relação com "Departamentos" denominada por "É composto por". Na figura 2 podemos observar uma instância da ontologia descrita pela figura 1.

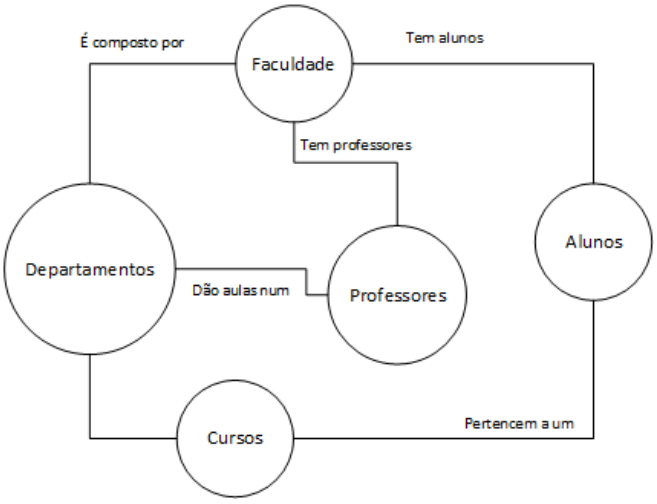

Figura 2.1- Exemplo de uma ontologia

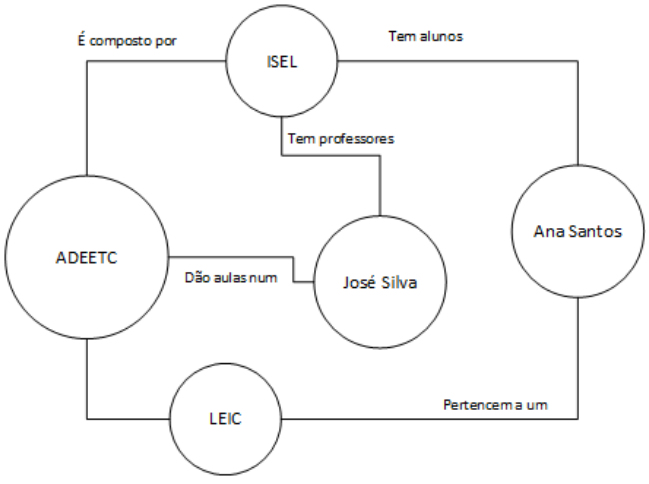

Figura 2.2- Exemplo de uma instancia da ontologia da figura

2.1 Ferramentas já existentes

Com o crescimento da popularidade de OWL para a descrição de dados, naturalmente apareceram também alguns softwares para a edição e manipulação de OWL.

Neste capítulo iremos falar sobre alguns destes programas e sobre as diferenças entre eles. Por fim iremos explicar como o nosso projeto de insere nesta área bem no que o difere destes programas que já existem. [5] [6]

Iremos comparar 3 programas populares: Apollo, Protege e Swoop.

2.1.1 Apollo

Apollo é um modelador de fácil utilização para o utilizador, de utilização gratuita e local. Nesta aplicação, um utilizador pode modelar ontologias com noções básicas de ontologias (classes, relações, instâncias, etc). Também é possível criar novas instancias a partir das classes presentes nessas ontologias. Alguns dos pontos fortes deste software é o seu corretor de tipos que mantem 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 interface gráfica intuitiva, ao contrario dos seus concorrentes. Apollo carece também de extração de informação em web e da possibilidade de dar uma experiencia multiutilizador aos seus utilizadores, para trabalhos em colaboração com mais do que um utilizador.

2.1.2 Protégé

Protege é um editor e modelador de ontologias, de utilização gratuita e tem uma vertente local bem como online (WebProtégé). 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.

2.1.3 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 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

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

JXML2OWL[2]. Estas bibliotecas foram desenvolvidas em Java e estão disponíveis para *developers* usarem também em Java.

Contudo, apesar da nossa pesquisa, não encontramos nenhuma aplicação que oferecesse todos os pontos fortes das aplicações descritas acima, tais como: versão online como local, mapear XML para OWL, etc.

A nossa ferramenta visa juntar os pontos fortes destas aplicações numa só aplicação. Na nossa ferramenta (H.O.M.I.) iremos ter uma vertente local assim como uma vertente online. Ambas terão uma visualização simples e fácil para os nossos utilizadores bem como a opção de apenas criar uma nova instância a partir de uma dada ontologia ou então, através de um exemplo concreto descrito em XML, realizar o mapeamento de conceitos presentes na ontologia em questão.

