Interoperabilidade Semântica

Inãki Uralde (E8261), João Nuno Almeida (A75209), Luís Fernandes (A74748)

Relatório Final do Trabalho Prático

Departamento de Informática
Universidade do Minho

Junho 2018

Contents

Chapter 1

Introdução

Cada vez mais se verifica a existência de softwares informáticos feitos à medida de cada entidade. É natural que tais sistemas, por serem feitos consoante a necessidade de cada um, não sigam determinados *standards*. Nestes casos é necessário garantir a existência de Interoperabilidade Semântica em sistemas onde a comunicação com outros sistemas, é uma prática frequente. A Interoperabilidade Semântica trata-se da capacidade que um sistema recetor possui, de receber e interpretar informações recolhidas, da mesma maneira que o sistema transmissor as interpreta [1].

A plataforma ORCID, tem como objetivo compilar os trabalhos e estudos de investigadores científicos, das áreas mais variadas [2]. Este sistema fornece a cada investigador um identificador único, e é de sua responsabilidade atualizar os seus dados ao longo do tempo. Este sistema fornece ainda uma API pública que permite extrair informações diversas guardadas na plataforma. Usaremos neste projeto a capacidade de extração da listagem de trabalhos de cada investigador em formato *JSON*.

O objetivo deste projeto passa por criar um programa que nos permita extrair informações acerca das publicações de cada autor, inserido na plataforma *ORCID*. Este programa deve ser capaz de extrair a informação da plataforma, recorrendo aos métodos que a sua API pública disponibiliza e a uma ferramenta de transferência de dados chamada *Curl*. Devemos ser capazes de seguida, de apresentar determinador parâmetros (título, ano, local de publicação, EID, WOS, DBLP) dos resultados extraídos numa interface Web (mais apelativa). Os resultados apresentados não devem conter discrepâncias em relação aqueles que podem ser consultados no sistema *ORCID*, provando assim a existência de Interoperabilidade Semântica entre o nosso sistema e a plataforma *ORCID*.

Uma segunda intenção deste projeto seria a de comprovar a existência de Interoperabilidade Semântica intra-plataformas, usando campos encontrados na plataforma *ORCID*, para retirar campos como autores, citações, número de citações nos últimos 3 anos e o SJR, de outras plataformas que guardam estas informações, como por exemplo o *SCOPUS*. No entanto, a API pública destas plataformas encontrava-se indisponível, impossibilitando a realização deste ponto.

No público alvo deste projeto, inserem-se todos os interessados da plataforma *ORCID*, visto a capacidade do mesmo em apresentar resultados, de forma mais agradável e apelativa, através da interface Web. A consistência e validade dos resultados é garantida através de dois programas implementados na linguagem *Python*.

Chapter 2

Implementação

Nesta secção iremos justificar as decisões tomadas no que toca à implementação do programa de extração e à interface Web. Iremos ainda descrever em que é consiste cada programa e a sua utilidade no projeto como um todo.

2.1 Programas Python

2.1.1 Programa Principal Python

O programa responsável pela extração das publicações dos autores, foi implementado na linguagem de programação *Python*. Tomámos esta decisão, devido à nossa familiarização com a mesma e à facilidade que esta proporciona em implementar o programa que idealizamos para realizar as tarefas propostas no enunciado, muito devido às bibliotecas disponibilizadas.

Este programa oferece um menu ao utilizador, apresentado mais em baixo neste documento, com várias funcionalidades, sendo que as mesmas permitem a manipulação da lista de ORCIDs (ou autores, uma vez que cada ORCID é relativo a um autor) cuja lista de trabalhos gostariam de ver exibidas no local para esse efeito.

Como já foi mencionado anteriormente, para atingir este objetivo foi utilizada a ferramenta de transferência de dados *Curl* e recorrendo à API pública disponibilizada pela plataforma ORCID. Uma vez conseguidos os ficheiros *JSON* relativos aos ORCIDs presentes na lista, é tempo de os integrar na página Web onde pretendemos apresentar as informações pedidas, este processo será apresentado mais à frente na secção relativa à **Interface Web**.

