 **INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**HYBRID ONTOLOGY MAPPING INTERFACE**

**Projeto Final de Curso**

Licenciatura em Engenharia Informática e Computadores

Ana Carolina Baptista

[41487@alunos.isel.ipl.pt](mailto:41487@alunos.isel.ipl.pt)

960314580

Eliane Almeida [41467@alunos.isel.ipl.pt](mailto:41467@alunos.isel.ipl.pt)

960271968

**Relatório de Progresso**

Orientadores:

Cátia Vaz, ISEL, [cvaz@cc.isel.ipl.pt](mailto:cvaz@cc.isel.ipl.pt)

José Simão, ISEL, [jsimao@cc.isel.ipl.pt](mailto:jsimao@cc.isel.ipl.pt)

Alexandre P. Francisco, IST, [aplf@ist.utl.pt](mailto:aplf@ist.utl.pt)

23 de Abril de 2018

Índice

[Lista de Figuras 3](#_Toc512113715)

[Lista de Tabelas 4](#_Toc512113716)

[1 Introdução 5](#_Toc512113717)

[2 Descrição do problema 6](#_Toc512113718)

[3 Chaos Pop 7](#_Toc512113719)

[4 Arquitetura 10](#_Toc512113720)

[4.1 Descrição 10](#_Toc512113721)

[4.2 Hybrid Ontology Mapping Interface (H.O.M.I) 11](#_Toc512113722)

[4.2.1 Tecnologias 11](#_Toc512113723)

[4.2.2 Base de dados 12](#_Toc512113724)

[5 Progresso do projeto 13](#_Toc512113725)

# Lista de Figuras

[Figura 3.0.1 - Exemplo de ficheiro de dados 7](#_Toc512007291)

[Figura 3.0.2 - Exemplo de árvore gerada 8](#_Toc512007292)

[Figura 3.0.3 - Grafo de uma ontologia 8](#_Toc512007293)

[Figura 3.0.4 - Diagrama UML dos services do Chaos Pop 9](#_Toc512007294)

[Figura 4.1 – Arquitetura da aplicação 10](#_Toc512007295)

# Lista de Tabelas

# Introdução

# Descrição do problema

# 3 Chaos Pop

Chaos Pop 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 OWL referente à uma ontologia (*OntologyFile*) e outro que representa um caso concreto da ontologia em questão (*DataFile*). Neste momento, os ficheiros que são suportáveis como *DataFile* são de extensões JSON e XML.

Aquando da submissão de 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.

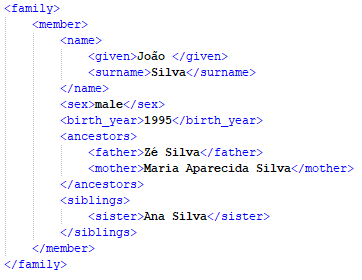


Figura 3.0.1 - Exemplo de ficheiro de dados

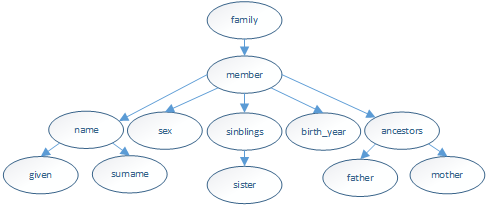


Figura 3.0.2 - Exemplo de árvore gerada

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

* uma *ontology* 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.

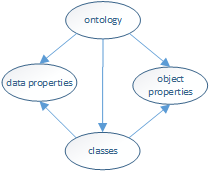


Figura 3.0.3 - Grafo de uma ontologia

Quando for 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 Chaos Pop que satisfazem estas necessidades. Estes *endpoints* são pertencentes ao *services* eestão divididos e implementados em determinadas classes de acordo com a sua funcionalidade. O diagrama UML apresentado abaixo mostra as funções relativas a cada *endpoint.*

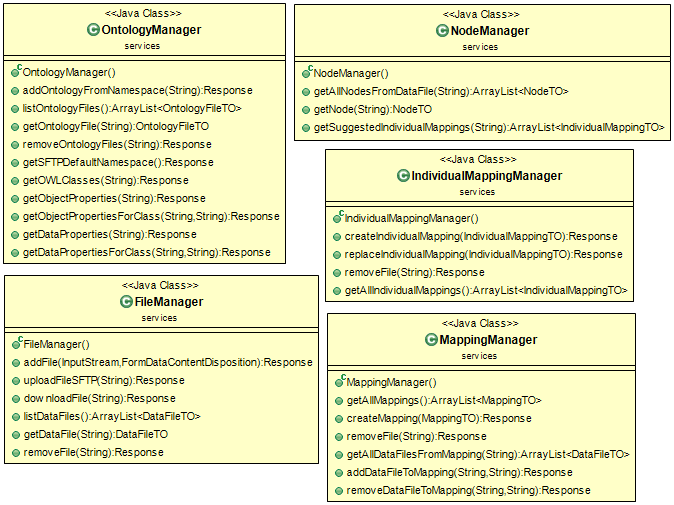


Figura 3.0.4 - Diagrama UML dos services do Chaos Pop

# 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 Chaos Pop. 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.

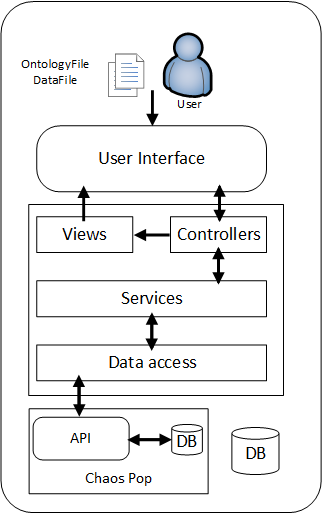


Figura 4.1 – Arquitetura da aplicação

## **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 em outras 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 manipular estes 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 bonitos visualmente.

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

**Electron**

**O que é?** Um framework utilizado para criar aplicações multiplatformas desktop com tecnologias web (HTML, JavaScript e CSS).

**Por que escolhemos?** Como a nossa aplicação será desenvolvida na tecnologia JavaScript e esta é a suportada pelo Electron, decidimos utilizá-lo 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 sendo realizado na base de dados do ChaosPop. Posteriormente, tencionamos alterar isto de modo a que tenhamos a nossa própria base de dados.

# Progresso do projeto