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 contem interface	Não	Sim
HOMI	Sim	Sim	Sim	Sim	Sim	Sim

Tabela 2-1 - Comparação entre várias ferramentas e a nossa aplicação

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

3 ChaosPop

ChaosPop é uma API que foi desenvolvida para permitir mapear ontologias dado um caso concreto da mesma.

Para que um mapeamento seja realizado são indispensáveis dois ficheiros: um ficheiro do tipo OWL referente a uma ontologia (*OntologyFile*) e outro que representa um caso concreto de uma instancia que queremos mapear (*DataFile*). Neste momento, os ficheiros que são suportáveis como *DataFile* são de extensões JSON e XML.

Quando são submetidos os ficheiros *DataFile* e *OntologyFile*, a informação presente nestes são guardadas numa base de dados remota. Quando pretende-se realizar o mapeamento de dados, é necessário buscar os ficheiros que tenciona mapear, sejam eles um ou mais *DataFile*'s e *OntologyFile*'s.

No que diz respeito ao *DataFile* os dados são guardados em *nodes,* formando assim uma árvore. Por exemplo, se for submetido o ficheiro XML da Figura 3.0.1, a árvore abstraída a partir dos *nodes* deste será a apresentada na Figura 3.0.2.

```
<family>
    <member>
        <name>
            <given>João </given>
            <surname>Silva</surname>
        </name>
        <sex>male</sex>
        <birth_year>1995</birth_year>
        <ancestors>
            <father>Zé Silva</father>
            <mother>Maria Aparecida Silva</mother>
        </ancestors>
        <siblings>
            <sister>Ana Silva</sister>
        </siblings>
    </member>
</family>
```

Figura 3.0.1 - Exemplo de ficheiro de dados

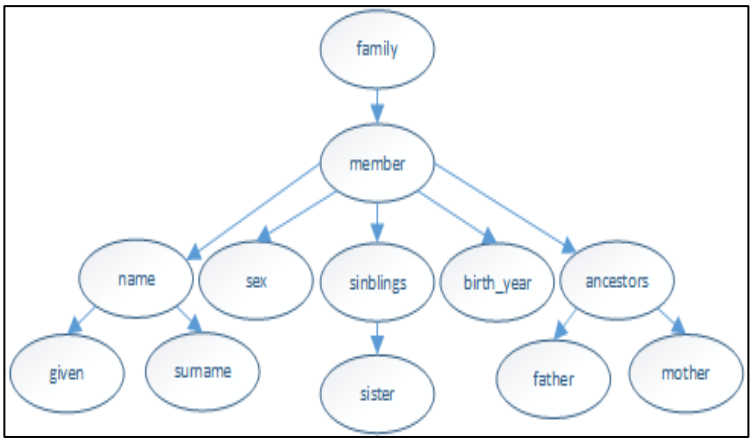

Figura 3.0.2 - Exemplo de árvore gerada

Por outro lado, quando é submetido um *OntologyFile,* a informação que contém neste é mantida em base de dados da seguinte forma:

- Como mencionado no capitulo 2, uma ontologia pode ter *classes, data properties* e *object properties*, sendo assim é possível obter estes dados referentes a uma ontologia com o identificador da mesma;
- Uma *class* pode ter *data properties* e *object properties,* logo é possível aceder a esta informação com base no identificador da ontologia na qual a classe pertence e no identificador da classe em questão.

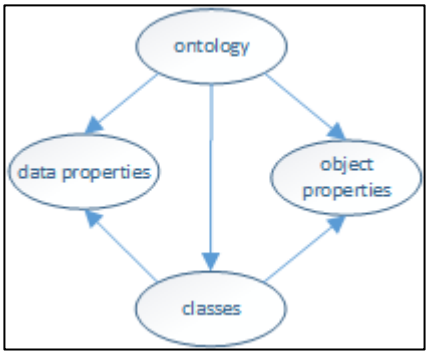

Figura 3.0.3 - Grafo de uma ontologia

Quando se realizar o mapeamento de uma ontologia, está ao critério do utilizador decidir quais dados desta ele pretende obter, assim como a ordem com que deseja mapear estes.

