Aplicações na Web

Relatório do Projecto

Etapa 03

World Of Music

Grupo 08

Bioinformática e Biologia Computacional

sexta-feira, 3 de junho de 2016

Alunos:

- Ana Abrantes; 43071;
- Ana Pena; 40258;
- Luís Campos; 43134;

Modelo do Relatório de Projecto da Disciplina Aplicações na Web da Universidade de Lisboa.
Criado a 18 de Março de 2010 pelo Prof. Francisco M Couto. Actualizado a 24 de Fevereiro de 2016 pelo Prof. Tiago Guerreiro.

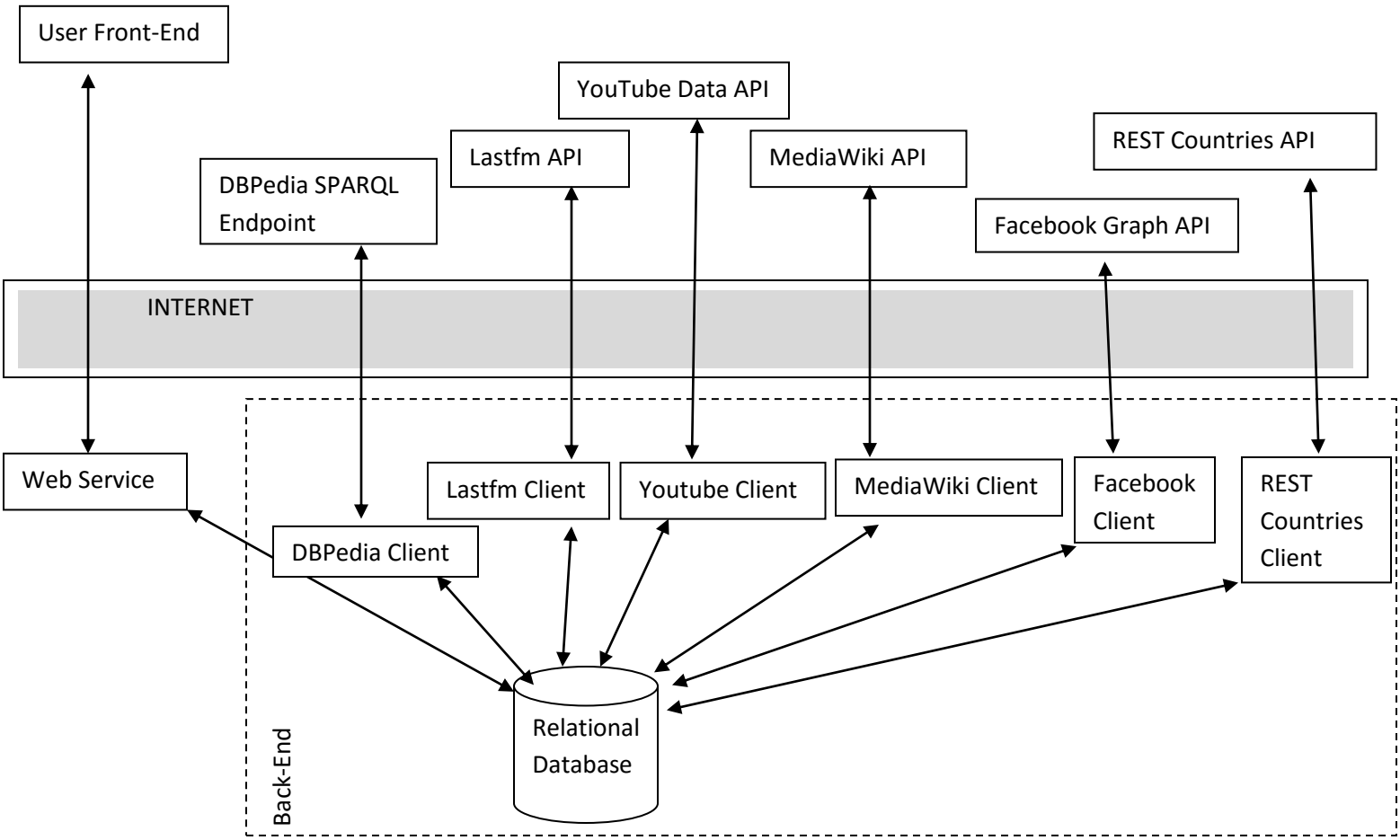
Índice

1. Introdução 3
2. Arquitectura 3
3. Back-end 4
   3.1. Tecnologias usadas 4
   3.2. Estrutura de Dados 5
   3.3. Fontes de Informação usadas e Componentes Implementadas 6
4. Web Service 9
   4.1. Tecnologias usadas e mais informação 9
   4.2. Lista de Métodos 9
   4.3. Exemplo 11
5. Front-end (admin e user) 12
   5.1. Tecnologias usadas 12
   5.2. RDFa 13
   5.3. Web Services usados 14
   5.4. Resumo da App 14
6. Discussão 19
   6.1. Problemas Encontrados 19
   6.1.1 - Problema 1 encontrado 19
   6.1.2 - Problema 2 encontrado 19
   6.1.3 - Problema 3 encontrado 20
   6.1.4 - Problema 4 encontrado 20
   6.1.5 - Problema 5 encontrado 20
   6.1.6 - Problema 6 encontrado 20
   6.1.7 - Problema 7 encontrado 20
   6.1.8 - Problema 8 encontrado ..... **Erro! Marcador não definido.**
   6.1.9 - Problema 9 encontrado 20
   6.2.Dicussão Geral 21

1. Introdução

Visionamos que esta seja uma app de música focada nas suas nacionalidades de forma a facilitar a divulgação de música de artistas cuja nacionalidade é um entrave à sua popularidade.

