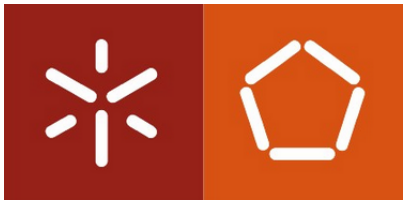

Universidade do Minho
Mestrado em Engenharia Informática

Interoperabilidade Semântica
2019/2020

Implementação de um sistema com Interoperabilidade Semântica

Autor:

Gabriel Santos PG41076

12 de Junho de 2020

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

Cada vez mais se verifica a existência de *software* informático feito à "medida" de cada entidade. É natural que tais sistemas, por serem feitos consoante a necessidade de cada um, não sigam determinados *standards*. Nesses casos é necessário garantir a existência de Interoperabilidade Semântica em sistemas onde a comunicação com outros sistemas, é uma prática frequente.

Interoperabilidade semântica trata-se da capacidade que um sistema recetor possui de receber e interpretar informações recebidas da mesma maneira que o sistema emissor as interpreta.

A plataforma ORCID [2], tem como objetivo compilar os trabalhos e estudos de investigadores científicos das áreas mais variadas. Este sistema fornece a cada investigador um identificador único, e é da responsabilidade da cada investigador manter o seu portolio atualizado. Além disso, este sistema disponibiliza uma API (*Application programming interface*) pública onde é possível ser feita a extração de diversas informações presentes na plataforma.

O trabalho demonstrado no seguinte relatório passa pela criação de um programa que permite extrair informações acerca de um determinado autor recorrendo à *API* da plataforma ORCID. Esta informação consiste em informação do autor em questão (biografia, nome) e das suas obras (título, *publisher*, WOS, EID, etc). A segunda parte do trabalho passa pela implementação de interoperabilidade semântica intra-plataformas, utilizando informação extraída da plataforma ORCID são recorridas a outras *API's* para ser obtida mais informação que não está disponível na plataforma ORCID, como por exemplo, é utilizada a API da plataforma SCOPUS [1] para ser obtida informação mais completa das publicações com *EID*.

A informação referente a cada autor é então transformada para ser armazenada numa base de dados relacional. É feito o armazenamento da mesma de modo a ser possível obtê-la *offline* e já organizada. Para estruturar e organizar todo o processo foi produzida uma API cujo objetivo é disponibilizar a informação da base de dados de uma forma manipulável. Finalmente foi desenvolvida uma *interface web* que consome os *end points* da API e faz a apresentação da informação presente na base de dados.

A informação apresentada não deve conter discrepâncias em relação à informação visível nas plataformas de onde foi retirada, provando assim a existência de uma correta interoperabilidade semântica entre o trabalho produzido e as plataformas onde se extraiu a informação.

2 Design

Nesta secção será apresentado o *design* da aplicação na granularidade arquitectural e detalhada com vista a descrever as decisões tomadas.

Devido à simplicidade do sistema apenas foram utilizadas duas vistas:

- **Vista de desenvolvimento** - Representa os vários módulos/componentes do sistema;
- **Vista de Processos** - Demonstra toda a comunicação entre processos, dentro do sistema, explicando assim a forma como os vários componentes interagem entre si.

