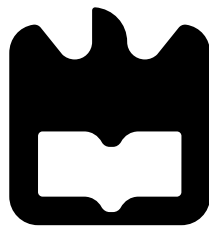


Universidade de Aveiro

InfoCountries

Diogo Ferreira, Luís Leira, Miguel Silva



InfoCountries

Departamento de Electrónica, Telecomunicações e
Informática

Engenharia de Dados e Conhecimento

Diogo Ferreira, Luís Leira, Miguel Silva
76504, 76514, 76450

diogodanielsoaresferreira@ua.pt, luisleira@ua.pt, maps@ua.pt

14 de Dezembro de 2017

Conteúdo

1	Introdução	2
2	Estrutura do projeto	4
3	Dados, suas fontes e sua transformação	6
4	Operações sobre os dados (<i>SPARQL</i>)	9
5	Inferências	11
6	A publicação de dados semânticos através de <i>RDFa</i>	13
7	A integração de dados da <i>Wikidata</i>	16
8	Funcionalidades da aplicação (<i>UI</i>)	19
9	Conclusões	25
10	Configuração para executar a aplicação	27

Capítulo 1

Introdução

O presente relatório descreve uma aplicação web efetuada no âmbito da unidade curricular de Engenharia de Dados e Conhecimento [1]. Neste projeto tivemos como principal objetivo construir uma aplicação web utilizando *Web Semântica* integrada nas páginas web, com recurso a uma base de dados baseada em grafos, nomeadamente *GraphDB*.

A principal inspiração para este projeto foi a possibilidade de comparar países relativamente a determinadas métricas e o nível de vida de cada um, e relacionar esses dados com outros dados atualizados existentes na *Internet*.

Para isso, foram utilizados dados *open-source* retirados de uma fonte encontrada na Internet, contendo uma lista extensa de países e informação sobre os mesmos (p.e. área, população, dívida pública, número de mortes por HIV/AIDS). Essa informação foi normalizada e convertida para *N-triples* com *XSLT*, para posteriormente ser inserida na base de dados.

Para além de informação estática (i.e. recolhida com base no ficheiro em *N-triples*), também é possível atualizar os dados armazenados com novos dados da *Wikidata*. É possível integrar dois tipos de relações com os novos dados: relações entre países (p.e. Portugal partilha fronteira com Espanha, ou Portugal têm uma relação diplomática com Inglaterra) ou novos atributos dos países (p.e. a bandeira do país, a capital, os primeiros-ministros, a moeda).

A aplicação web também permite efetuar dois tipos de inferências. Existem inferências reflexivas, que são efetuadas com as novas relações entre países (se Portugal faz fronteira com Espanha, Espanha faz fronteira com Portugal) e inferências com base nos dados presentes (por exemplo, inferir quais são os melhores países para investir, ou os melhores países para se viver).

Dado que cada utilizador deve possuir uma conta na aplicação, é possível marcar países como favoritos para acesso rápido.

Neste relatório é explicado detalhadamente todo o processo de implementação da aplicação, bem como as escolhas de engenharia efetuadas e as principais dificuldades encontradas.

Capítulo 2

Estrutura do projeto

Neste capítulo é descrita a estrutura geral do projeto, com a indicação dos ficheiros e pastas mais importantes.

No diretório do projeto podemos encontrar duas pastas relevantes. A pasta *Projeto2* contém informação relativamente ao projeto e à sua configuração. Dentro da pasta *app* temos acesso a vários ficheiros importantes.

- O ficheiro *dbInterface* contém as funções necessárias para a criação automática do repositório e conversão de dados entre diferentes formatos. Para além disso, contém as funções necessárias para efetuar *queries* à base de dados e inserção de novos dados.
- O ficheiro *businessLogic* contém funções que efetuam *queries* e alterações de dados e permitem uma abstração relativamente ao motor de base de dados utilizado pelas camadas superiores.
- O ficheiro *inferences* contém as funções que efetuam as inferências tendo em conta os dados presentes na base de dados. Para além disso, também inserem as inferências efetuadas na base de dados.
- O ficheiro *wikidataInterface* contém as funções necessárias para a comunicação com a *Wikidata* e a obtenção de novos atributos para os países.
- O ficheiro *presentationLogic* agrega os dados necessários para cada vista de acordo com a sua necessidade, e envia os sujeitos, predicados e objetos das relações para a interface devidamente filtrados.
- O ficheiro *views* contém as funções que irão receber os pedidos *web* e efetuam o processamento necessário para devolver uma página *HTML* para o cliente.

Existem ainda quatro pastas relevantes para a leitura do relatório:

- A pasta *static* contém os conteúdos estáticos para a página web como imagens, ícones, ficheiros de Javascript, ficheiros de estilo CSS, entre outros.
- A pasta *templates* contém os ficheiros HTML para cada página que será apresentada.
- A pasta *xml_files* contém os dois ficheiros de *XML* utilizados: o ficheiro com todos os países e os seus dados e o ficheiro *XSLT* com o processo de transformação efetuado para transformar o ficheiro *XML* em *N-triples*.
- A pasta *nt_files* contém o ficheiro com os dados dos países no formato *N-triples* após a transformação referida no ponto anterior. Os dados podem ser utilizados como *backup* ou serem inseridos na base de dados.

```
Projeto2
+-- Projeto2
+-- app
|  |-- dbInterface
|  |-- businessLogic
|  |-- inferences
|  |-- wikidataInterface
|  |-- presentationLogic
|  |-- views
|  +-- static
|  +-- templates
|  +-- xml_files
|  +-- nt_files
```

Listing 2.1: Estrutura do projeto em árvore com as pastas e ficheiros mais relevantes.