2. Arquitectura



3. Back-end

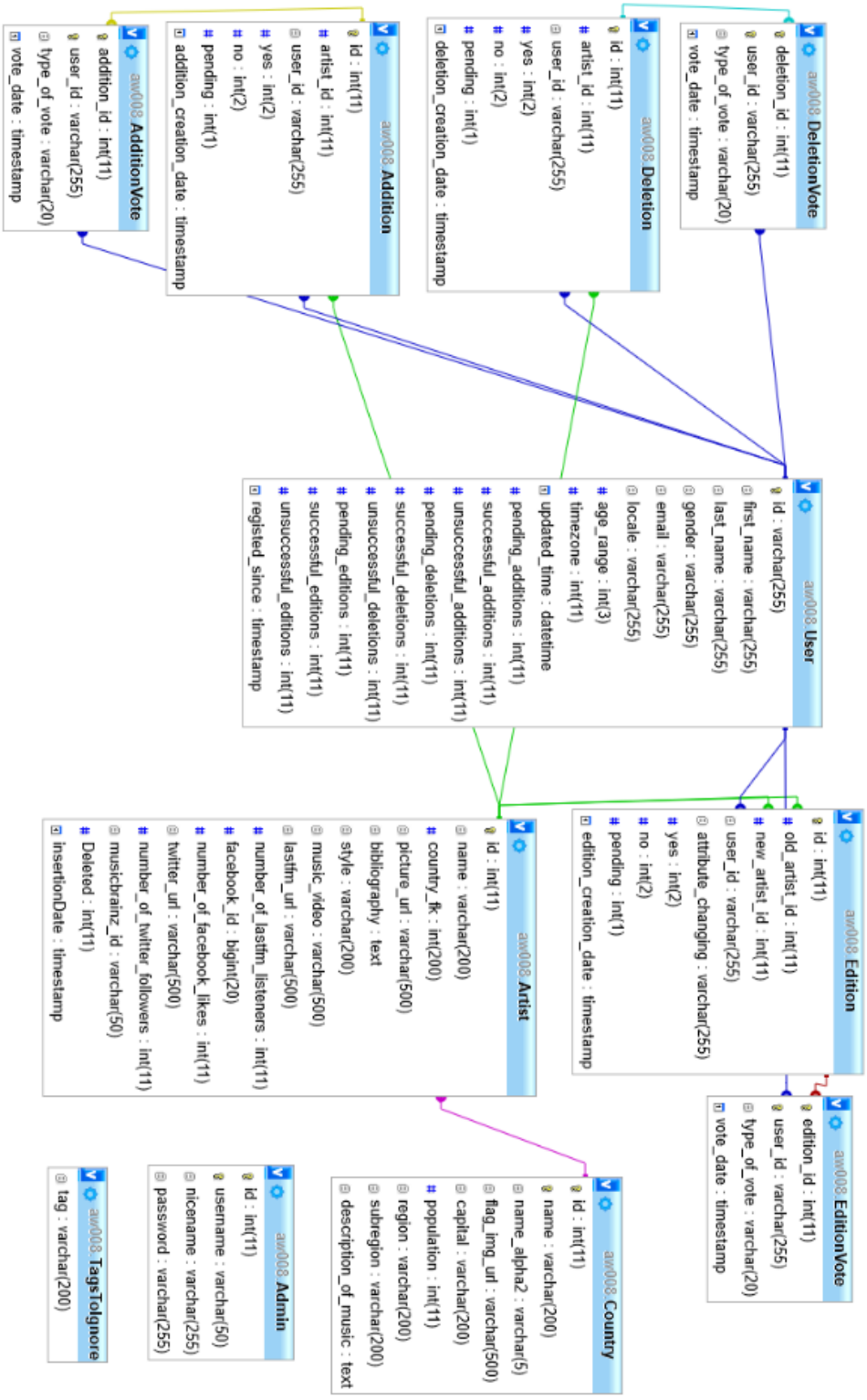
3.1. Tecnologias usadas

Utilizámos o PHP como linguagem de programação no back-end e MySQL como tecnologia de base de dados

Nome	URL	Descrição
PDO	https://secure.php.net/manual/en/intro.pdo.php	Utilizámos a extensão PDO do PHP para fazer ligações à nossa base de dados;
libcURL	https://secure.php.net/manual/en/book.curl.php	Utilizámos a biblioteca libcURL do PHP, quando achámos conveniente, para fazer ligação a servidores.
Facebook SDK v5 for PHP	https://developers.facebook.com/docs/reference/php	Biblioteca utilizada para fazer pedidos ao Facebook no back-end. Actualmente apenas o utilizamos para obter autenticação e ID dos utilizadores da nossa app.
PHP SPARQL Lib	https://github.com/cgutteridge/PHP-SPARQL-Lib	Para facilitar o envio de queries SPARQL.

Tabela 1 - Bibliotecas utilizadas no Back-end

3.2. Estrutura de Dados



Nome	Descrição
Artist	Dados sobre os artistas.
Country	Dados sobre os países.
TagsToIgnore	Lista de *tags* a ignorar no *Last.fm* quando se quer obter o estilo de um artista; neste momento esta tabela tem apenas os nomes dos países e seus gentílicos (isto para evitar que o artista tenha o estilo "portugal" ou "portuguese", por exemplo).
User	Guarda informação sobre o utilizador.
Addition **Deletion** **Edition**	Guarda informação sobre a tentativa de adição/deleção/edição de artistas, como por exemplo, que artista está a tentar ser modificado, quem fez o pedido para modificação e número de votos contra e a favor da modificação.
AdditionVote **DeletionVote** **EditionVote**	Regista as votações feitas pelos users quanto a tentações de adição/deleção/edição de artistas.
Admin	Tabela com informação de login na interface web dos administradores.

Tabela 2 - Descrição das Tabelas na Base de Dados Relacional que apoia a nossa app

3.3.Fontes de Informação usadas e Componentes Implementadas

Fonte 1 - Last.fm

Para obter **nomes de artistas** de cada país utilizámos o método tag.getTopArtists da API do Last.fm, procurando pela *tag* correspondente ao nome do país. Por exemplo, para procurar artistas portugueses, pedimos uma lista de artistas que tivessem a *tag* "Portugal". Para obter tais dados fizemos pedidos GET a URLs análogos a este:

http://ws.audioscrobbler.com/2.0/?method=tag.getTopArtists&limit=**20**tag=**Portugal**&format=**json**

Para obter **informação sobre cada artista** (nomeadamente: URL de foto, bibliografia, URL do Last.fm, número de *listeners* no Last.fm e ID no Musicbrainz) utilizámos o método artist.getInfo.