2.1 Vista de Desenvolvimento

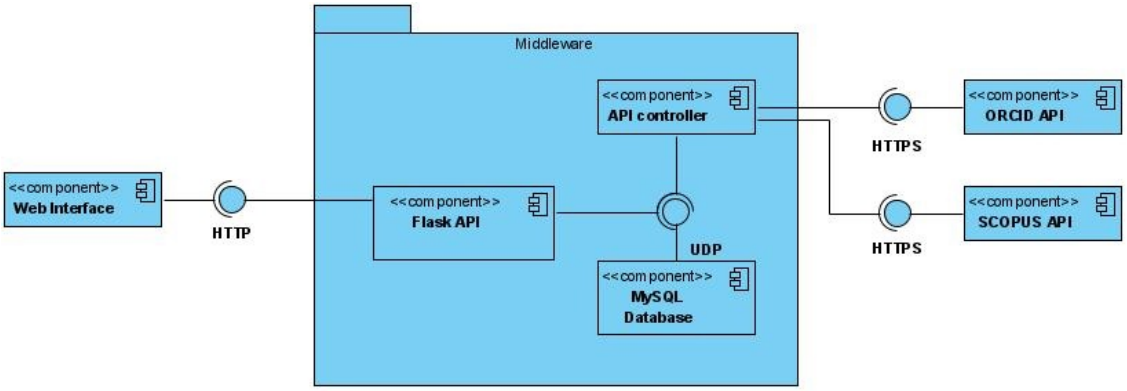

Figura 1: Diagrama de componentes

A figura 1 mostra os componentes desenvolvidos do sistema bem como as ligações entre si. Três dos componentes produzidos, *API controller*, MySQL *Database* e Flask *API*, constituem o *middleware* entre as *API*'s externas e o quarto componente, *web interface*.

- **API controller** - Contém funções e métodos responsáveis por extrair os dados das diferentes *API*'s externas, fazer o tratamento dos mesmos e inseri-los na base de dados.
- **MySQL *Database*** - Base de dados relacional onde é feito o armazenamento dos dados.

- **Flask API** - *API* desenvolvida em Flask que disponibiliza a informação presente na base de dados em diferentes *endpoints*.
- ***Web interface*** - Apresenta a informação contida na base de dados, esta pode ser especificada consoante o critério do utilizador.

Todos estes componentes serão explicados de uma forma mais minuciosa na secção 4.

2.2 Vista de Processos

As figuras 2 e 3 demonstram, de forma sequencial, o fluxo dos processos de uma inserção de um autor na base de dados e os diferentes processos de visualização de informação na *web interface*. Foram elaborados dois diagramas pois os processos presentes em cada um não se interligam.

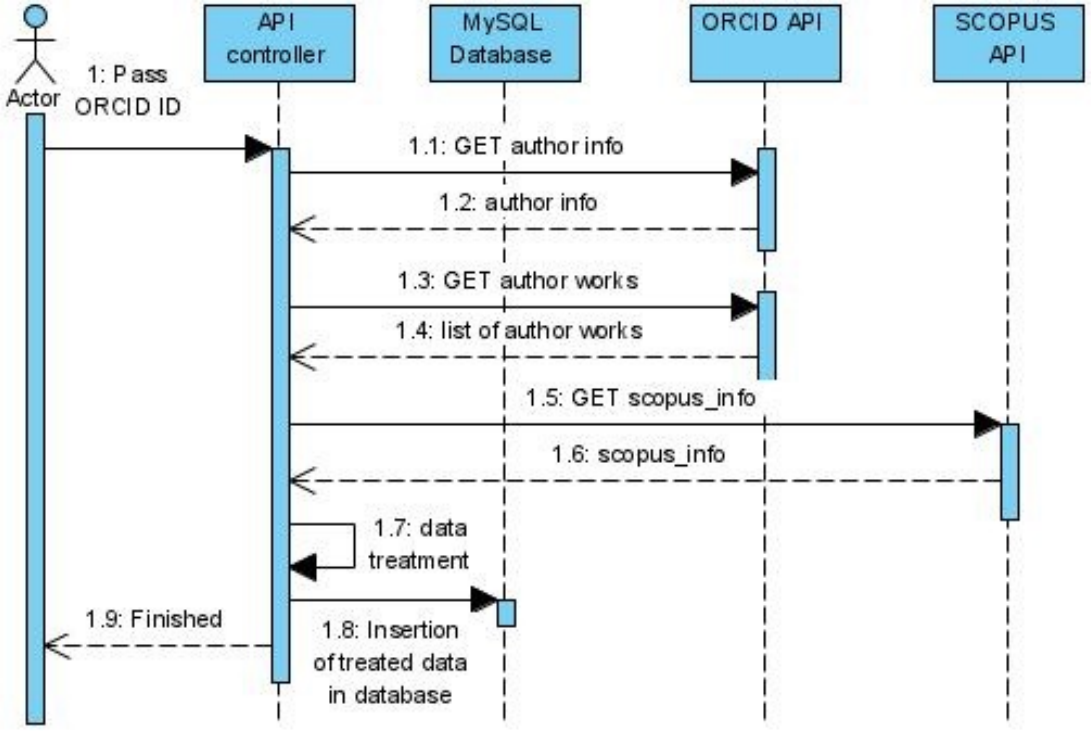

Figura 2: Diagrama de sequências (Inserção autor e publicações)

De modo a que se possam executar os processos da figura 3 é necessário executar o processo da figura 2. Este processo, conjunto de sequências, consiste na inserção de um autor e suas publicações na base de dados.