Capítulo 3

Dados, suas fontes e sua transformação

Neste capítulo é feita a descrição dos dados recebidos e das suas fontes.

De forma a tornar o processo da criação e importação de dados para o repositório da GraphDB rápido e transparente para o utilizador, foi desenvolvida uma função, *setupRepository()* (presente no ficheiro *dbInterface*), que irá fazer toda a configuração de forma automática. Esta função começa por verificar a existência do repositório usado pelo nosso projeto, o repositório *countries*. Caso seja verificada a não existência do repositório, este é criado usando um ficheiro de configuração necessário por parte da API usada.

Após a criação do repositório, é efetuada a transformação dos dados presentes no ficheiro *XML* para o formato *RDF N-triples* usando uma transformação *XSLT* específica. De seguida, é feita a inserção dos triplos no repositório *countries* recém-criado. Assim sendo, apenas é necessário executar o servidor (ativo na porta 8000) para a base de dados se configurar automaticamente.

O *dataset* utilizado foi recolhido da *Internet* [2] e contém dados sobre vários países. Os dados estão representados em 45 métricas diferentes e opcionais, e são representados em forma de elementos *XML*. Algumas dessas métricas são: produto interno bruto per capita; área; dívida externa; taxa de mortalidade infantil; utilizadores de *Internet*; exportações de gás natural; população; entre outros. A transformação de *XML* para *N-triples* é realizada usando *XSLT*.

Na transformação usando *XSLT*, processamos cada país um a um, e para cada país é verificada a existência de cada uma das métricas na forma de elemento *XML*. Caso o país tenha a métrica em questão, é então criado um triplo *RDF* no formato *N-triples*. Isto repete-se para todas as métricas em que cada métrica dá origem a um triplo novo.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output method="text" indent="yes" omit-xml-declaration="yes" />

<xsl:variable name='base_uri'>
  <xsl:text>http://www.ua.pt/ensino/uc/2380/projeto2/</xsl:text>
</xsl:variable>

<xsl:variable name='newline'>
  <xsl:text>&#xa;</xsl:text>
</xsl:variable>

<xsl:template match="/">
<xsl:for-each select="countries/country">
  <xsl:if test="name != ''">
    <xsl:value-of select="concat('&lt;', $base_uri, 'country/',
      position(), '&gt; ')" />
    <xsl:value-of select="concat('&lt;', $base_uri,
      'countryProperty/1', '&gt; ')" />
    <xsl:value-of select="concat('&quot;', name, '&quot;', ' .',
      $newline)" />
  </xsl:if>
  (...)
  <xsl:if test="unemployment_rate != ''">
    <xsl:value-of select="concat('&lt;', $base_uri, 'country/',
      position(), '&gt; ')" />
    <xsl:value-of select="concat('&lt;', $base_uri,
      'countryProperty/45', '&gt; ')" />
    <xsl:value-of select="concat(unemployment_rate, ' .',
      $newline)" />
  </xsl:if>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>

```

Listing 3.1: Excerto do ficheiro que realiza a transformação *XSLT* de *XML* para *N-triples*. Podemos verificar que, para métrica (no caso, `name` e `unemployment_rate`), caso exista no país a ser analisado, é criado um novo triplo com o país, a respetiva métrica e o seu valor.

Como *URI* base a ser utilizado em todos os triplos foi utilizada a *URL*

da cadeira, seguida do número do projeto. A essa *URI* base é concatenado um identificador extra. Por exemplo, para os predicados criados a partir da transformação, a *URI* do predicado 1, a *URI* será `http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/1`, em que o número 1 indica que é a propriedade 1. Da mesma forma, o país com o ID 1 terá o *URI* `http://www.ua.pt/ensino/uc/2380/projeto2/country/1`.

```
<http://www.ua.pt/ensino/uc/2380/projeto2/country/1>
  <http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/1>
    "Afghanistan" .
<http://www.ua.pt/ensino/uc/2380/projeto2/country/1>
  <http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/2>
    647500 .
<http://www.ua.pt/ensino/uc/2380/projeto2/country/1>
  <http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/3>
    47.02 .
(...)
```

Listing 3.2: Excerto do ficheiro *N-triples* resultante da transformação *XSLT*. Podemos ver três triplos referentes ao país Afeganistão.

Capítulo 4

Operações sobre os dados (*SPARQL*)

Neste capítulo é feita a descrição e exemplos de operações sobre os dados através de *SPARQL*.

No ficheiro *businessLogic* estão contidas a maior parte das funções que efetuam operações utilizando a linguagem *SPARQL* para receber dados armazenados. As 11 funções presentes no *businessLogic* correspondem a *queries*, inserções e remoções de dados presentes na base de dados.

Uma dessas funções é a função *getMetricsOfCountry(country_URI)*. Dado uma *URI* de um país, a função fará uma pesquisa por todos os predicados associados ao país inserido e devolverá um dicionário com a *URI* de cada predicado como sendo a chave, e os valores do dicionário como sendo os valores das relações onde o país inserido é o sujeito e a chave é o predicado.