Para submeter ficheiros, obter dados referentes aos ficheiros submetidos e mapear os conceitos existentes é necessário ter conhecimento dos *endpoints* existentes no ChaosPop que satisfazem estas necessidades. Estes *endpoints* pertencem ao *services* e estão divididos e implementados em determinadas classes de acordo com a sua funcionalidade. O diagrama UML apresentado abaixo mostra as funções relativas a cada *endpoint.*

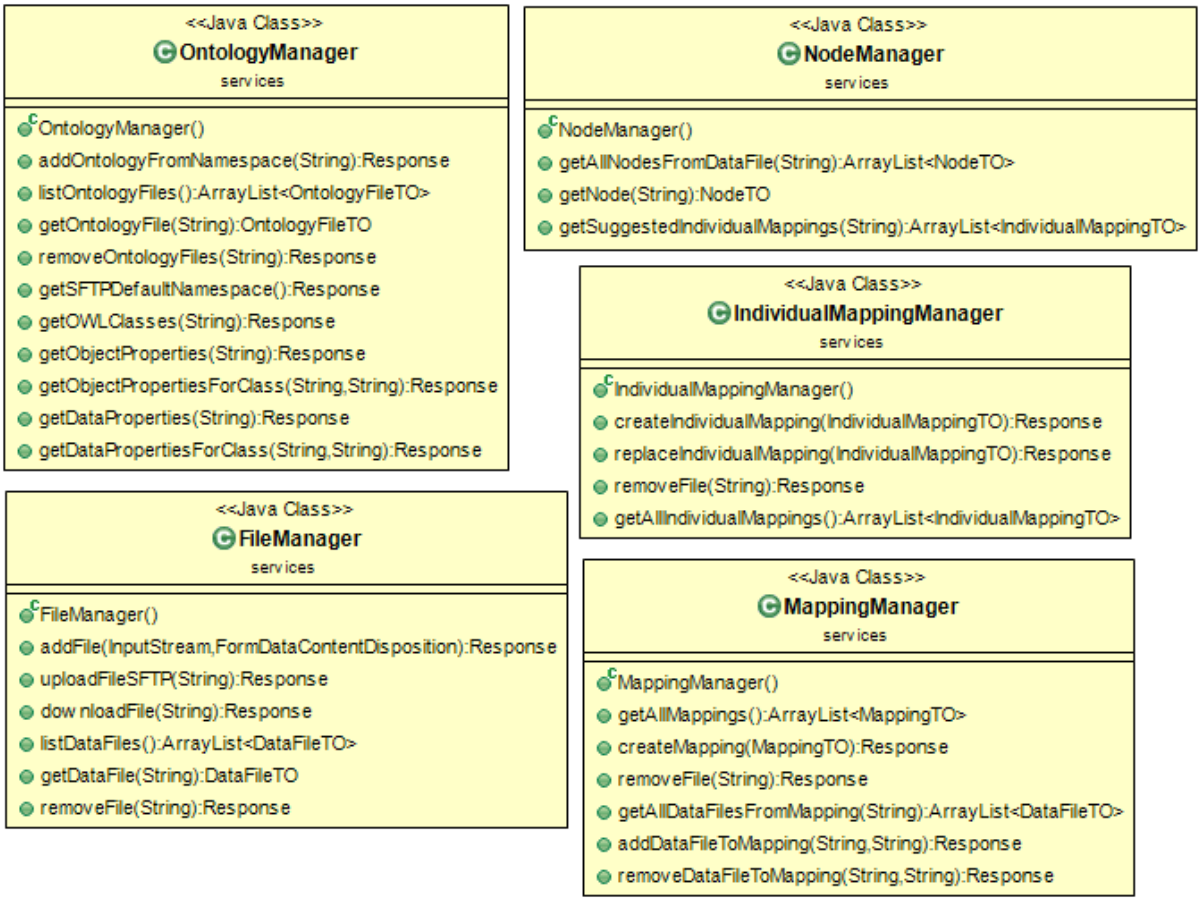

Figura 3.0.4 - Diagrama UML dos services do Chaos Pop

4 Arquitetura

Neste capítulo descreveremos detalhadamente cada módulo da aplicação, assim como a forma como estes interagem entre si e as tecnologias utilizadas na implementação de cada um.