Fonte 2 - Facebook

Para obter de uma forma automáticas os **IDs das páginas de Facebook dos artistas** na nossa base de dados, utilizámos a funcionalidade "search" do Graph API do Facebook. Por exemplo, se fizermos um pedido ao seguinte URL

https://graph.facebook.com/search?q=**Capicua**&type=**page**

obtemos como resultado um JSON com uma lista de IDs de páginas que surgem numa procura por "Capicua". Decidimos adoptar a heurística de que o primeiro resultado deverá ser o correcto a maior parte das vezes, e foi o ID do primeiro resultado que guardámos na nossa base de dados como sendo o ID da página do Facebook do artista.

Para obter o **número de likes** da página do Facebook dos artistas, também utilizámos o Graph API, fazendo chamadas GET com o seguinte formato:

https://graph.facebook.com/**{id}**?fields=**likes**

Sendo {id} o ID da página do artista.

Fonte 3 - REST Countries

Para obter informações sobre os países existentes no Mundo foi utilizada a API REST Countries. Esta disponibiliza várias informações, tendo sido recolhidas apenas nome, nome abreviado (em ISO 3166-2), capital, região e sub-região, dos 247 países disponibilizados. Estes dados foram posteriormente guardados na nossa base de dados.

Fonte 4 - Wikipedia

Para **obter nomes dos países e seu gentílicos para preencher a tabela TagsToIgnore** realizou-se *scrapping* da página, utilizando **XPath**:

https://en.wikipedia.org/wiki/List_of_adjectival_and_demonymic_forms_for_countries_and_nations.

Foi utilizado o plug-in **XPath Checker** (https://code.google.com/archive/p/xpathchecker/) do Mozilla Firefox para se encontrar a melhor expressão a utilizar, que foi a seguinte: "//td/a".

Fonte 5 – MediaWiki

Para obter a descrição da música de cada país foi usada a API MediaWiki que permitiu retirar a introdução das páginas do Wikipedia que continham este assunto. Para cada país guardado na nossa base de dados foi feito o seguinte pedido à API (em php):

https://en.wikipedia.org/w/api.php?format=json&action=query&prop=extracts&exintro=&explaintext=&titles=Music_of_" . [nome_pais]

O texto obtido foi depois guardado na nossa base para facilidade de acesso em fases posteriores. Alguns países tem o seu conteúdo musical descrito como “There isn't available information about music in this country.”, devido ao problema 2 referido mais à frente.

Fonte 6 – Youtube

De modo a obter um vídeo representativo de cada artista utilizámos a Youtube Data API (v3), mais concretamente o método *search*. Para cada artista foram usados dois grupos de termos:

- O nome do artista, retirado diretamente da nossa base de dados;
- A sua música mais popular segundo o Last.fm, tendo sido usado para tal o método *artist.getTopTracks,* que recebia também como input o nome do artista usado acima.

Para cada artista foi então obtido um ID de um vídeo do Youtube, ID esse que foi guardado na nossa base de dados de modo a ser utilizado de modo mais fácil e rápido no nosso site.

Fonte 7 – DBPedia

Para obter mais nomes de artistas e de melhor qualidade (ver ponto 6.1.1 deste relatório), fizemos *queries* SPARQL ao SPARQL endpoint da DBPedia. Com os nomes obtidos, fomos buscar mais informações sobre o artista ao Last.fm, como descrito em cima.

Fizemos vários pedidos utilizando o seguindo esqueleto de *query* ($country é o nome do país do qual queremos artistas e $n é o máximo de artistas que queremos receber de volta):

```
SELECT DISTINCT ?thing, ?name
        WHERE {?thing rdfs:label ?name .
        FILTER (lang(?name) = 'en')
        FILTER (EXISTS {?thing dbo:hometown dbr:$country} ||
            EXISTS {?thing dbp:origin dbr:$country} ||
            EXISTS {?thing dbp:origin ?origin .
                ?origin dbo:country dbr:$country} ||
            EXISTS {?thing dbo:hometown ?hometown .
                ?hometown dbo:country dbr:$country} ||
            EXISTS {?thing dbo:birthPlace dbr:$country})
        FILTER (EXISTS {?thing a schema:MusicGroup} ||
            EXISTS {?thing a dbo:MusicalArtist})
        }
        LIMIT $n":
```

A tradução para linguagem humana desta *query* é algo do género: “Devolve $n nomes diferentes de entidades que tenham um nome (querendo esse esse nome em inglês), que sejam do país $country e que sejam grupos ou artistas musicais.” Há várias maneiras de dizer que um artista/grupo é de um certo país, dai esta *query* ser mais longa do que seria se houvesse uma melhor standardização.

4. Web Service

4.1. Tecnologias usadas e mais informação

Construímos o nosso Web Service segundo o estilo REST. As respostas do serviço vêm por defeito no formato JSON. Se o cliente incluir “text/xml” no Accept do header do pedido, a resposta é devolvida em formato XML. Esta não deve ser a melhor de maneira de verificar o que o cliente quer, mas depois de alguma pesquisa, não foi encontrada alternativa melhor. O Web Service também permite JSONP: no caso de o pedido não ser XML, se o cliente incluir um parâmetro “callback” no URL do pedido, é devolvido um JSON englobado no valor que foi dado nesse parâmetro. Por exemplo, se for incluído no URL “callback=function1” é devolvido “function1({resposta_JSON})”.