```
def getMetricsOfCountry(country_uri):
    query = """
        select ?predicate ?value
        WHERE{
            ?country ?predicate ?value .
            filter(?country=<"""+country_uri+""">) .
        }
    """
    results = executeQuery(query)
    (...)
```

Listing 4.1: Excerto da função *getMetricsOfCountry(country_uri)*. Podemos verificar que a query tem em conta o *URI* do país inserido, e que devolve os predicados e objetos referentes a esse *URI*.

Uma das funções mais utilizadas é a *getNameOfSubject(subject)*. Tendo como argumento a *URI* de um predicado, irá devolver a *label* desse predicado para ser legível para o utilizador, sabendo que o nome desse predicado se encontra na base de dados representado com o predicado *foaf:name*.

Outro exemplo de uma função é *addToFavorites()* que dado um ID de um utilizador e uma *URI* de um país presente na base de dados, cria uma relação de favorito. Assim sendo, o país fica associado como favorito do utilizador.

```
def addToFavorites(user_id, countryURI):
    data = "<"+BASE_URL+"/user/"+user_id+">
          <"+BASE_URL+"/favorite">
          <"+BASE_URL+"/country/"+countryURI+"> .\n"
    update = ""INSERT DATA{"" + data + ""}""
    ins = executeInsert(update)
    (...)
```

Listing 4.2: A função *addToFavorites* irá inserir um elemento na base de dados com o predicado */favorite* entre o ID utilizador e a *URI* país.

Outras operações sobre dados encontram-se descritas no capítulo 5 - *Inferências*.

Capítulo 5

Inferências

Neste capítulo são descritas as inferências realizadas com os dados disponíveis.

Na aplicação existem quatro tipos de inferências que são efetuadas tendo em conta os dados armazenados na base de dados. Estas operações e as funções que serão invocadas na descrição encontram-se no ficheiro *inferences*.

A primeira inferência, chamada inferência reflexiva, é a mais simples de realizar e decorre da propriedade reflexiva de alguns predicados. Para as relações entre países, se a relação se verifica num sentido, verifica-se também no sentido contrário, ou seja, se se verifica sujeito-predicado-objeto, também se verifica objeto-predicado-sujeito. Neste caso, aplica-se às relações de partilha de fronteira e de relações diplomáticas (p.e., se Portugal tem relações diplomáticas com Espanha, Espanha tem relações diplomáticas com Portugal). Esta inferência é realizada pela função *getReflexiveInferences()*, que devolve todas as inferências reflexivas realizadas com os dados existentes.

As outras três inferências são inferências por classificação. É possível deduzir quais os melhores países para viver (função *getBestCountriesToLive()*), quais os melhores países para investir (função *getBestCountriesToInvest()*) e quais os países que geram mais tráfego de *Internet* (função *getCountriesWithMoreInternetTraffic()*) apenas com regras simples aplicadas aos seus atributos.

```
PREFIX countries: <"" + BASE_URL + ""/>countryProperty/>
select ?countryURI
WHERE{
    ?countryURI countries:45 ?unemploymentRate .
    ?countryURI countries:26 ?laborForce .
    ?countryURI countries:23 ?investment .
    ?countryURI countries:39 ?publicDebt .
```

```

    ?countryURI countries:20 ?inflationRate .
    ?countryURI countries:25 ?lifeExpectancy .
    ?countryURI countries:9 ?exports .
    filter(?unemploymentRate<10.0)
    filter(?laborForce>100000)
    filter(?investment>15.0)
    filter(?publicDebt<175.0)
    filter(?inflationRate<10.0)
    filter(?lifeExpectancy>75.0)
}
ORDER BY DESC(?exports)

```

Listing 5.1: A função *getBestCountriesToInvest()* irá efetuar uma *query* que filtrará os países por algumas métricas, nomeadamente a taxa de desemprego abaixo de 10%, a inflação abaixo de 10%, a esperança média de vida acima de 75 anos, entre outras.

```

PREFIX countries: <"" + BASE_URL + """/countryProperty/>
select ?countryURI
WHERE{
    ?countryURI countries:43 ?mobileCelularUsers .
    ?countryURI countries:42 ?mainLinesInUse .
    ?countryURI countries:21 ?internetHosts .
    filter(?mobileCelularUsers>10000000)
    filter(?mainLinesInUse>10000000)
    filter(?internetHosts>1056950)
}
ORDER BY DESC(?internetHosts)

```

Listing 5.2: A função *getCountriesWithMoreInternetTraffic()* irá efetuar uma *query* que filtrará os países por algumas métricas, nomeadamente o número de utilizadores de telemóvel acima de 10 milhões, o número de linhas telefónicas ativas acima de 10 milhões, e o número de *hosts* de *Internet* acima de 1056950.

Capítulo 6

A publicação de dados semânticos através de *RDFa*

Neste capítulo é descrita a publicação de dados semânticos através de *RDFa* e apresentados exemplos da sua utilização.

A implementação de *RDFa*, sendo realizada ao nível da camada de apresentação, teve impacto no ficheiro *presentationLogic* e nos ficheiros *HTML*. O primeiro permite recolher e fornecer os dados necessários à implementação de *RDFa* nos ficheiros de apresentação (*HTML*). Estes ficheiros fazem uso de determinados atributos tais como *about*, *content*, *href*, *property*, *rel*, *resource* para classificar as informações adicionais inseridas.