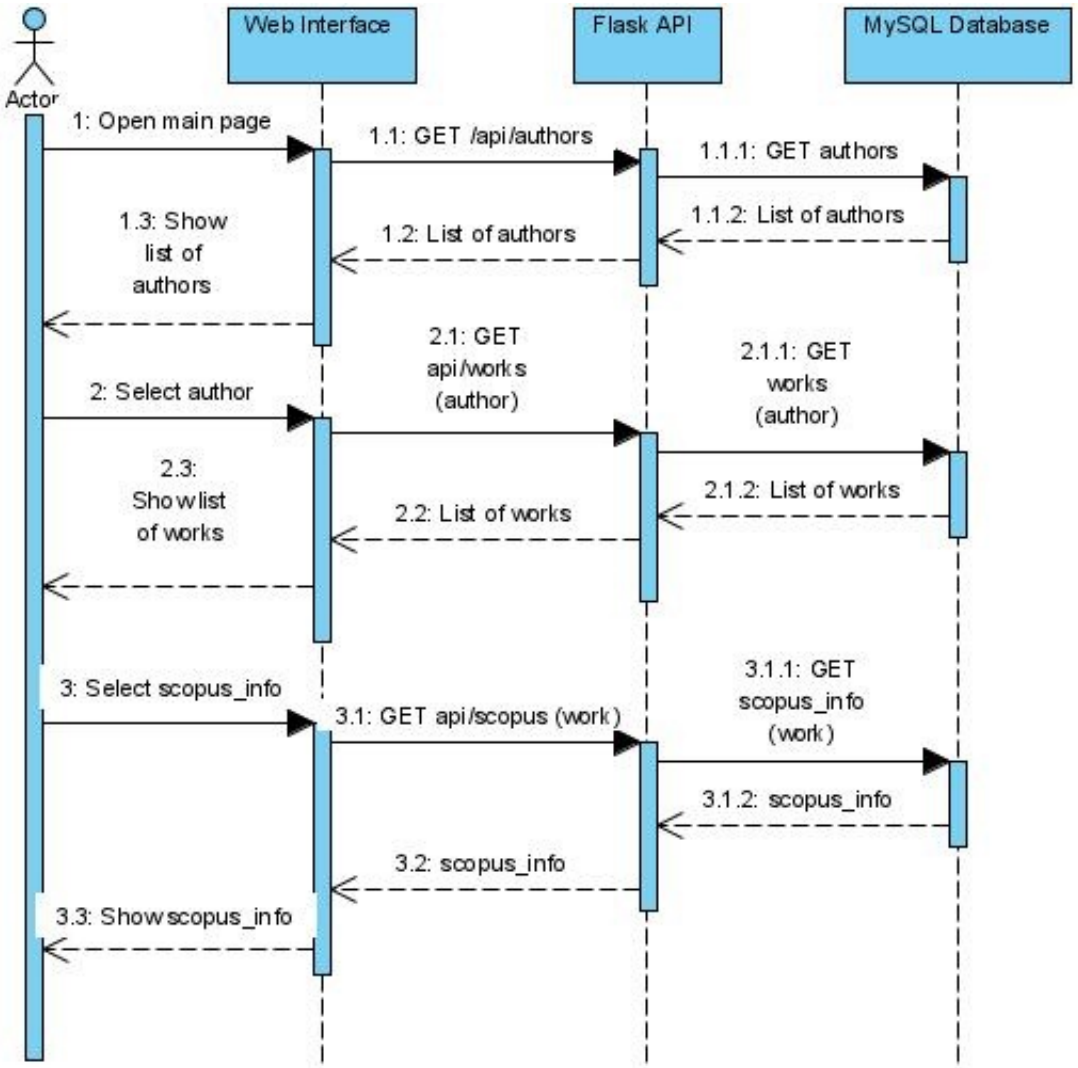

Figura 3: Diagrama de sequências (Visualização de dados)

Este diagrama contém as sequências de visualização de dados da *web interface*, consistindo o processo 1 na visualização dos autores disponíveis, 2 na visualização das publicações de um determinado autor e 3 na visualização da informação SCOPUS de uma determinada publicação (com EID).