4.1 Descrição

A Figura 4.1 mostra a arquitetura da aplicação, na qual está representada a interação entre os distintos módulos existentes. Esta interação inicia-se quando o utilizador (*User*) insere um ficheiro com a definição de uma ontologia (*Ontology File*) e, opcionalmente, um segundo ficheiro (*Data File*). Estes ficheiros irão ser submetidos a API ChaosPop. De seguida irá ser gerada uma interface gráfica onde o usuário poderá anotar valores aos vários conceitos presentes no *Ontology File* ou anotar os conceitos do *Data File* com os termos do *Ontology File*. No final deste processo, é gerado um novo ficheiro OWL que contém os dados descritos de acordo com *Ontology File*. Iremos também dar a opção ao usuário de guardar os ficheiros de input e output numa base de dados remota.

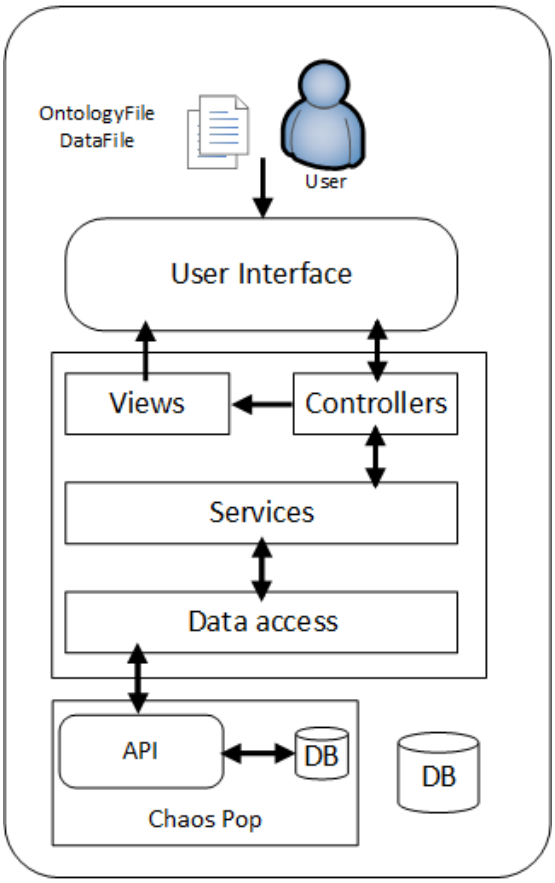

Figura 4.1 – Arquitetura da aplicação

4.2 Hybrid Ontology Mapping Interface (H.O.M.I)

A aplicação foi implementada por níveis de acesso, na qual cada nível apenas comunica com o superior ou inferior, nunca saltando um nível durante a comunicação. Esta implementação proporciona uma maior facilidade na manipulação das funcionalidades.

Os níveis existentes são:

- ***controllers:*** contém todos os *endpoints* possíveis de serem acedidos através da *user interface* e comunica com *views* e *services*;
- ***services:*** nível intermédio 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.*

Em cada um dos níveis existe diferentes módulos, nestes estão implementadas funções usufruindo de algumas tecnologias das quais temos conhecimento.

4.2.1 Tecnologias

Node.js

O que é? Um *runtime* de JavaScript, que pelo facto de processar o código JavaScript desvinculando-o do browser, possibilita o desenvolvimento de aplicações estáveis e rápidas.

Por que escolhemos? Escolhemos esta tecnologia porque já tivemos experiências noutras unidades curriculares e concluímos que fornece uma maneira fácil de construir uma aplicação.

Onde utilizamos? Todo o projeto é realizado utilizando esta tecnologia.

D3.js

O que é? Uma ferramenta para JavaScript que associa os dados ao *Document Object Model(DOM)* e permite manipulá-los, gerando gráficos usando diretamente padrões web como o HTML e o CSS.

Por que escolhemos? Optamos por tal pelo facto de suportar comportamentos dinâmicos de interação e animação e grandes conjuntos de dados. Outra característica que nos chamou a atenção foi com facilidade obter gráficos visualmente apelativos.

Onde utilizamos? Utilizamos na apresentação dos dados referentes aos ficheiros de entrada, de modo a obter uma representação intuitiva e de fácil entendimento destes pelo utilizador.

Electron

O que é? Uma framework utilizada para criar aplicações multi-platforma desktop com tecnologias web (HTML, JavaScript e CSS).

Por que escolhemos? Como a nossa aplicação será desenvolvida em JavaScript e esta é a suportada pelo Electron, decidimos utilizá-la para assim não ter a necessidade de aprender a usar uma nova tecnologia no desenvolvimento do sistema desktop.