Figure 2.1: Menu principal do Programa Python.

2.1.2 *Backgroud Server*

Este segundo programa desenvolvido consiste num pequeno servidor que deve ficar sempre a correr em *background* como garantia de atualização permanente dos dados dos ORCIDs (autores) presentes na lista. O intervalo de tempo em que a atualização é efetuada é adaptável e pode tanto ser produzida de 10 em 10 segundos como de 24 em 24 horas, não havendo qualquer limitação a este nível.

Esta atualização permanente dos dados é importante porque é assim que asseguramos a coerência dos dados entre plataformas, uma vez que, sem este segundo servidor, os trabalhos dos autores presentes na lista seriam apenas atualizados quando nova inserção ou remoção à lista e respetivo *export* fosse efetuado.

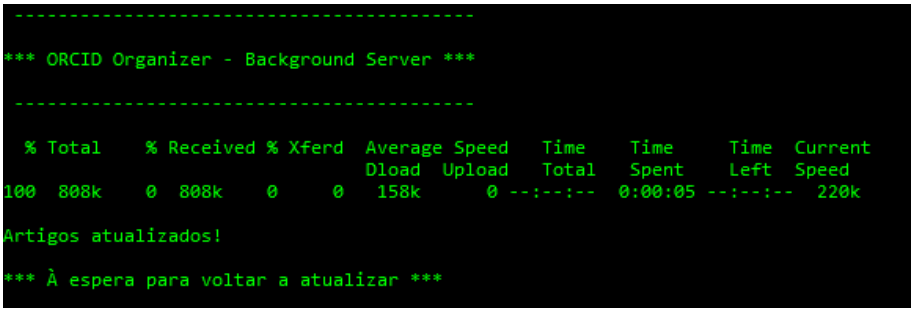

Figure 2.2: Server a atualizar lista de trabalhos.

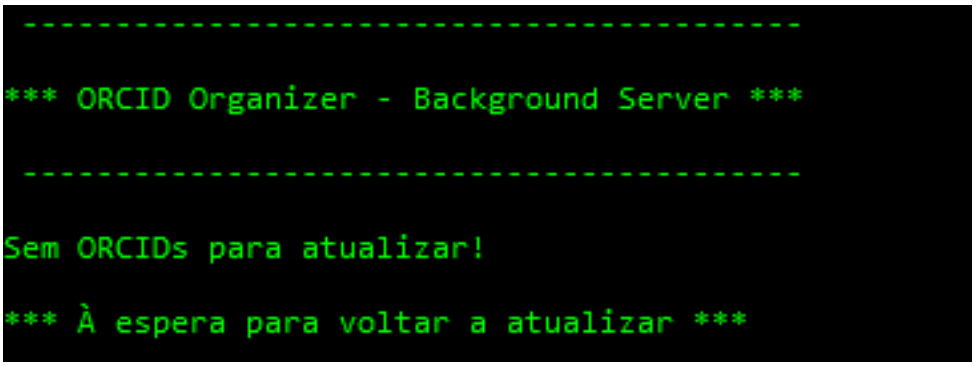

Figure 2.3: Server sem trabalhos para atualizar (lista encontra-se vazia).

2.2 Interface Web

A componente de *Frontend* da interface Web foi desenvolvida recorrendo à linguagem HTML e à *framework Bootstrap*, por ser a mais adequada para proceder ao desenvolvimento de páginas Web, e, devido às qualidades gráficas que esta *framework* acrescenta às mesmas. A componente de *Backend* foi desenvolvida na linguagem *Javascript*, com recurso à *framework jQuery*. Tomámos esta decisão devido à nossa familiarização com o *Javascript* e devido à aptidão que o *jQuery* exibe no tratamento e leitura de ficheiros *JSON* (formato em que foram extraídas as informações do sistema *ORCID*).

Figure 2.4: Interface Web.