3 *API*'s consumidas

A seguinte secção apresenta informação geral relativa às *API*'s consumidas bem como os *endpoints* e razão pela qual foram consumidos. Todos os *requests* foram feitos com *Application-Type* a *application/json* para que todas respostas obtidas fossem no formato *JSON*.

Foram também adicionados alguns filtros (*query string*) aos *endpoints* para ser obtida informação extra que não é obtida por defeito.

3.1 ORCID *API*

Esta é a primeira e principal *API* a ser utilizada no projeto. É com esta que é obtida informação referente ao autor e às suas publicações.

Esta *API* tem duas variantes, uma pública e uma privada (requer *api key*). Como foi possível obter toda a informação requerida com a *API* pública não foi necessário gerar uma *api key*.

Os *endpoints* consumidos desta *API* requerem o identificador único (ORCID iD) de um utilizador, este identificador segue o formato "xxxx-xxxx-xxxx-xxxx" correspondendo cada 'x' a um número entre 0 e 9.

O primeiro *endpoint* utilizado permite obter informação relativa ao utilizador associado ao identificador único fornecido:

https://pub.orcid.org/v3.0/**ORCID iD**/person

Da mesma forma foram obtidas as publicações de um autor, utilizando o seu ORCID iD e o seguinte *endpoint*:

https://pub.orcid.org/v3.0/**ORCID iD**/works

Foi ainda utilizado o *endpoint* abaixo apresentado para obter informação específica de uma publicação utilizando um ORCID iD e o putCode da publicação:

https://pub.orcid.org/v3.0/**ORCID iD**/works/**putCode**

3.2 SCOPUS *API*

A segunda API diz respeito ao repositório SCOPUS, no sistema em questão nesta apenas podem ser feitos *requests* às obras com *EID* (Scopus ID). Esta é utilizada de modo a obter informação complementar à anterior.

Esta API já requer uma *api key* que pode ser passada na *query string* ou *header*, tem limite de *requests* mensal e apenas podem ser feitas *requests* de determinadas redes (redes académicas, etc).

O *endpoint* consumido para obter informação de uma publicação específica requer o seu *EID* e corresponde ao seguinte *url*:

https://api.elsevier.com/content/abstract/scopus_id/**EID**

4 Componentes

Esta secção apresentará todos os componentes desenvolvidos, sua função e razão pela qual foram implementados.

4.1 *API controller*

Este *script*, desenvolvido em *python*, pode ser considerado a base de todo o sistema, é nele que são feitos pedidos às *API*'s (Secção 3), processada a informação obtida e feita a inserção da mesma na base de dados relacional.

Será explicado o funcionamento e os processos executados no *script* com um exemplo, caso o utilizador queria inserir o autor de ORCID iD "3213-1234-1321-5131", terá de executar o *script get_articles.py* com a *flag* "a" e passar o ORCID iD. A imagem 4 mostra a ordem pela qual é realizado este processo.

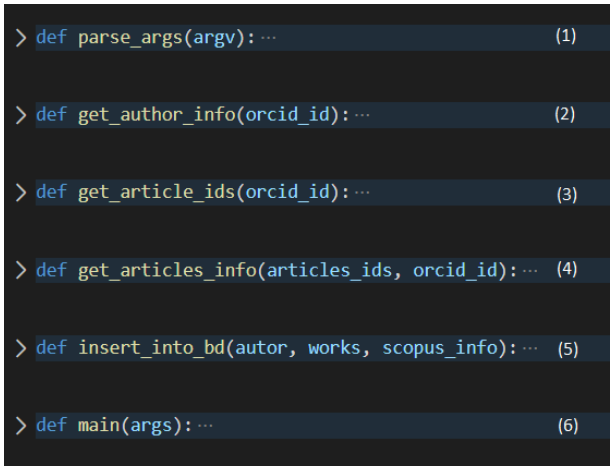

Figura 4: Funções *API controller*

A função 1 (parse_args) faz o *handling* e verificações do ORCID iD recebido, caso este não vá de acordo com as regras definidas é rejeitado.