Onde utilizamos? Na criação da aplicação desktop iremos aplicar esta tecnologia.

Express

O que é? Um framework back-end em Node.js que cria rotas, middlewares, entre outras para facilitar a criação de API's. Este cria e obtém dados a partir do servidor, independentemente da linguagem que irá utilizá-los.

Por que escolhemos? Por já termos experiências com esta tecnologia em outras alturas e a sua utilização ser fácil.

Onde utilizamos? Toda as rotas disponíveis na camada *controllers* estão implementadas com base neste.

4.2.2 Base de dados

O armazenamento dos dados relativos aos ficheiros *OntologyFile* e *DataFile* está neste momento a ser realizado na base de dados do ChaosPop. Posteriormente, tencionamos alterar isto de modo a que tenhamos a nossa própria base de dados.

5 Progresso do projeto

A calendarização definida anteriormente na proposta foi criada baseada numa arquitetura por módulos. Após uma análise concluímos que a aplicação seria mais bem-adaptada numa implementação por níveis de acesso, o que por consequência causou alterações na arquitetura e calendarização inicial.

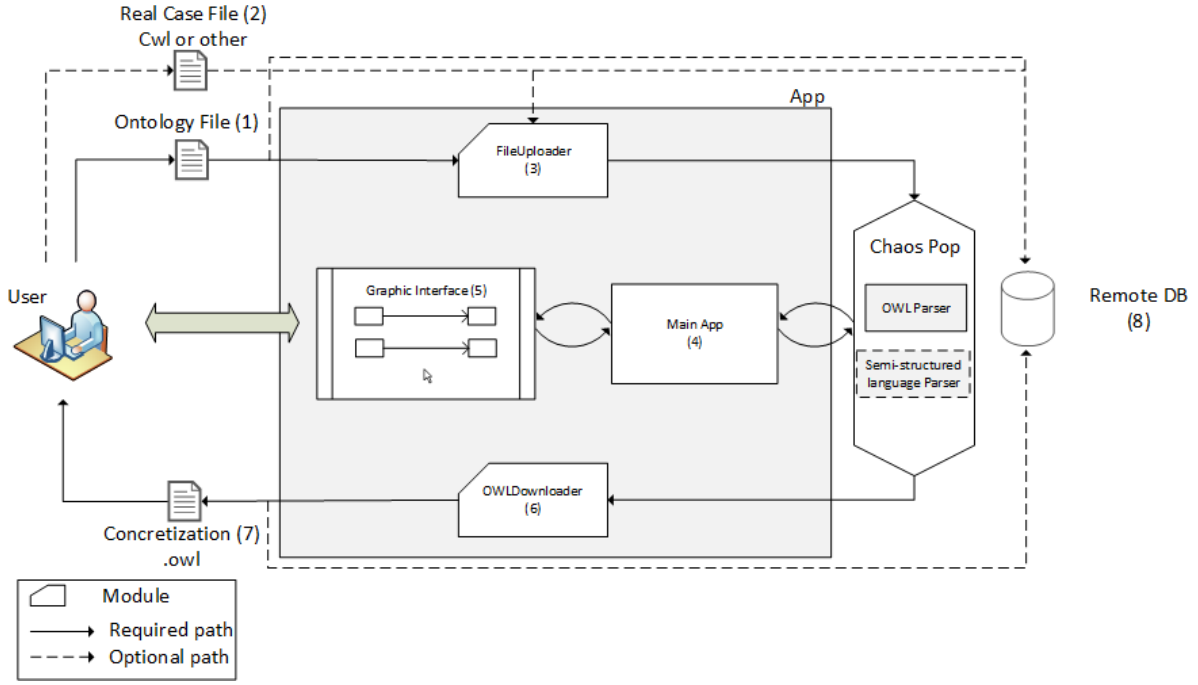

Figura 5.1 – Arquitetura inicial da aplicação

Os módulos *FileUploader, OWLDownloader e MainApp* estão todos implementados, por níveis, em *controllers, services* e *data-access.* Por exemplo, quando é feito um *upload* de um ficheiro é realizado um *request* que recebe este na *route* que foi criada com esta funcionalidade e está disponível em *controllers.* Posteriormente, este é enviado por intermédio de uma função implementada nos *services* para o *data-access*, sendo assim submetido ao ChaosPop.

