

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Spotify-ed: Music Recommendation and Discovery in Spotify

José Lage Bateira



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Fabien Gouyon

Co-Supervisor: Matthew Davies

June 23, 2014



# **Spotify-ed: Music Recommendation and Discovery in Spotify**

**José Lage Bateira**

Mestrado Integrado em Engenharia Informática e Computação

June 23, 2014





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	1
1.2	Motivation and Goals . . . . .	1
1.3	Project . . . . .	2
1.4	Dissertation Structure . . . . .	3
<b>2</b>	<b>State of The Art</b>	<b>5</b>
2.1	Introduction . . . . .	5
2.2	Related and Similar Services . . . . .	5
2.2.1	Liveplasma - liveplasma.com . . . . .	5
2.2.2	Tuneglue - audiomap.tuneglue.net . . . . .	7
2.2.3	MusicRoamer - musicroamer.com . . . . .	8
2.3	Conclusions . . . . .	9
<b>3</b>	<b>Context and Methodologies</b>	<b>13</b>
3.1	Introducing Spotify . . . . .	14
3.1.1	Development Tools . . . . .	14
3.1.2	Experimentações Feitas . . . . .	20
3.1.3	Conclusão . . . . .	21
3.2	Technologies used . . . . .	21
3.2.1	<i>Spotify Desktop Client</i> . . . . .	21
3.2.2	Webkit Development Tools - webkit.org . . . . .	25
3.2.3	Npmjs - npmjs.org . . . . .	25
3.2.4	Gruntjs - gruntjs.com . . . . .	28
3.2.5	Arborjs - arborjs.org . . . . .	28
3.3	Conclusions . . . . .	28
<b>4</b>	<b>Plano de Trabalho</b>	<b>29</b>
4.1	Fases do Projeto . . . . .	29
4.1.1	Fase 1 - Desenho da Aplicação . . . . .	29
4.1.2	Fase 2 - Mapeamento de Metadados Spotify em Last.fm . . . . .	30
4.1.3	Fase 3 - Criação e Edição do grafo . . . . .	30
4.1.4	Fase 4 - Reprodução de Música . . . . .	31
4.1.5	Fase 5 - Avaliação e Validação . . . . .	31
4.2	Calendarização . . . . .	32
4.3	Resumo . . . . .	32
<b>5</b>	<b>Conclusões</b>	<b>35</b>

## CONTENTS

# List of Figures

2.1	liveplasma: search result for "Amália Rodrigues"; upper left corner: artist albums; lower left corner: youtube's <i>mini-player</i> . . . . .	6
2.2	liveplasma: interface to start playing tracks. <i>Similar</i> button plays tracks from similar artists, whereas, the <i>only</i> button only plays tracks from the specified artist. . . . .	6
2.3	Tuneglue: menu que aparece ao clicar num nó. . . . .	7
2.4	Tuneglue: grafo depois do primeiro nó ser expandido. . . . .	8
2.5	MusicRoamer: Search options. by artist; by keyword and by Last.fm username . . . . .	9
2.6	MusicRoamer: Visual representation of the artist graph . . . . .	10
2.7	MusicRoamer: Personalizable parameters for the graph . . . . .	11
2.8	MusicRoamer: The graph after expanding one node . . . . .	12
3.1	Spotify: desktop client's discovery mode interface. . . . .	15
3.2	Spotify: Last.fm's Spotify Application opened. . . . .	16
3.3	Spotify: <i>Play Button</i> . . . . .	17
3.4	Spotify: <i>Follow Button</i> Allows the user to follow the music artist. . . . .	17
3.5	Experiência com <i>Metadata API</i> e <i>Play Button Widget</i> (código fonte: <a href="https://github.com/carsy/spotify-playground">github.com/carsy/spotify-playground</a> ) . . . . .	22
3.6	<i>Website</i> do RAMA embebido numa Aplicação Spotify . . . . .	23
3.7	Resultado do teste do elemento <i>canvas</i> . . . . .	23
3.8	Menu <i>Develop</i> . . . . .	24
3.9	Webkit: Vista da tab <i>Inspector</i> . Outras ferramentas disponíveis (tabs): <i>Resources</i> , <i>Network</i> , <i>Sources</i> , <i>Timeline</i> , <i>Profiles</i> , <i>Audits</i> e <i>Console</i> . . . . .	25
3.10	Webkit Network . . . . .	26
3.11	Webkit Profile: É possível ver que a renderização do grafo é o que ocupa mais tempo de processamento como esperado. No entanto, existe uma parte de <i>JQuery</i> que ocupa 12.72% do tempo de processamento, o que pode indicar um possível ponto de melhoria de performance. . . . .	26
3.12	Webkit Audit: 96% do código <i>CSS</i> não está a ser usado, sendo por isso, um ponto de melhoria reduzir a quantidade de informação descarregada. . . . .	27
3.13	Webkit Console: Erros de <i>Javascript</i> aparecem destacados para chamar a atenção. . . . .	27
4.1	Calendarização do Plano de Trabalho . . . . .	33