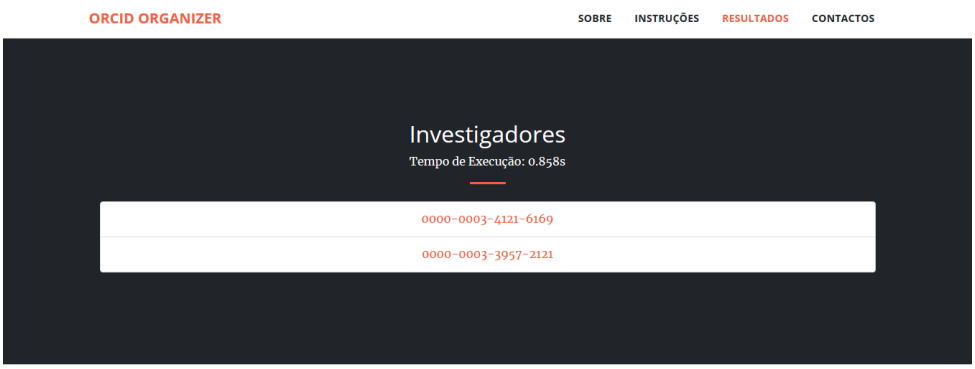

Figure 2.5: Lista de ORCIDs (autores).

TRABALHOS ACADÉMICOS DE 0000-0003-4121-6169

Titulo	Ano	Local da Publicação	EID	WOS	DBLP	SJR
Iron Value Classification in Patients Undergoing Continuous Ambulatory Peritoneal Dialysis using Data Mining	2018	Authenticus	-----	-----	dblp:conf/ict4ageingwell /PeixotoP0A518	-----
Mobile collaborative augmented reality and business intelligence: A system to support elderly people's self-care	2018	Scopus - Elsevier	2-s2.0-85045332393	-----	-----	2194-5357
New approach to an openEHR introduction in a portuguese healthcare facility	2018	Scopus - Elsevier	2-s2.0-85045320739	-----	-----	2194-5357
Step towards a pervasive data system for intensive care medicine	2018	Scopus - Elsevier	2-s2.0-85045347385	-----	-----	2194-5357
A Data Mining Approach for Cardiovascular Diagnosis	2017	Authenticus	-----	-----	dblp:journals/cejcs /PereiraP0A17	-----
A case based methodology for problem solving aiming at knee osteoarthritis detection	2017	Scopus - Elsevier	2-s2.0-85009818557	WOS:000416517100028	dblp:conf/scdm /EstevesV0AN16	-----
A case-based approach to colorectal cancer detection	2017	Scopus - Elsevier	2-s2.0-85016116954	WOS:000425796900050	-----	-----
A pervasive business intelligence solution to manage Portuguese misericordia	2017	Scopus - Elsevier	2-s2.0-85025135279	-----	dblp:conf/ict4ageingwell /CoelhoPS0A17	-----

Figure 2.6: Tabela com as informações relativas aos ORCIDs (autores).

2.3 Execução

O manual de instruções para a execução de todo o fluxo do nosso projeto, está descrito numa das secções da nossa interface Web. No entanto, não deixaremos de enumerar aqui quais as tarefas a desempenhar para usufruir de todas as funcionalidades que oferecemos no nosso programa, esclarecendo o modo de funcionamento do mesmo.

1. **Bash** - É necessário proceder à abertura de duas linhas de comandos na sua máquina pessoal, pois esses serão os ambientes de execução dos programas em Python.

2. **Programa Principal Python** - É necessário de seguida executar o programa *ORCID.py*. O utilizador terá aos seu dispôr um menu principal constituído por 5 opções:

   a) **1 - Listar ORCIDs** - Produz uma listagem dos ORCIDs dos investigadores, cujas informações das publicações vão ser extraídas.

   b) **2 - Adicionar ORCID** - Permite adicionar um determinado ORCID da lista a extrair.

   c) **3 - Eliminar ORCID** - Permite eliminar um determinado ORCID da lista a extrair.

d) **4 - Exportar Trabalhos** - Extrai as informações das publicações de todos os investigadores (no formato JSON), cujos ORCIDs se encontravam na listagem de ORCIDs a extrair no instante da execução do comando.

e) **5 - Limpar Lista** - Limpa toda a lista de ORCIDs aramazenados até ao momento.

f) **0 - Sair** - Termina a execução do programa.