O Web Service devolve “HTTP status codes” adequados para cada caso. Como exemplos, o serviço um “200 OK” quando o utilizador pede informação do país com o código “pt”, um “404 Not Found” se o cliente pedir informação sobre o país com o código “ap” (que não existe) e um “403 Forbidden” se o utilizador estiver a tentar adicionar um artista sem estar autenticado.

Há casos em que houve dúvida em relação ao código a usar. Por exemplo, que *status code* devolver quando um cliente envia um pedido PUT tentando editar o URL do Facebook de um artista, enviando um URL que não é do Facebook? Colocámos o serviço a devolver “400 Bad Request” mas a verdade é que não há nenhuma “malformed syntax” (http://www.restapitutorial.com/httpstatuscodes.html).

Foi tentado que quando o utilizador faz um pedido com algum problema a mensagem de erro seja útil para o utilizador perceber o que está mal (embora não garantamos que isto aconteça sempre). Por exemplo, se o cliente pedir informação sobre um país que não existe, a mensagem devolvida é “Country does not exist”.

4.2. Lista de Métodos

Documentação: http://appserver.di.fc.ul.pt/~aw008/webservices/doc/

Os pedidos que envolvem os métodos mais sensíveis (POST, PUT e DELETE) requerem autenticação na *app* para serem usados. Isto envolve entrar no URL da app e fazer login com conta do Facebook.

No entanto, criámos um código especial para ser usado por nós, para fins de teste, e pelos professores para fins de avaliação. Este código, 746gftsho, é incluindo no url do pedido, como valor do parâmetro “access_token”.

A título de exemplo:

DELETE /artist/Boss AC?access_token=746gftsho

#	Recurso	Método	Resultado
1	country	GET	Lista dos países
2	country/{code}	GET	Informação sobre o país
3	country/{name}/artists	GET	Lista de artistas desse país
4	artist/{name}	POST	Faz pedido de criação de um novo artista
5	artist/{name}	GET	Informação sobre o artista.
6	artist/{name}	PUT	Actualiza dados do artista.
7	artist/{name}	DELETE	Faz pedido de deleção de um artista
8	pending_additon	GET	Recebe lista de *pending additions*
9	pending_additon/{id}	GET	Recebe informação sobre a *pending addition*
10	pending_additon/{id}/{type_of_vote}	POST	Vota numa pending *addition*
11	pending_deletion	GET	Recebe lista de *pending deletions*
12	pending_deletion/{id}	GET	Recebe informação sobre a *pending deletion*
13	pending_deletion/{id}/{type_of_vote}	POST	Vota numa pending *deletion*
14	pending_edition	GET	Recebe lista de *pending editions*
15	pending_edition/{id}	GET	Recebe informação sobre a *pending edition*
16	pending_edition/{id}/{type_of_vote}	POST	Vota numa pending *edition*
17	user/{id}	GET	Informação sobre o utilizador
18	user/{id}/votes	GET	Votações feitas por esse utilizador

Tabela 3 – Recursos do Web Service do World Of Music

Para mais informação sobre estes métodos e sobre parâmetros que podem ser usados nos métodos, ver documentação.

Todos os métodos GET envolvem apenas fazer uma query SELECT à base de dados e devolver, num formato adequado, os resultados ou parte destes.

Quando um cliente faz um pedido POST ao recurso artist/{name}, é criado um novo artista, mas este não fica de imediato exposto ao público (por exemplo, através do recurso GET country/{name}/artists). Este artista fica escondido até aprovação por outros utilizadores. Isto porque quando se faz um pedido POST neste recurso é também criada uma "pending-addition", que consiste num registo da tentativa de adição desse artista e do número de votos contra e a favor a adição do artista. Quando o número de votos positivos chega a um certo valor, o artista deixa de ficar "escondido". Se, pelo contrário, o número de votos negativos chegar a um certo valor, essa "pending-addition" deixa de estar válida e o artista não fica exposto.

As votações podem ser feitas através de pedidos POST ao método pending_additon/{id}/{type_of_vote}, sendo {id} o ID da “pending-addition” e {type_of_vote} o valor “positive_vote” ou “negative_vote”.

Algo análogo acontece para as deleções de artistas (pedidos DELETE ao recurso artist/{name}) e edição de atributos de artistas (pedidos PUT ao recurso artist/{name}), mas neste caso é criada uma “pending-deletion” ou “pending-edition”, respectivamente.

4.3.Exemplo

Se fizermos um pedido GET ao URL:

http://appserver.di.fc.ul.pt/~aw008/webservices/artist.php/Capicua

Obtemos o seguinte texto no formato JSON (bibliografia não aparece toda neste exemplo):


```
{
    "name":"Capicua",
    "style":"hip-hop",
    "country":"Portugal",
    "picture_url":"http:\/\/img2-ak.lst.fm\/i\/u\/174s\/4a7fbbf4749645cda4025c1deb829273.png",
    "Last.fm_url":"http:\/\/www.last.fm\/music\/Capicua",
    "number_of_Last.fm_listeners":"3852",
    "bibliography":"About CAPICUA (…)",
    "music_video":null,
    "facebook_id":"272101826169708",
    "number_of_facebook_likes":null,
    "twitter_url":null,
    "number_of_twitter_followers":null,
    "musicbrainz_id":"451107dc-7d40-4bd4-86e6-f76e566ff17b"
}
```


Para ver mais exemplos, aceder à documentação.

5. Front-end (admin e user)

5.1. Tecnologias usadas

HTML, CSS e Javascript. Utilizamos técnicas AJAX.

Nome	URL	Descrição
jQuery	https://jquery.com/	Para facilitar o desenvolvimento em Javascript.
Bootstrap	https://getbootstrap.com/	Utilizado para facilitar o desenvolvimento do design do front-end.
Facebook SDK for JavaScript	https://developers.facebook.com/docs/reference/php	Utilizado para fazer pedidos ao Facebook. Mais especificamente, utilizamos esta ferramenta para implementar a autenticação dos utilizadores e para obter informação sobre estes.
jVectorMap	http://jvectormap.com/	Utilizado para fazer os mapas no front-end.