```
# Return list (URI, url, label) of all countries
def getListOfCountries():
    countries = getAllCountries()
    listCountries = [{"uri":uri.split("/", 7)[-1],
                      "url":uri.split("/")[-1], "label":countries[uri]} for uri in
                      countries]
    return listCountries
```

Listing 6.1: A função *getListOfCountries()* irá devolver uma lista de dicionários com informação sobre cada país. Como pode ser visto, a função devolve não apenas o nome do país, mas também a *URI* e a *URL*, para ser integrado no ficheiro *HTML* com *RDFa*. O envio de *URI*'s mantém-se para todas as funções que devolvem dados presentes na base de dados para a interface *HTML*.

No template *HTML* é possível integrar a semântica dos dados sem alterar a apresentação, desde que sejam conhecidos os *URI*'s e os predicados dos objetos que estão a ser mostrados.

```

<ul>
{% for predicate, metric in metrics %}
  <li>
    <a href="/metric/{{ predicate.url }}">
      <span about="{{ predicate.uri }}" property="foaf:name">{{
        predicate.label }}</span>
    </a> -
    {% ifequal metric.uri metric.value %}
      <span about="countries:country/{{ country_id }}"
        property="{{ predicate.uri }}">{{ metric.value
        }}</span>
    {% else %}
      {% ifnotequal metric.url "" %}
        <a href="/country/{{ metric.url }}">
          <span about="countries:country/{{ country_id }}"
            rel="{{ predicate.uri }}">
            <span about="{{ metric.uri }}" property="{{
              metric.pred_nameof }}">{{ metric.value
            }}</span>
          </span>
        </a>
      {% else %}
        <span about="countries:country/{{ country_id }}"
          rel="{{ predicate.uri }}">
          <span about="{{ metric.uri }}" property="{{
            metric.pred_nameof }}">{{ metric.value
          }}</span>
        </span>
      {% endifnotequal %}
    {% endifequal %}
  </li>
{% endfor %}
</ul>
<br>


```

Listing 6.2: Excerto da página HTML de um país. Podemos ver que existem triplos que são inseridos na página através de marcadores *RDFa*. Neste exemplo em particular, também é utilizado o sistema de templates do *Django*.

Assim sendo, é possível extrair dados semânticos a partir da página

HTML, sem ser necessária nenhuma API específica, apenas analisando as relações presentes no conteúdo *HTML*.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ns1:
  <http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/> .

<http://www.ua.pt/ensino/uc/2380/projeto2/country/1>
  <http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/1>
    "Afghanistan";
  <http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/10>
    "21500000000";
  (...)
  <http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/9>
    "446000000";
  ns1:flag <https://upload.wikimedia.org/wikipedia/commons/9/9a/
    Flag_of_Afghanistan.svg> .

<http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/1>
  foaf:name "Country" .

<http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/10>
  foaf:name "GDP" .

(...)

<http://www.ua.pt/ensino/uc/2380/projeto2/countryProperty/9>
  foaf:name "Exports" .
```

Listing 6.3: Exemplo de uma análise de *RDFa* e extração de relações efetuada por uma ferramenta online [3], relativamente à página do país *Afeganistão*.

Capítulo 7

A integração de dados da *Wikidata*

Neste capítulo é explicada como foi efetuada a integração dos dados com outros dados presentes na *Wikidata*.

No contexto deste projeto, a *Wikidata* é utilizada para obter mais dados relativamente a países, de forma a complementar a informação obtida através do *dataset* inicial. Estas operações são efetuadas no ficheiro *wikidataInterface*. Para cada país é efetuada uma pesquisa *SPARQL* para obter a URI utilizada pela *Wikidata* para cada país, se ela existir (função *getWikidataCountry(country_name)*). Essa pesquisa é efetuada utilizando diretamente a API REST providenciada pela *Wikidata*. O resultado vem em formato *XML*, pelo que é necessário transformar o resultado num dicionário para ser facilmente interpretado em *Python* (função *queryWiki(query)*).

```
def getWikidataCountry(country_name):
    query="""
        SELECT ?country ?countryLabel WHERE {
            ?country wdt:P31 wd:Q3624078 .
            ?country rdfs:label ?countryLabel .
            FILTER (lang(?countryLabel) = "en")
            filter(regex(?countryLabel,"^"""+country_name+"""$", "i"))
        }

    """
    result = queryWiki(query)
    if len(result)>0:
        return result[0]
    return None
```

Listing 7.1: A função *getWikidataCountry(country_name)* irá pesquisar por todos os países presentes na *Wikidata* cujo label é o nome do país passado como argumento. Se existir pelo menos um país, a sua *URI* será utilizada verificar os valores dos triplos do país.