As funções 2 e 3 (get_author_info e get_articles_ids) consomem os *endpoints* da ORCID *API* (subsecção 3.1), obtendo a informação do utilizador e das suas publicações.

A função 4 (get_articles_info) consome o *endpoint* da SCOPUS *API* (Subsecção 3.2) de modo a obter informação do repositório SCOPUS das publicações com EID.

Finalmente, a função 5 (insert_into_bd) transforma e organiza a informação para ser inserida na base de dados e faz a inserção da mesma. É nesta que também são feitas verificações de informações já presentes na base de dados.

4.2 MySQL *Database*

4.2.1 Diagrama entidade relacional

De modo a armazenar os dados pretendidos na base de dados foi necessário considerar três entidades: autor (*author*), publicações (*work*) e informação scopus (*scopus_info*). Estas entidades são traduzidas em tabelas na base de dados, em que os atributos destas tabelas são os dados extraídos das *API*'s. A Tabela *author_work* representa a relação N-N entre a tabela *author* e a tabela *work*, significando que um autor pode ter várias publicações e uma publicações pode ter ser realizada por vários autores.

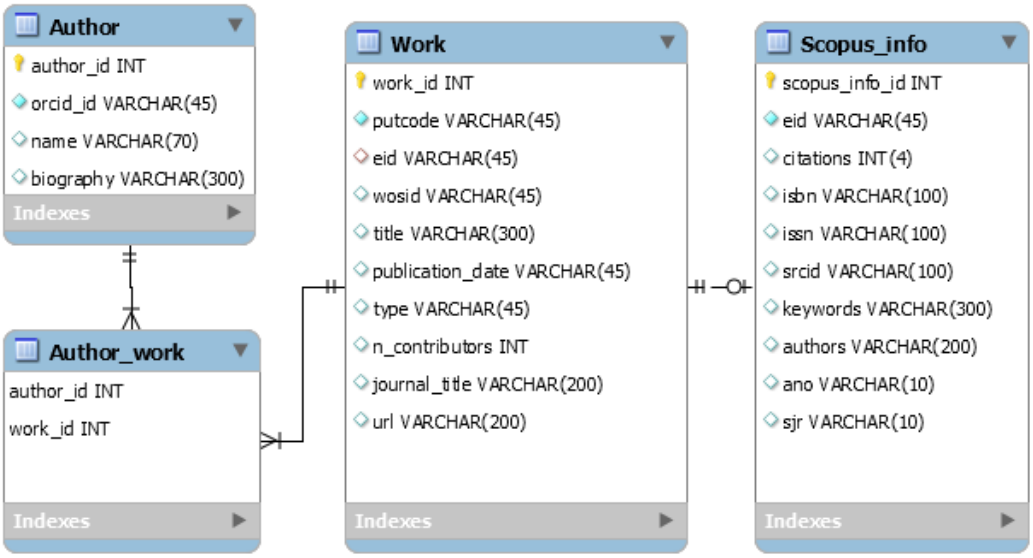

Figura 5: Base de Dados

4.2.2 Descrição tabelas

Esta secção faz a descrição dos atributos das tabelas que representam cada entidade.

Author		
Atributo	**Descrição**	**Info.**
author_id	Identificador único interno do autor	PK, NN, AI
orcid_id	ORCID iD do autor	NN
name	Nome do autor	-
biography	Pequena biografia do autor	-

PK - *Primary key*; **NN** - *Not null*; **AI** - *Auto increment*;

Tabela 1: Descrição de atributos da tabela *Author*

Work		
Atributo	**Descrição**	**Info.**
work_id	Identificador único interno da publicação	PK, NN, AI
putcode	Identificador ORCID da publicação	NN, UQ
eid	Identificador Scopus da publicação	-
wosid	Identificador *Web of Science* da publicação	-
title	Título da publicação	-
publication_date	Data da publicação	-
type	Tipo de publicação	-
n_contributors	Número de contribuidores	-
journal_title	Nome do *publisher*	-
url	*Link* de acesso à publicação	-

PK - *Primary key*; **NN** - *Not null*; **AI** - *Auto increment*; **UQ** - *Unique*;