Tabela 4 - Bibliotecas utilizadas no Front-End

5.2. RDFa

```
@prefix dbp: <http://dbpedia.org/property/> .
@prefix ns1: <http://schema.org/> .
@prefix ns2: <http://www.w3.org/1999/xhtml/vocab#> .
@prefix ns3: <http://www.w3.org/ns/rdfa#> .

<> ns3:usesVocabulary ns1: .

<#about_modal> ns2:role ns2:dialog .
<#edition_modal> ns2:role ns2:dialog .
<#edition_modal_form> ns2:role ns2:form .
<#feedback_modal> ns2:role ns2:dialog .
<#feedback_modal_vote_buttons> ns2:role ns2:group .
<#warning_modal> ns2:role ns2:dialog .

[] a ns1:MusicGroup;

   ns1:description """Formed in Amadora, Lisbon, Portugal, in 2006, (...)""";

   ns1:foundingLocation "Portugal";

   ns1:genre "Kuduro";

   ns1:image <http://img2-ak.lst.fm/i/u/174s/458416acccfe4d079c7d5fc34a9e4b76.png>;

   ns1:name "Buraka Som Sistema";

   ns1:video [ a ns1:VideoObject ] .


[] a ns1:Country;

   dbp:capital "Lisbon";

   dbp:populationTotal "10374822";

   ns1:name "Portugal";

   ns1:region "Europe";

   ns1:subregion "Southern Europe" .
```

5.3.Web Services usados

O Front-End utiliza apenas o Web Service por nós implementado.

Explicamos quais os recursos utilizados e como na secção seguinte.

5.4. Resumo da App

Vídeo de Demonstração: https://www.youtube.com/watch?v=GCvFzA9SgCI

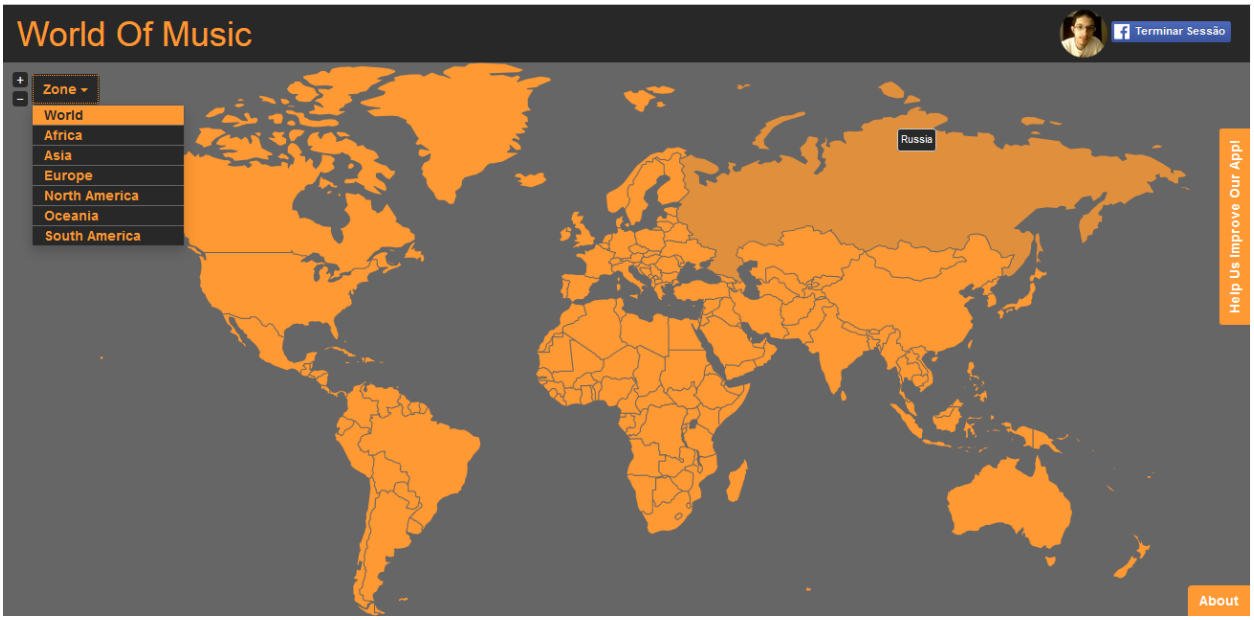


Figura 1

De momento a nossa app pode ser acedida através do endereço http://appserver.di.fc.ul.pt/~aw008/. Inicialmente é apresentada uma simples página na qual se destaca um mapa onde é possível explorar os vários países do mundo (Figura 1). Caso o user carregue num dos países a app é atualizada de forma assíncrona, aparecendo agora uma secção com informação sobre o país escolhido. Esta informação é obtida fazendo um pedido AJAX GET ao recurso country/{código do país}.

O utilizador também pode premir no botão presente do lado direito do cabeçalho da app para se autenticar na app através da conta do Facebook e assim poder votar em mudanças na app, adicionar e apagar artistas e editar atributos de artistas.