Após obter a *URI* de cada país, é possível aceder à entidade e verificar se existem novos atributos que podem ser armazenados localmente. A função *getWikidataImageURL(entity)*, dada uma entidade existente na *Wikidata*, utilizando a API *Wikidata* 0.6.1, devolverá o *URL* da imagem da bandeira de cada país, se existir.

É possível obter dois tipos de dados diferentes a partir de uma entidade de um país da *Wikidata*. A função *createCountryNewPredicates(country, countryEntity)* irá verificar se existe um conjunto de métricas para cada país, como a moeda atual, o local geográfico mais alto, o hino, o continente, a capital, entre outros. Se existirem, a função procederá à criação das respetivas relações na base de dados. É também possível obter relações entre países a partir da *Wikidata*, como a partilha de fronteiras ou relações diplomáticas. A função *createCountriesRelations(countryURI, countryEntity)* pesquisa por essas relações e, caso existam, insere-as na base de dados.

```
def createCountryNewPredicates(country, countryEntity):
    for predicate in WIKIDATA_PREDICATES:
        pred = client.get(predicate[1])
        (...)
        if pred in countryEntity:
            for element in countryEntity.getlist(pred):
                predid = pred.id
                elementid = element.id
                elementlabel = element.label
                newRelations += [(country, predid, elementid)]
                newValues += [(elementid, nameof, elementlabel)]

    (...)
    return newRelations
```

Listing 7.2: A função *createCountryNewPredicates(country, countryEntity)* irá pesquisar por todos os predicados armazenados que podem ser aplicados a uma entidade na *Wikidata* que é um país. Se existir algum objeto para o predicado do país, irá armazená-lo, tal como o se nome, para facilitar a compreensão por humanos na camada de apresentação.

Quando os novos predicados ou relações são inseridas na base de dados, é também inserida uma relação que indica que o país contém relações extraídas da *Wikidata*.

Capítulo 8

Funcionalidades da aplicação (*UI*)

Neste capítulo são explicadas as funcionalidades que estão acessíveis através da interface.

Inicialmente, o utilizador necessita de se registar na aplicação, fornecendo um nome de utilizador e uma palavra-passe. Após o registo, é possível efetuar *login* na aplicação e aceder à página inicial intitulada de "Countries" onde é possível ter uma lista de todos os países existentes na base de dados (figura 8.1).

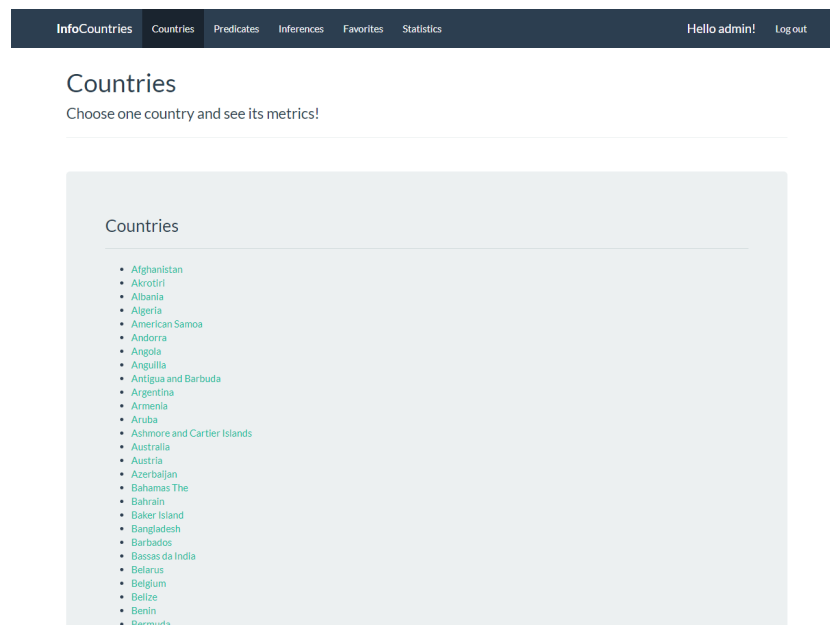


Figura 8.1: Página inicial com a lista de países.


Carregando no nome de um país temos acesso à página do país em questão (figura 8.2), onde podemos ver os seus atributos atualmente na base de dados, tal como o seu valor. Se o país existir na *Wikidata*, também se tem acesso à bandeira do país no final da lista. É possível adicionar o país aos favoritos do utilizador no botão "Add to favorites".

InfoCountries
Countries
Predicates
Inferences
Favorites
Statistics
Hello admin!
Log out