3. **ORCID Server** - Trata-se do segundo programa desenvolvido em Python. Este oferece um *backgroud server* que, de x em x tempo (sendo este x adaptável), corre a lista de ORCIDs e atualiza as informações das publicações de todos os ORCIDs presentes.

4. **Consulta de Resultados** - Poderá a partir da interface Web criada para o efeito, consultar os resultados encontrados, referentes às publicações do(s) investigador(es) que procurou.

Chapter 3

Medidas de Desempenho

Apesar de neste caso, não se tratar da nossa preocupação mais imediata, a *performance* de um programa marca em muito aquela que é a sua usabilidade. Ainda que funcional, um programa que não realiza as tarefas a que se propõe em tempo útil, torna-se inútil. Não sendo este o caso, visto que a operação mais custosa demora, no pior dos casos, uma dezena de segundos, ainda assim decidimos dedicar esta secção para estudar o estado atual do nosso sistema em termos de tempo de execução, e propôr duas alterações que futuramente poderiam ser implementadas para aprimorar a *performance* do mesmo.

Quase todas as operações que podem ser realizadas no nosso sistema, são praticamente instantâneas, à excepção de uma: a extração das informações dos investigadores através do *Curl* e da API do *ORCID*.

Table 3.1: Medidas de desempenho na execução de "Exportar Trabalhos"

Teste (#)	Nº de ORCID	Tempo (s)
1	1	2.648
2	3	8.547
3	7	19.942
4	15	41.786
5	30	89.878

O *Curl* trata-se de uma ferramenta que permite a tranferência de dados com o recurso a diversos protocolos de redes [3]. Deste modo, muitos dos tempos medidos, estarão a ser influenciados, pela qualidade da rede onde foram executadas as extrações das informações a partir da plataforma *ORCID*, e pelo tamanho dos resultados extraídos. Dito isto, tivemos a preocupação e o cuidado de medir os tempos, não só das extrações em si, mas logo a partir do momento em que começam as operações que nada têm a ver com a transferência dos dados, até terminar a escrita do último resultado no último ficheiro. Para efeitos de teste extraímos sucessivas vezes os trabalhos do mesmo investigador para garantir que todos os resultados tinham o mesmo tamanho.

Analisando os dados, podemos ver que os tempos seguem uma distribuição linear, i.e, o à medida que o número de pedidos em lista se acumula, o tempo para a realização da operação de extração cresce na mesma proporção. Tal como já tinha sido referido acima, tratam-se de tempos perfeitamente aceitáveis, não havendo nenhum caso a partir do qual seja detetada uma quebra abrupta na *performance* do programa.

Ainda assim entendemos que existem duas medidas que poderiam ser tomadas no futuro, e que no nosso entender ajudariam a melhorar ainda mais estes tempos de execução.