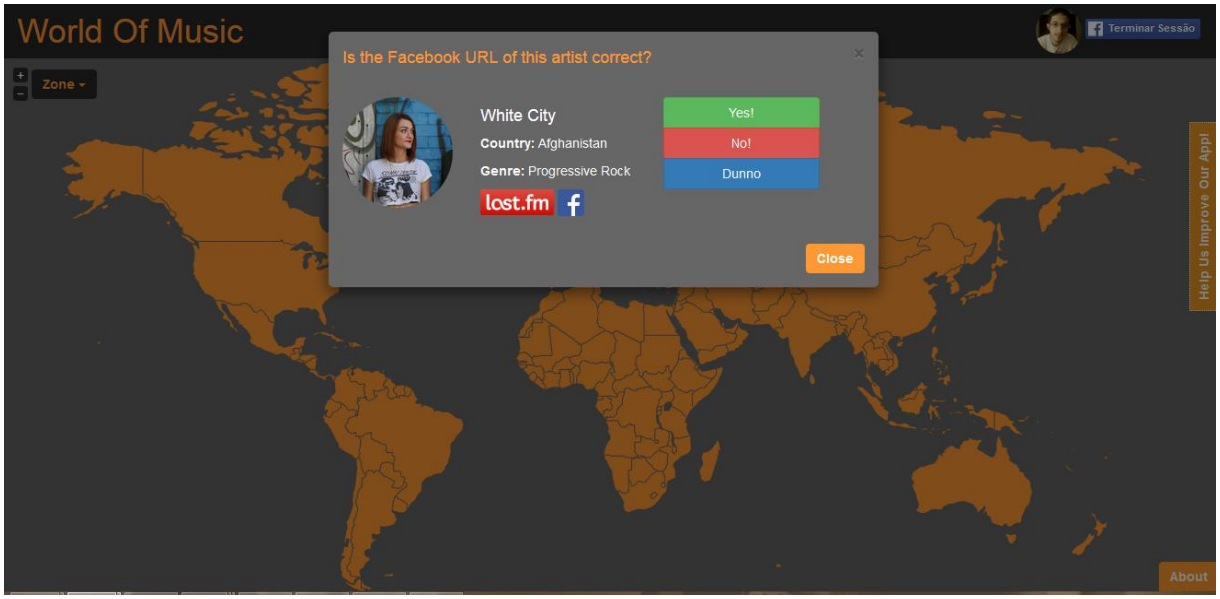


Figura 2

Em qualquer sítio da app encontra-se "preso" ao lado direito do ecrã um botão laranja que, quando pressionado leva ao aparecimento de um "modal" (Figura 2) onde o utilizador pode dar a sua opinião sobre a adição, deleção e edição de atributos de artistas, sugeridas por outros utilizadores. Esta funcionalidade de o utilizador poder dar feedback está assente em vários pedidos GET a recursos que contêm no URL "pending_addition", "pending_deletion" ou "pending_edition".

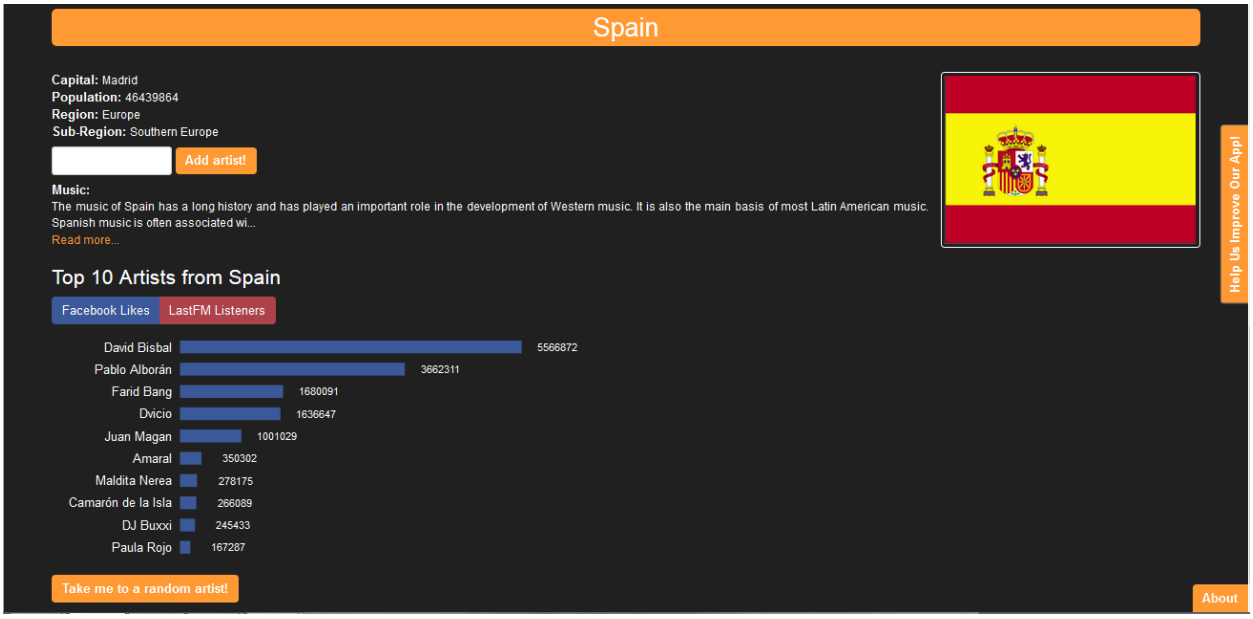


Figura 2

Na secção do país (Figura 3), para além de informação sobre o país e sobre a música do país, existe um botão "Take me to a random artist!" que quando pressionado leva ao aparecimento de uma secção com informação sobre um artista aleatório desse país, sendo que o nome do artista aleatório é obtido através do recurso GET country/{código do país}/artists.

Também foi implementado um "Top 10 de Artistas" do país, segundo o número de "likes" do Facebok ou "listeners" do Lastfm, utilizando a biblioteca D3.js (no código que se encontra no seguinte URL: http://bl.ocks.org/erikvullings/51cc5332439939f1f292). As listas de artistas são obtidas aquando a geração da secção do país através de pedidos ao recurso GET country/{código do país}/artists, ficando estas listas guardadas num objeto Javascript, para que a mudança do tipo de top (por likes ou listeners) seja mais rápida e agradável para o utilizador.

Para além disto, nesta secção dos países o utlizador pode sugerir a adição de um novo artista, preenchendo o formulário e premindo o botão "Add artist!", sendo posteriormente feito um pedido POST ao recurso artist/{nome do artista}.