## LIST OF FIGURES

## Chapter 3

# Context and Methodologies

The primal objective of this dissertation, as referred in chapter 1, is to develop one or more software modules that will improve Spotify Users' music discovery and recommendation experience using visual tools to represent the music artists' relations and Spotify's streaming service to provide high quality music stream.

The initial proposal was to develop a module that implements, at least, one of the following features:

1. Integrate Spotify's music stream into RAMA's website
2. Integrate information from the Spotify user into RAMA
3. Improve RAMA's features and design
4. Integrate the RAMA concept into a Spotify Application
5. Integrate RAMA's playlist generation into a Spotify Application
6. Integrate some of the above mentioned modules into a Mobile Application

The first three functionalities (1, 2 and 3) focus on improving RAMA using Spotify's API, i.e. to integrate Spotify into RAMA. Whereas 4 and 5 aim to integrate RAMA's concept into Spotify, through a Spotify Application (it would work as a plugin to Spotify's Desktop Client). The last one (6) would focus on implementing the previous functionalities into an Android, iOS or Windows Phone Application.

This chapter aims to analyse every single drawback of each possibility that affects the choice of which modules do develop, and on which environments it fits better: Spotify Application, Mobile Application, or RAMA improvements.

At first, Spotify's development environment will be introduced 3.1 in order to assess which tools are available for developers. Next, the available tools will be evaluated in order to determine which ones fit the proposed modules to be developed, mostly, through experiments.

By the end of this chapter the modules to be developed should be clearly stated, as well as which tools will be used in the prototype.

The prototype should pursue the objective of contributing to an improved user experience when discovering new music taking advantage of visual tools that implement RAMA's concept.

### 3.1 Introducing Spotify

Spotify is a Music Streaming Service that allows, through an Internet connection, the user to listen to any track (if available in the user's country) in Spotify's catalogue. The service was launched in 2008 with a native desktop client application.

Now, the service has several types of clients available to the users: desktop client, webplayer and mobile applications.

**Desktop Client** Desktop version of Spotify, with Windows and Mac versions (and also a Linux preview version).

**Webplayer** Web version of Spotify. This was released in 2013, and spotify still advises the use of the native applications for a better user experience.

**Mobile Applications** The mobile applications are available for Android and iOS devices.

#### 3.1.1 Development Tools

Spotify provides a set of tools<sup>1</sup> to develop third-party applications (websites, native applications and mobile applications) and Spotify Applications (that run inside Spotify's Desktop Client). There are five tools, each with different purposes.

##### 3.1.1.1 Spotify Apps

Spotify Applications<sup>2</sup> are a special case in the whole set of tools provided by Spotify. These applications are designed to run *inside* the Desktop Client. Hence, its development is also inside the same environment.

Spotify users can run and install applications from the store called "App Finder". All the applications are free.

In 3.1 on can see the interface of the desktop client. In this case, the discovery mode's interface.

On the left side, in the menu, below the "App Finder" item, appears all the applications the user has installed from the store.