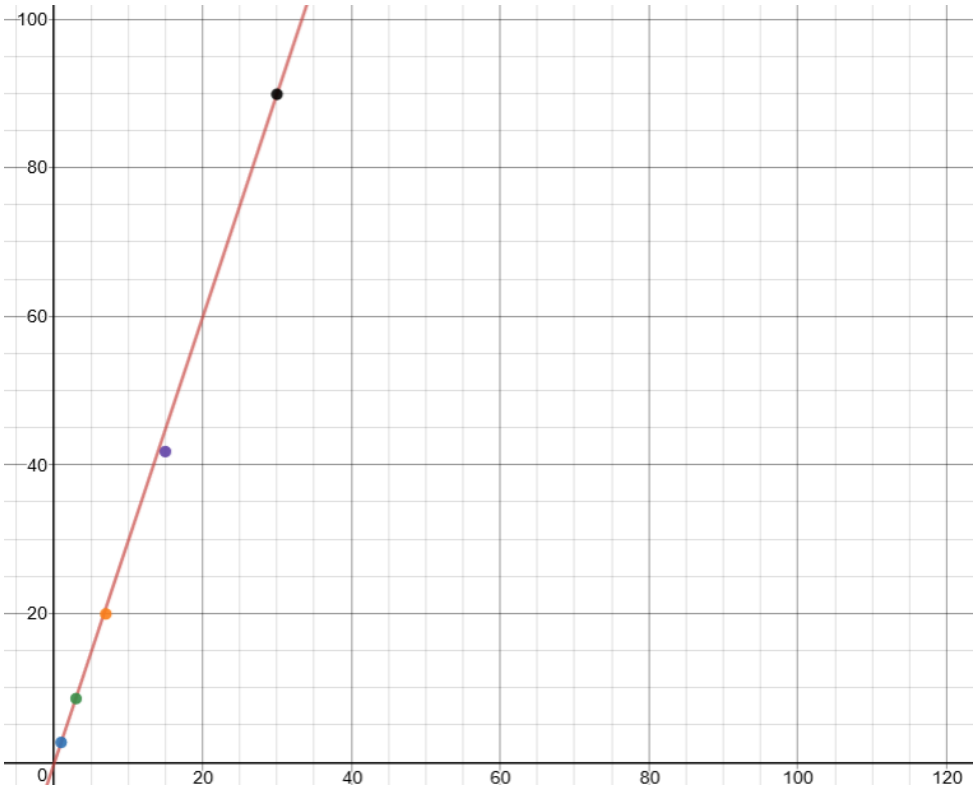

Figure 3.1: Regressão Linear dos tempos estudados.

A primeira passa por remover a escrita dos resultados num ficheiro *JSON*. Esta é uma operação que tem custos, e a decisão de a realizar foi tomada devido à facilidade que depois teriamos em tratar os dados na *Backend* usando a *framework jQuery*, que tem métodos próprios para leitura de ficheiros no formato *JSON*. Posto isto, e pensando apenas na *performance* entendemos que seria vantajoso, guardar futuramente os resultados numa estrutura dentro do ambiente de execução do programa, em detrimento da escrita num ficheiro.

A segunda medida passa por criar mais do que um ambiente de execução. Com a criação de ambientes de execução paralelos, em que cada um ficaria responsável por extrair um determinado número de trabalhos da listagem total, seriamos capazes de reduzir o tempo total de execução numa razão igual ao número de programas a executar em paralelo (metade com dois programas, um terço com três programas...).

Chapter 4

Conclusão

Em suma, com este trabalho fomos capazes de, pela primeira vez, interagir com o conceito de Interoperabilidade Semântica. Pela primeira vez, preocupamo-nos em garantir consistência de dados, não só, no nosso ambiente de execução, mas também na eventualidade de existir a necessidade de garantir a transmissão entre dois sistemas. Fomos capazes de desenvolver um programa na linguagem de programação *Python*, capaz de recorrer a uma API e a um serivço Web para extrair resultados de uma plataforma. Desenvolvemos as nossas habilidades de programação Web através da criação de um interface, responsiva e adaptável, para apresentação de resultados. Implementamos um *Backend* capaz de ler e tratar os dados extraídos no formato JSON, sem os deturpar. Integramos esse mesmo *Backend*, no *Frontend* desenvolvido com ajuda da *Framework Bootstrap*. E, por último, mas não menos importante, desenvolvemos um pequeno programa em Python capaz de garantir que os dados se mantenham atualizados, sendo que o mesmo atualiza no intervalo de tempo que o utilizador quiser.

Estudamos questões relacionadas com a *performance* dos nossos programas, identificamos os maiores problemas e deduzimos soluções para os mesmos, a serem implementadas futuramente.

Podemos dizer que com este projeto, não só desenvolvemos as nossas capacidades nas respetivas linguagens de programação usadas, mas sobretudo percebemos o significado do conceito de Interoperabilidade Semântica e a sua importância nos dias que correm.

Bibliography

[1] https://pt.wikipedia.org/wiki/Interoperabilidade_semântica

[2] https://elearning.uminho.pt/bbcswebdav/pid-802491-dt-content-rid-1816231_1/courses/1718.H508Q7_2/Enunicado

[3] https://en.wikipedia.org/wiki/CURL