Portugal
Choose one metric to compare it with other countries

Country information

- Country - Portugal
- Area(sq.km) - 92371
- Birth rate(births/1000 population) - 10.82
- Death rate(deaths/1000 population) - 10.43
- Debt - external - 274700000000
- Electricity - consumption(kWh) - 42150000000
- Electricity - production(kWh) - 43280000000
- Exports - 37680000000
- GDP - 188700000000
- GDP - per capita - 17900
- GDP - real growth rate(%) - 1.10
- HIV/AIDS - adult prevalence rate(%) - 0.40
- Highways(km) - 17135
- Imports - 32100000000
- Infant mortality rate(deaths/1000 live births) - 5.05
- Inflation rate (consumer prices)(%) - 2.10
- Internet users - 360000
- Labor force - 5480000
- Life expectancy at birth(years) - 77.53
- Military expenditures - dollar figure - 3497800000
- Military expenditures - percent of GDP(%) - 2.30
- Natural gas - consumption(cu.m) - 2542000000
- Natural gas - exports(cu.m) - 0
- Natural gas - imports(cu.m) - 2553000000
- Natural gas - production(cu.m) - 0
- Oil - consumption(bbl/day) - 329800
- Oil - production(bbl/day) - 0
- Population - 10566212
- Telephones - main lines in use - 4278800
- Telephones - mobile cellular - 9341400
- Total fertility rate(children born/woman) - 1.47
- Current account balance - 8120000000
- Industrial production growth rate(%) - 1.10
- Internet hosts - 346078
- Investment (gross fixed)(% of GDP) - 22.30
- Oil - exports(bbl/day) - 28830
- Oil - imports(bbl/day) - 257300
- Railways(km) - 2050
- Reserves of foreign exchange & gold - 12300000000
- Unemployment rate(%) - 6.50
- HIV/AIDS - deaths - 1000
- HIV/AIDS - people living with HIV/AIDS - 22000
- Public debt(% of GDP) - 61.50



Country is not on favorites
Add to favorites
Gather data from Wikidata!
Try to get relations with other countries!

© 2017 - InfoCountries

Figura 8.2: Página de um país

Nesta página também é possível recolher atributos ou relações entre países da *Wikidata*. Para isso, é necessário carregar em "Gather data from Wikidata!" ou "Try to get relations with other countries!", respetivamente. A obtenção de dados da *Wikidata* pode levar algum tempo, tendo o utilizador que aguardar até que a página seja novamente mostrada sem o círculo de *loading*. Os resultados obtidos podem ser visualizados nas figuras 8.3 e 8.4.



Figura 8.3: Dados retirados da *Wikidata* relativamente ao país em questão.

Se carregarmos num predicado, temos acesso à sua página, onde podemos ver numa tabela os valores desse predicado para todos os países que o possuem (figura 8.6). É possível efetuar pesquisas por país ou por valor, e ainda ordenar os dados ou os países por ordem crescente ou decrescente. Podemos ver a lista completa de predicados se carregarmos em "Predicates" (figura 8.5) e aceder igualmente à informação associada a um predicado como descrito anteriormente.

Se existirem inferências reflexivas efetuadas, ou seja, inferências entre países, elas são mostradas na página "Inferences". É possível adicionar essas inferências efetuadas anteriormente de forma permanente à base de dados carregando no botão "Save inferences on database!" (figura 8.7).

A página "Favorites" apresenta uma lista de todos os países favoritos do utilizador para acesso rápido a esses países (figura 8.8).

A página "Statistics" apresenta três inferências efetuadas em *real-time*, tendo em conta os atributos dos vários países (figura 8.9). Apresenta ao utilizador uma lista ordenada dos melhores países para viver, melhores países para investir e países que geram maior tráfego de *Internet*.



Figura 8.4: Relações de um país com outros (diplomática e de fronteira).

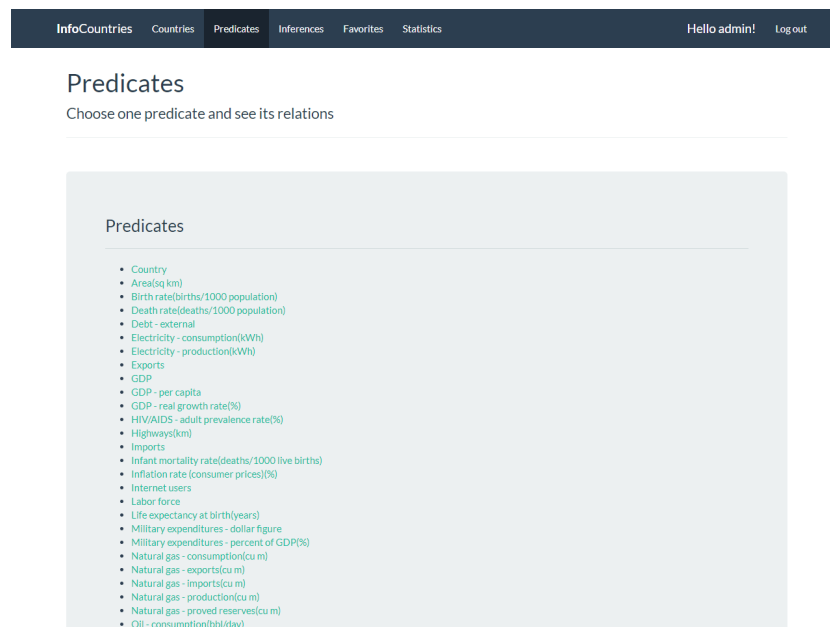


Figura 8.5: Predicados que se encontram na base de dados.