In 3.2 the official Last.fm application is opened. Note how the space filled by the applications are always the same.

---

<sup>1</sup><http://developer.spotify.com/technologies>

<sup>2</sup><https://developer.spotify.com/technologies/apps>

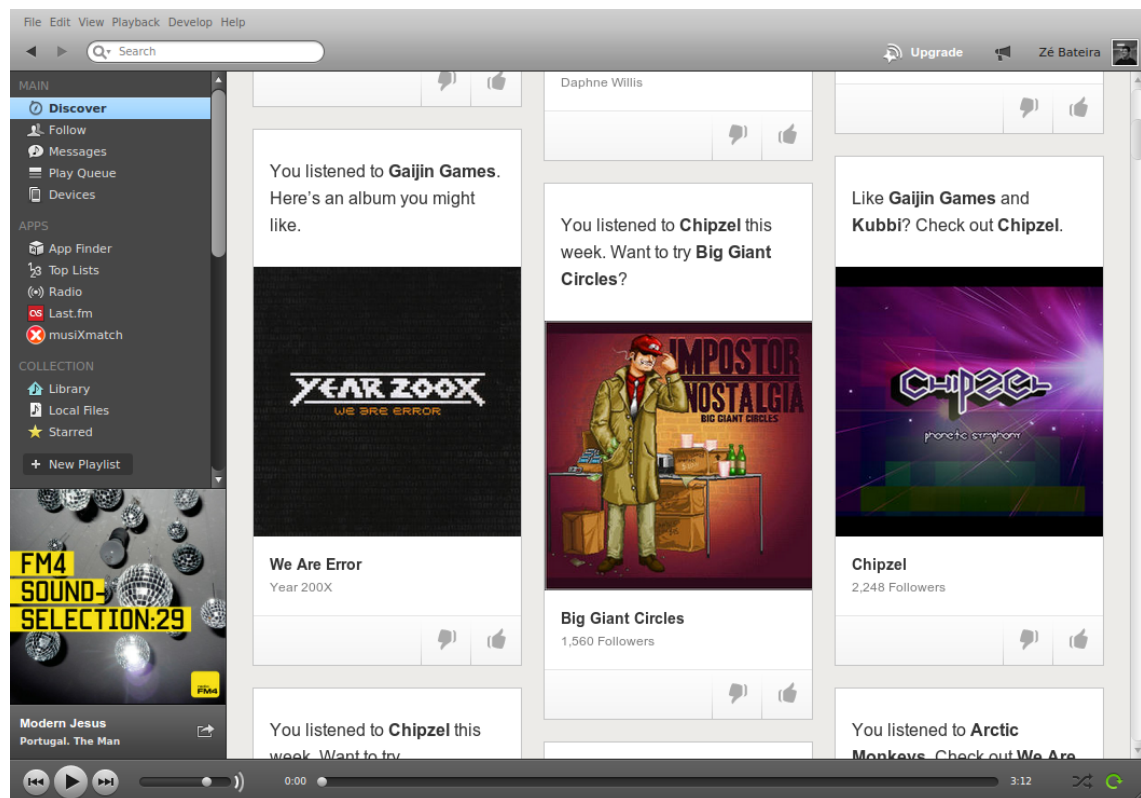


Figure 3.1: Spotify: desktop client's discovery mode interface.