Tabela 2: Descrição de atributos da tabela *Work*

Scopus_info		
Atributo	**Descrição**	**Info.**
scopus_info	Id. único interno da informação SCOPUS de uma publicação	PK, NN, AI
eid	Identificador Scopus da publicação	NN, UQ
citations	Número de citações	-
isbn	*ISBN* do *publisher*	-
issn	*ISSN* do *publisher*	-
srcid	*SRCID* do *publisher*	-
keywords	*Keywords* da publicação	-
authors	Autores da publicação	-

PK - *Primary key*; **NN** - *Not null*; **AI** - *Auto increment*; **UQ** - *Unique*;

Tabela 3: Descrição de atributos da tabela *Scopus_ info*

4.3 Flask *API*

A *API* desenvolvida é constituída por três *endpoints*, cada *endpoint* fornece informação de uma entidade/tabela da base de dados (autor, publicação,

informação *scopus*).

A seguinte lista apresenta os *endpoints* produzidos:

1. api/authors faz a seleção de todos registos presentes na tabela *author* da base de dados. Caso seja adicionada a *flag* "orcid_id" à *query string* é obtido um utilizador específico. Este *endpoint* é consumido na listagem de todos os autores na *web interface*.

2. api/works faz a seleção de todos as publicações da base de dados, no entanto utilizado a *flag* "from" com o *ORCID iD* de um autor é possível obter apenas as suas publicações. Neste trabalho não foi necessário recorrer ao *endpoint* sem *flags* no entanto é utilizado com a *flag* "from" para a listagem das publicações de um autor.

3. api/scopus faz também a seleção de todos os registos da tabela *scopus_info* da base de dados, sendo possível fazer a filtragem de apenas um registo passando a *flag* "eid". Este *endpoint* é consumido quando é feita uma listagem da informação *scopus* de uma publicação.

4.4 *Web Interface*

A aplicação *web* é uma simples aplicação gerada com o *express generator* com *view engine pug*. A primeira página apresenta todos os utilizadores na base de dados. Destes é apresentado o ORCID iD, nome e uma curta biografia.

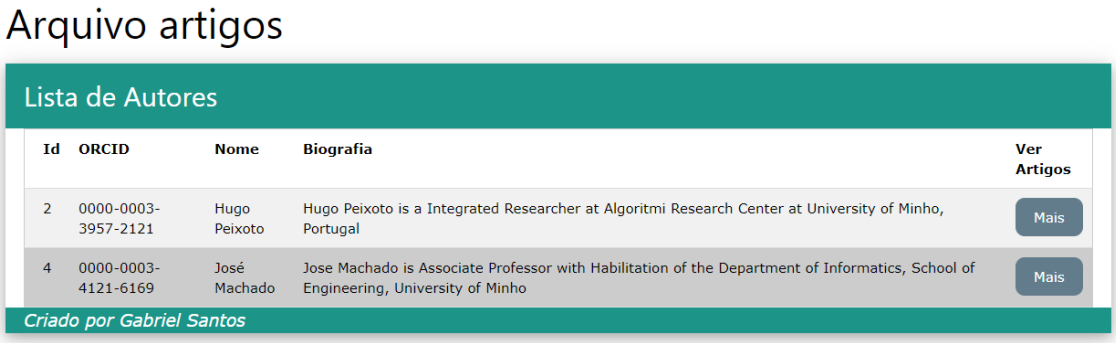

Figura 6: Lista de Autores

Clicando no botão "Mais" de um determinado autor são apresentadas todas as publicações em que o autor contribuiu. A seguinte tabela mostra algumas das publicações de um autor (Figura 5), além de toda a informação é possível ver a informação SCOPUS das publicações. Esta apenas existe em publicações com *EID*. É também possível fazer a organização das publicações (crescente, decrescente) por qualquer coluna da tabela.