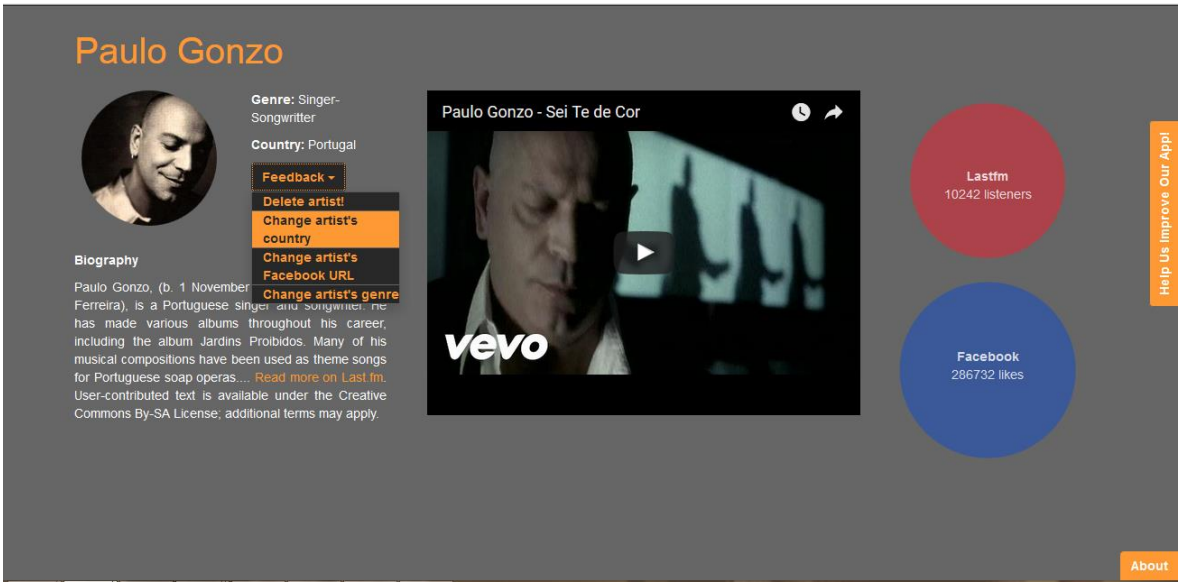


Figura 3

Na secção do artista (Figura 4) é apresentada informação sobre o artista, obtida através de um pedido GET ao recurso artist/{nome do artista}. Nesta página é apresentado o nome do artista, o género musical, o país de origem, uma parte de uma biografia do artista e um vídeo musical. Também são apresentados dois círculos que representam o número do likes/listeners do Facebook/Lastfm. O tamanho do círculo é logaritmicamente proporcional ao número de likes/listeners, sendo que o tamanho máximo corresponde ao número máximo de likes/listeners de artistas desse país. Assim, dá para ter uma ideia visual da popularidade do artista no país onde pertence. Estes ciclos são gerados apenas quando o utilizador carrega em algum botão que leve ao aparecimento da secção do artista. E está dependente de alguns pedidos GET aos recursos artist/{nome do artista} e GET country/{código do país}/artists.

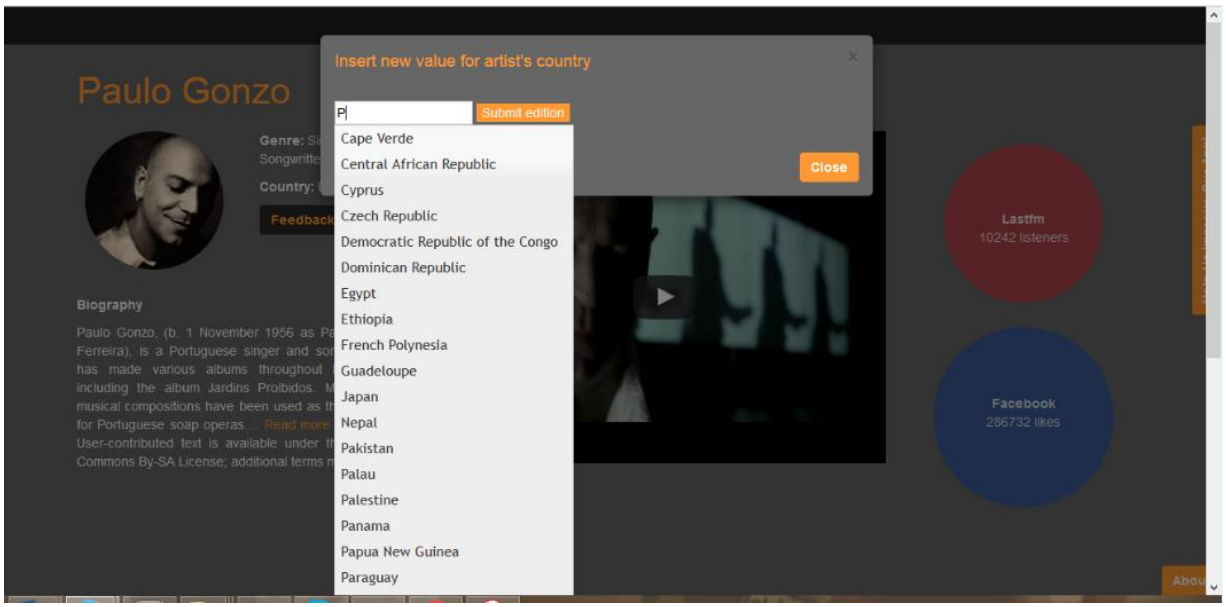


Figura 4

Ao lado da foto do artista está um botão que quando premido leva ao aparecimento de um *dropdown* onde o utilizador pode escolher entre vários tipos de feedback. O utilizador pode sugerir que o artista desapareça da app escolhendo a opção "Delete artist!", que ao ser clicada faz um pedido DELETE ao recurso artist/{nome do artista}. Ou pode escolher editar algum atributo do artista carregando numa das outras opções, que leva ao aparecimento de um "modal" com uma *form* que o user deve preencher para submeter as suas sugestões. Isto leva a uma chamada PUT ao recurso artist/{nome do artista}. Caso o user tenha escolhido sugerir uma alteração do país do artista, quando o user começar a escrever na caixa de texto, vai aparecer uma lista dos países possíveis para escolha. Esta lista é obtida através do resultado dum pedido GET ao recurso country.