Já o *Graphic Interface* corresponde à *User Interface* da atual arquitetura (Figura 4.1).

Data de início	Semana	Descrição
19/02/2018	1-2	- Compreensão da necessidade da ferramenta nos dias atuais - Estudo do Chaos Pop
05/03/2018	3-4	- Estudo da ferramenta Electron - Desenvolvimento da proposta
19/03/2018	5	- Entrega da proposta do projeto - Utilização do Chaos Pop em alguns exemplos
26/03/2018	6-7	- Desenvolvimento do módulo *FileUploader* e *OWLDownloader*
09/04/2018	8-10	- Realização do esqueleto da *Main App* - Desenvolvimento da *Graphic Interface*
30/04/2018	11	- Apresentação individual e entrega do relatório de progresso
07/05/2018	12-13	- Definição da descrição de ferramentas em OWL
21/05/2018	14	- Desenvolvimento da aplicação *desktop* - Criação do cartaz
28/05/2018	15 - 17	- Entrega do cartaz e da versão beta - Otimização dos módulos
18/06/2018	18	- Testes de escalabilidade
25/06/2018	19-21	- Finalização do relatório e entrega da versão final

Tabela 5-1 - Calendarização inicial do projeto

As alterações referentes a calendarização não foram apenas devido ao facto de mudarmos a arquitetura, mas também na divisão das semanas para cada tarefa especifica. Isto ocorreu porque gastamos mais semanas do que definido inicialmente para ter acesso a API do Chaos Pop, estudá-la e usá-la em alguns exemplos. Neste momento estamos a mapear os conceitos das ontologias e a utilizar a tecnologia D3.js (4.2.1 Tecnologias). Sendo assim, nos encontramos 2 semanas atrasadas para concluir a implementação da *User Interface* e dos níveis de acesso.

Na Tabela 5-2 está apresentada a nova calendarização do projeto com as mudanças necessárias.

Data de início	Semana	Descrição
19/02/2018	1-2	- Compreensão da necessidade da ferramenta nos dias atuais - Estudo do Chaos Pop
05/03/2018	3-4	- Estudo da ferramenta Electron - Desenvolvimento da proposta
19/03/2018	5-7	- Entrega da proposta do projeto - Esclarecimentos sobre a API Chaos Pop e utilização do mesmo em alguns exemplos
09/04/2018	8-10	- Início da implementação dos níveis de acesso (*controllers, services, data-access*) - Início do desenvolvimento da *User Interface*
30/04/2018	11	- Apresentação individual e entrega do relatório de progresso
07/05/2018	12	- Continuação do desenvolvimento da *User Interface* e dos níveis de acesso
14/05/2018	13	- Definição da descrição de ferramentas em OWL
21/05/2018	14	- Desenvolvimento da aplicação *desktop* - Criação do cartaz
28/05/2018	15 - 17	- Entrega do cartaz e da versão beta - Otimização dos módulos
18/06/2018	18	- Testes de escalabilidade
25/06/2018	19-21	- Finalização do relatório e entrega da versão final

Tabela 5-2 - Calendarização atual do projeto

6 Referências

[1] “W3C,” [Online]. Available: https://www.w3.org/standards/semanticweb/ontology. [Acedido em 19 03 2018].

[2] “W3C,” [Online]. Available: https://www.w3.org/OWL/. [Acedido em 09 03 2018].

[3] “Protege,” [Online]. Available: https://protege.stanford.edu/. [Acedido em 15 03 2018].

[4] “Ontologias,” 22 04 2018. [Online]. Available: https://pt.wikipedia.org/wiki/Ontologia_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o).

[5] “Editores 1,” 21 04 2018. [Online]. Available: https://www.w3.org/wiki/Ontology_editors.

[6] “Artigo sobre editores de ontologias,” 21 04 2018. [Online]. Available: http://www.ef.uns.ac.rs/mis/archive-pdf/2013%20-%20No2/MIS2013-2-4.pdf .

[7] “Editores 2,” 21 04 2018. [Online]. Available: https://www.w3.org/wiki/SemanticWebTools#Semantic_Web_Development_Tools:_Introduction.

7 Bibliografia

Jamie Taylor, Colin Evans, Toby Segaran. (2009). Programming the Semantic Web.

Jim R. Wilson. (2013). Node.js the Right Way: Practical, Server-side JavaScript that Scales.