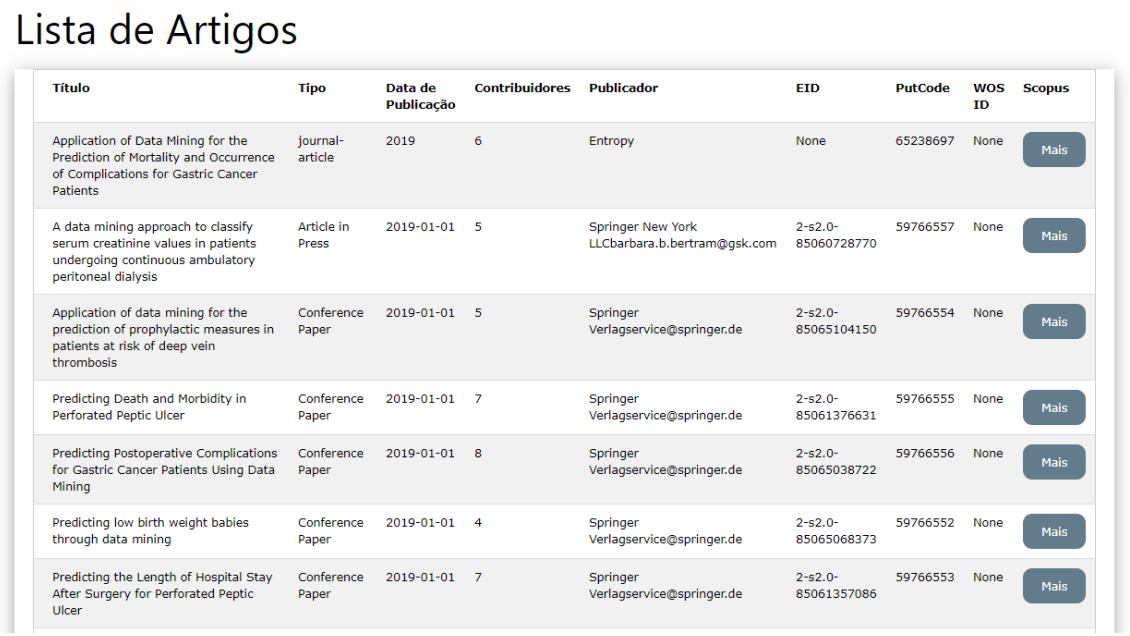

Figura 7: Lista de publicações de um autor

Finalmente, clicando no botão "Mais" numa publicação com *EID* é possível ter alguma informação extra. A informação obtida em relação ao *sjr* bem como *quartil* são obtidas através da *API* pública da plataforma scimago.