Figura 5

Tentámos que a app fosse responsiva em relação a diferentes tamanhos de janelas, de forma a ficar com um bom visual em qualquer dispositivo fixo ou móvel. Na Figura 6, podemos ver um exemplo de como a parte da app que está na Figura 1 apareceria num ecrã com menores dimensões. Testámos a app em 3 browsers: Mozilla Firefox 46.0.1, Google Chrome 50 e Internet Explorer 11.

6. Discussão

6.1.Problemas Encontrados

6.1.1 - Problema 1 encontrado

Como referido no ponto 4.2, Fonte 1, utilizámos o método tag.getTopArtists da API do Last.fm para obter nomes de artistas com uma certa tag corresponde a um país, por exemplo, "Portugal". O problema é que ao contrário do que intuitivamente esperávamos, os resultados não vêm ordenados por popularidade no Last.fm, mas sim por *tag count,* o que presumo que corresponda ao número de users que deram a esse artista a tag em questão. Em consequência disto, os nomes de artistas que recebemos como output não são os mais populares, às vezes longe disso, havendo alguns com pouquíssimos *listeners* no Last.fm.

Possíveis soluções:

1. Simplesmente pedir informação sobre mais artistas através deste método. Eventualmente também vamos receber os nomes dos mais populares.

2. Em alternativa ao Last.fm, usar o MusicBrainz para obter lista de nomes de artistas de uma certa nacionalidade.

Para resolver este problema, não utilizámos nenhumas das opções inicialmente propostas. Em vez disso, utilizámos a DBPedia para obter nomes de artistas de uma certa nacionalidade. A lógica que seguimos para tomar esta decisão foi que se um artista está na Wikipedia é porque é relativamente popular.

6.1.2 - Problema 2 encontrado

Como foi descrito no ponto 4.2 fonte 6, não foi possível recolher informação sobre a música de alguns países. Cada país tem uma página no Wikipedia, em inglês, com um formato comum: https://en.wikipedia.org/wiki /Music_of_{nome_pais} .

Deparámo-nos com alguns países sem página própria (por exemplo as Ilhas Åland) ou cujo nome do país dado no URL da página era diferente do guardado da nossa base de dados (por exemplo a República do Congo que temos registada como "Republic of the Congo" e no Wikipedia tem a página https://en.wikipedia.org/wiki/Music_of_the_Republic_of_the_Congo)

Possíveis Soluções:

1. Introduzir manualmente a informação sobre a música destes países, recolhidas quer do Wikipedia ou de outras fontes.

2. Utilizar uma fonte alternativa para recolha de informação musical de países que complete estas entradas automaticamente.

Uma vez que não foi possível encontrar um mecanismo automático para recolha desta informação foi utilizada a primeira solução. Para alguns países, principalmente pequenas ilhas oceânicas, não foi encontrada informação relevante sobre a sua música, pelo que estes países mantiveram o texto original ("There isn't available information about music in this country.")

6.1.3 - Problema 3 encontrado

Como referido no ponto 4.1, o nosso Web Service devolve XML se o cliente incluir "text/xml" no Accept do header do pedido. Mas "*there's a catch*". Se o cliente colocar *apenas* "text/xml" no Accept, o servidor devolve um erro com o código "406". Não conseguimos resolver o erro. No entanto, a mensagem de erro já não aparece se no Accept estiver "text/html; text/xml".

6.1.4 - Problema 4 encontrado

No Internet Explorer 11, a *scrollbar* sobrepõe-se em parte do botão lateral, o que pode dificultar o clique neste. Isto não acontece nos outros dois browsers testados. Não sabemos como resolver, mas também não considerámos este problema uma prioridade.

6.1.5 - Problema 5 encontrado

Quando aparece o modal para o user dar feedback sobre adições/deleções/edições de artistas, o tempo de loading da informação pode ser longo, e é previsto que seja mais longo quando mais sugestões de alterações forem submetidas na app. Não houve reflexão sobre como resolver o problema, mas também não considerámos uma prioridade.

6.1.6 - Problema 6 encontrado

O carregamento de cada uma das componentes da app pode ser um pouco lenta e pouco bonita. Por exemplo, quando se vai para a área do artista existem uns momentos em que não há vídeo do Youtube nem gráficos D3. No Chrome até acontece algo bizarro: enquanto o vídeo do Youtube não carrega, no local onde este deveria aparecer, aparece em vez disso uma versão da app em miniatura.

Opções para resolver/melhorar o problema:
- Olhar para o código de Javascript e tentar melhorá-lo de forma a tornar as coisas mais rápidas;
- Só fazer aparecer uma secção quando todos os elementos da secção tiverem carregado. Isto levaria o mesmo tempo, mas seria menos feio do que ver os elementos a apareceram aos poucos.

Não implementámos nenhuma destas opções.

6.1.7 - Problema 7 encontrado

O Web Service tens uns quantos problemas não-graves que não foram resolvidos, como por exemplo, o user poder sugerir uma edição de um atributo do artista, sugerindo um valor desse atributo igual ao atual.

6.1.8 - Problema 8 encontrado

Este não é um problema tecnológico, mas artístico: nenhum dos membros do grupo tem especial talento para design pelo que o aspeto do site está aquém do ideal.

6.2. Discussão Geral

A maioria dos objetivos do projeto, como trabalho da cadeira de Aplicações na Web, foram cumpridos, tendo ficado por implementar uma interface web para os administradores da app (foi desenvolvida uma versão muito rudimentar, mas com nenhuma funcionalidade especialmente útil).

Uma eventual continuação do desenvolvimento da app poderia envolver:

- ✓ a correção dos problemas indicados na secção anterior;
- ✓ tornar a *user experience* mais interessante, de forma a cumprir o objetivo de esta ser uma ferramenta para divulgar artistas menos conhecidos de outros países;
- ✓ a interface do site poderia ser também melhorada, de modo a ser mais interessante e estética para os utilizadores.