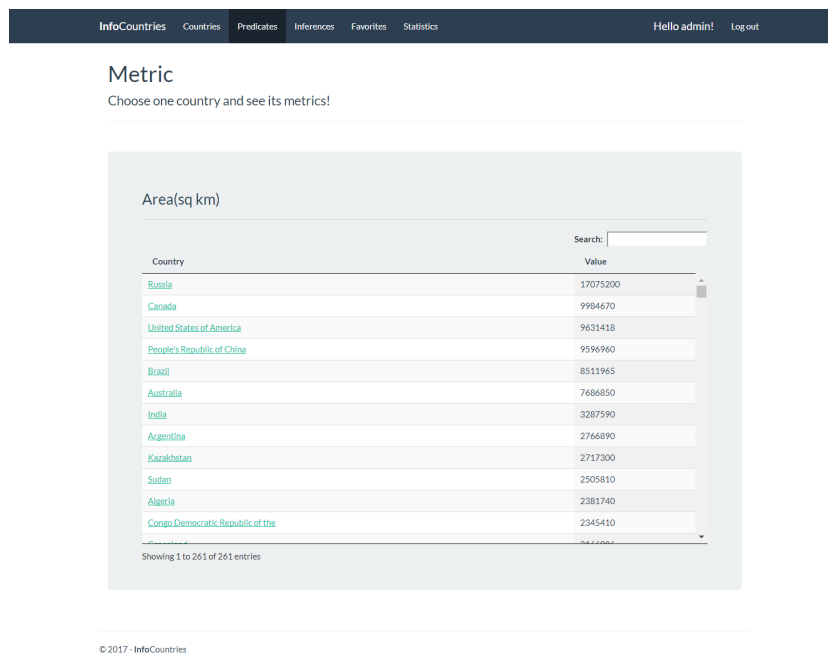


Figura 8.6: Relações estabelecidas com base no predicado escolhido, apresentando o respetivo sujeito e objeto em forma tabular.

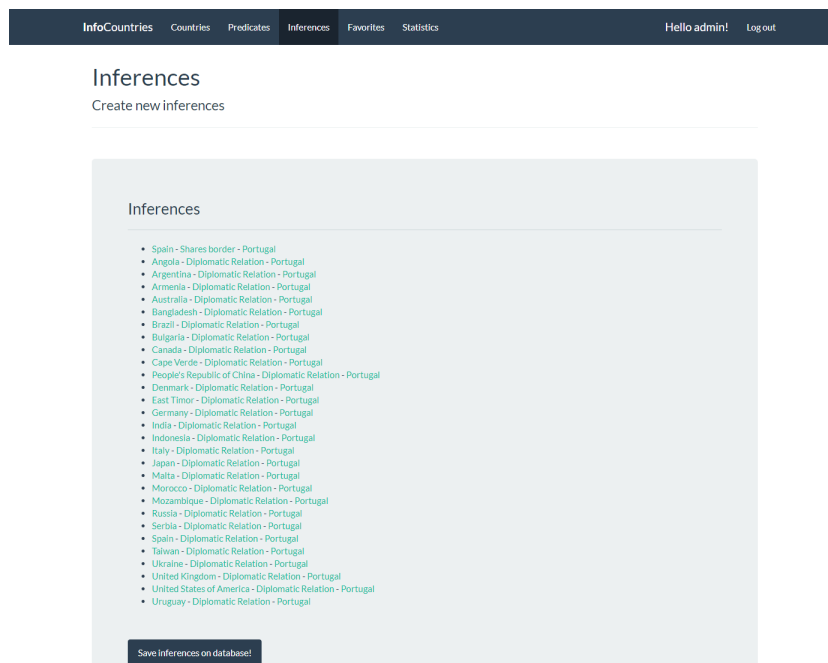


Figura 8.7: Informação recolhida após as inferências realizadas.

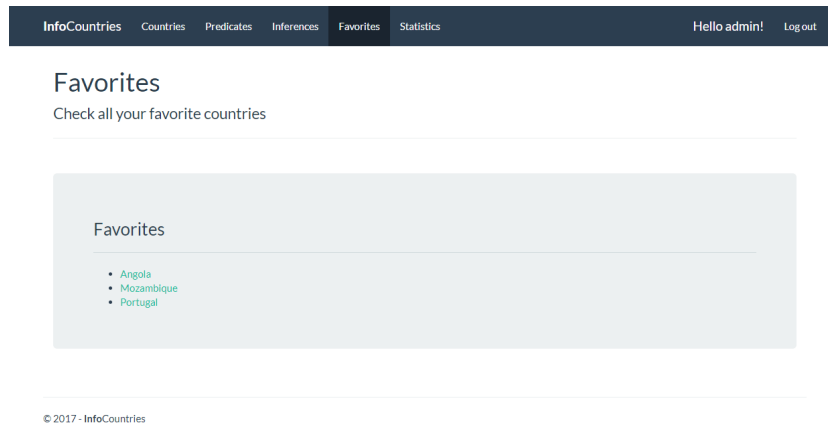


Figura 8.8: Países favoritos do utilizador.