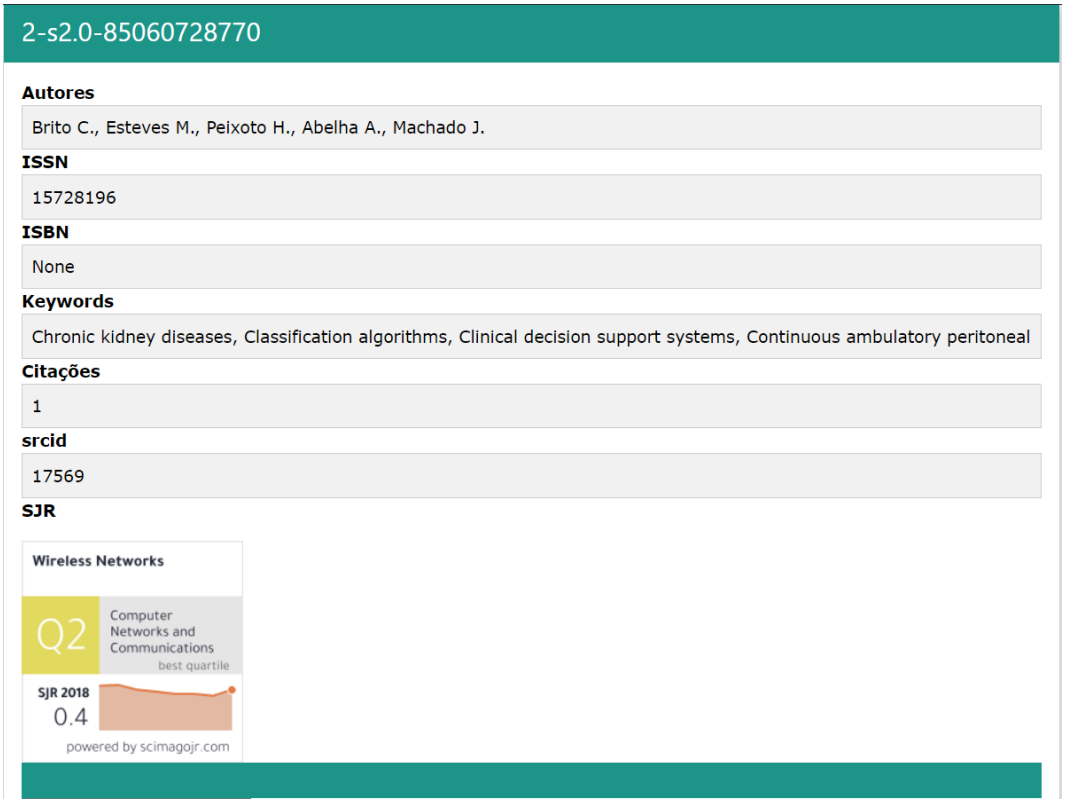

Figura 8: Informação SCOPUS de uma publicação

5 Análise de Desempenho e Possíveis Melhorias

Apesar de neste caso, não se tratar da maior preocupação, a *performance* de um programa é um grande fator que determina a sua usabilidade. Ainda que funcional, um programa que não realiza as tarefas a que se propõe em tempo útil, torna-se útil. Não sendo este o caso, visto que a operação mais custosa, é no consumo dos *endpoints* da plataforma SCOPUS. A obtenção desta informação demora à volta de 10 minutos para 200 publicações.

Existem algumas otimizações que podiam ser implementadas de modo a reduzir este tempo, como por exemplo a adição de *threading*. Vários processos a correr ao mesmo tempo podiam acelerar o processo de obtenção de informação diminuindo o tempo que o processo originalmente demoraria, este tempo dependeria do número de *threads* utilizadas (aproximadamente metade do tempo original com 2 *threads*, um terço com 3 *threads*, etc..). Esta implementação seria benéfica a nível temporal no entanto mais pesada a nível computacional, sendo talvez necessário encontrar um balanço entre ambos.

Na implementação do sistema atual não são verificadas publicações já existentes na base de dados antes de fazer a obtenção de novas publicações. Caso fossem feitas estas verificações podia-se reduzir o número de pedidos a realizar a cada nova inserção de um autor. Esta medida só seria benéfica caso publicações de um autor já se encontrassem na base de dados graças a anteriores inserções. Outro fator a ter é conta é, informações acerca de uma publicação (*work*) tendem a não sofrer alterações (título, número contribuintes, etc) no entanto informações presentes na plataforma SCOPUS tendem (sjr, citações, etc), por isso talvez seja oportuno manter os pedidos à *API* SCOPUS para que a informação do sistema está sempre atualizada.

As duas medidas anteriormente mencionadas são apenas duas possíveis melhorias para acelerar o processo em questão, certamente existem outras cuja implementação fosse mais benéfica para o sistema. Não serão apresentadas medidas para os restantes processos pois a sua execução é quase imediata.

6 Conclusão

A resolução deste trabalho foi bastante importante e enriquecedora, pois permitiu perceber e interiorizar melhor os temas abordados nas aulas da Unidade Curricular de Interoperabilidade Semântica, bem como obter conhecimentos de como se organiza e funciona a área de *Research and Development* no mundo académico.

Numa primeira fase, de tratamento das *responses* das *API*'s foi tida alguma dificuldade devido à inconsistência de dados, a grande quantidade de valores nulos e em falta em alguns campos gerava constantes erros, foram então produzidos *handlers* para tratar estes campos.

Já numa fase posterior, no que toca à implementação dos restantes componentes não foram sentidas grandes dificuldades, graças a conhecimentos obtidos em outras unidades curriculares.

Em suma, é feita uma apreciação positiva em relação ao trabalho desenvolvido, visto que a implementação de todas as funcionalidades propostas foi conseguida. O grupo conseguiu tirar partido dos conhecimentos obtidos no projecto, sentido-se capaz de, num contexto futuro, aplicar os conceitos subjacentes de forma eficaz. É evidente que, num outro contexto, seria benéfico que fossem implementadas um maior conjunto de funcionalidades para a obtenção de um sistema mais completo.

Referências

[1] Elsevier. Elsevier Developer Portal. API Catalog: https://dev.elsevier.com/.

[2] ORCID. Orcid member support center. API Documentation: https://members.orcid.org/api.