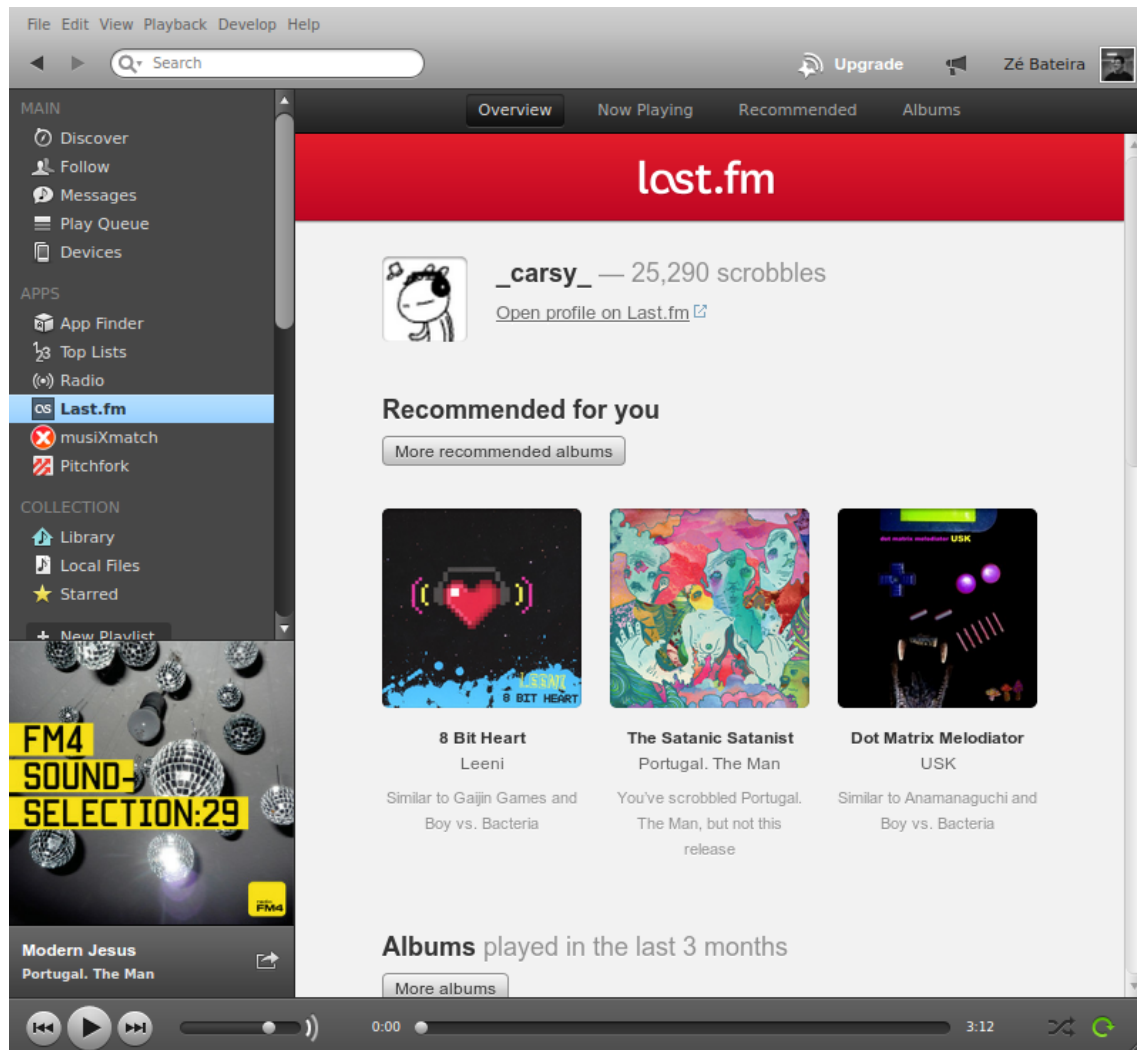
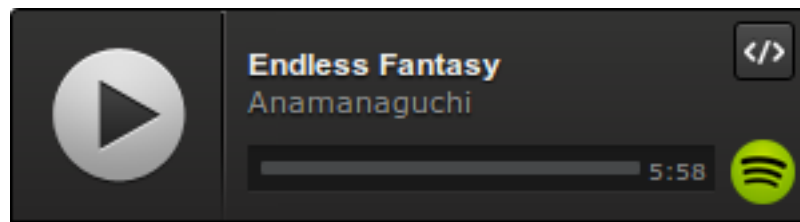


Figure 3.2: Spotify: Last.fm's Spotify Application opened.

Figure 3.3: Spotify: *Play Button*.