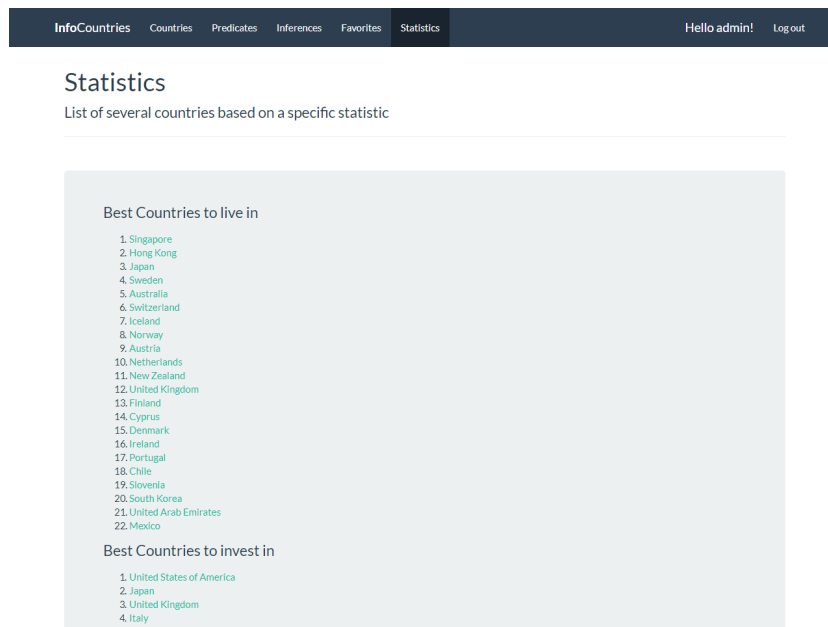


Figura 8.9: Estatísticas de países que envolvem inferências com base em determinados atributos para cada caso concreto.

Capítulo 9

Conclusões

Após a elaboração deste projeto, consideramos que atingimos com sucesso o objetivo de criar uma aplicação *web* integrada com web semântica, tornando-se mais fácil recolher dados sobre países e relações entre eles. Para além disso, conseguimos também incorporar dados da *Internet* em tempo-real vindos da *Wikidata* sobre cada país, tal como efetuar inferências sobre os dados armazenados.

O projeto pode ser desenvolvido para obter mais inferências, e inferências mais específicas, consoante o pretendido. No entanto, o tempo revelou-se escasso para construir e implementar mais inferências relevantes. É de notar que todas as relações, quer locais, quer retiradas da *Wikidata*, são bastante flexíveis para inserção de novos predicados caso seja necessário. Apesar de a interface não ser o foco principal deste projeto, conseguimos efetuar uma interface agradável para o utilizador e, acima de tudo, uma interface que lhe permite visualizar a informação armazenada e criar novas relações sem problemas, tal como a possibilidade de recolha desses mesmos dados por parte de outras máquinas.

Aquando da criação de novos tuplos com dados da *Wikidata* na página de um país, esta pode demorar algum tempo a atualizar. Isso deve-se à *library Wikidata 0.6.1*, que quando é pedido um conjunto de atributos, a pesquisa é efetuada um a um, ou seja, por cada atributo de um país, existe uma ligação à API REST da *Wikidata*, tornando o processo mais lento do que o desejável.

Podemos concluir que a utilização de *RDF* tem um propósito e poderá muito bem ser utilizada em aplicações web, facilitando a disposição de dados assim como a recolha de dados de cada página por terceiros. No caso da aplicação que desenvolvemos, é fácil recolher dados de determinados países ou de determinadas métricas apenas fazendo *crawl* sobre o *HTML* gerado pela nossa aplicação, facilitando a utilização dos dados dos países presentes na nossa aplicação por outras aplicações de terceiros.

Consideramos que utilização de *RDF* é útil em ambientes onde os dados e o seu esquema se podem alterar rapidamente. Neste projeto foi notado que, comparando com o *SQL*, onde o esquema não é alterável em *runtime*, o principal benefício deste modelo de dados semi-estruturados é a possibilidade de alteração do esquema da base de dados ao longo da execução de um processo, sem ser necessária uma reengenharia de processos.

Por outro lado, um dos problemas do *RDF* é o facto de não ser muito utilizado atualmente nos sistemas *web*, o que implica que a quantidade de dados disponíveis seja bastante limitada, para além de nem sempre serem os mais atualizados.

Capítulo 10

Configuração para executar a aplicação

O projeto foi desenvolvido no sistema operativo Windows 10 utilizando a linguagem de programação *Python* (3.6.2) e a framework *Django*(1.11.5). Para executar a aplicação são necessários os seguintes *packages* e as suas versões correspondentes:

- lxml==4.0.0
- Wikidata==0.6.1
- s4api==1.1.0
- requests==2.18.4

Para executar a aplicação, é necessário executar o servidor *GraphDB*, depois deve abrir a pasta do projeto (Projeto2) e executar o seguinte comando: `python3 manage.py runserver` (ou simplesmente fazer Run do projeto caso utilize o PyCharm). A aplicação fica acessível no *browser* via *URL* 127.0.0.1:8000 (equivalente a localhost:8000).

Se não existirem as bases de dados necessárias para a aplicação, serão automaticamente criadas e preenchidas com os dados presentes nos ficheiros de backup.

Existe inicialmente um utilizador registado na plataforma, mas podem ser adicionados mais utilizadores. Os tuplos (username, password) dos utilizadores atuais são (admin, rootroot).

Bibliografia

- [1] URL: <http://www.ua.pt/deti/uc/2380>.
- [2] URL: <https://perso.telecom-paristech.fr/eagan/class/igr204/datasets>.
- [3] URL: <https://www.w3.org/2012/pyRdfa/Validator.html>.