The Applications' runtime environment is one of a browser-based. More specifically, powered by the Chromium Embedded Framework<sup>3</sup>. This means that the code to develop a Spotify Application follows the same principles as a web application: HTML, CSS and Javascript.

Spotify developed two Frameworks<sup>4</sup> to help developers create these applications: the API 1.x Framework<sup>5</sup> and the Views Framework<sup>6</sup>.

The first one provides an interface to use object models, access metadata, control the player, among others. The second offers support for web components like buttons, lists, tabs, among others.

In order to develop the proposed modules 4 and 5, these are the most appropriate tools.

### 3.1.1.2 Spotify Widgets

Spotify Widgets<sup>7</sup> are small web components that can be embedded in external websites. Spotify provides two components: *Play Button* (3.3) and a *Follow Button* (3.4)

However, there are some limitations. In Spotify, only logged in users can use the service (listen to tracks, etc). This also applies to these widgets - Even if they are in an external application, only Spotify users can interact with them.

This limitation does make sense in the case of the *Follow Button*, but the *Play Button* becomes useless to non-spotify users.

<sup>3</sup><https://code.google.com/p/chromiumembedded>

<sup>4</sup><https://developer.spotify.com/technologies/apps/reference>

<sup>5</sup><https://developer.spotify.com/docs/apps/api/1.0/>

<sup>6</sup><https://developer.spotify.com/docs/apps/views/1.0/>

<sup>7</sup><https://developer.spotify.com/technologies/widgets>

Figure 3.4: Spotify: *Follow Button* Allows the user to follow the music artist.

In truth, these widgets are nothing but an hyperlink to a Spotify Client (Web Player or Desktop). With the Play Button, the stream of tracks always played inside Spotify's environment, and not on external applications.

To embed a widget, it is only required to copy-paste Html code into the website, where appropriate:

```

1 <iframe src="https://embed.spotify.com/?uri=spotify:track:1
   EsdqTsiQPauJ82iy7KfS1"
2     frameborder="0"
3     width="300"
4     height="380">
5 </iframe>

```

Listing 3.1: Html code to embed the *Play Button*

These widgets are useful to develop the proposed modules 1 and 3.

### 3.1.1.3 Libspotify SDK

Libspotify SDK<sup>8</sup> é uma API que permite adicionar os serviços do Spotify em aplicações externas. No entanto, existem algumas limitações para os utilizadores destas aplicações.

Existem, dois tipos de conta a que o utilizador pode subscrever: conta grátis e conta *premium*. Como foi referido anteriormente (3.1.1.2) apenas utilizadores Spotify podem interagir com qualquer componente do Spotify, dentro ou fora das aplicações nativas do mesmo. Libspotify fornece uma interface que permite a um utilizador fazer *login* no Spotify em aplicações externas por forma a poder ouvir música do Spotify, criar playlists e outras funcionalidades. No entanto, os únicos utilizadores que pode fazer *login* nestas aplicações que usam Libspotify, são utilizadores *premium*. Para além de que, os *developers* da própria aplicação também precisam de ser utilizadores *premium*.

Neste sentido, uma aplicação que, para funcionar, necessita de que o utilizador, para além de possuir uma conta Spotify, também pague uma subscrição mensal *premium*, é uma aplicação bastante restritiva.

Esta ferramenta pode ser usada para desenvolver os módulos 1, 2 e 6.

### 3.1.1.4 Metadata API

A *Metadata API*<sup>9</sup> disponibiliza publicamente informação de músicas, álbuns e artistas da Base de dados do Spotify.

Através de pedidos HTTP é possível obter informação da base de dados do Spotify. Existe dois tipos de pedidos que esta API disponibiliza: *search*<sup>10</sup> e *lookup*<sup>11</sup>. Para obter informação detalhada

<sup>8</sup><https://developer.spotify.com/technologies/libspotify>

<sup>9</sup><https://developer.spotify.com/technologies/web-api>

<sup>10</sup><https://developer.spotify.com/technologies/web-api/search>

<sup>11</sup><https://developer.spotify.com/technologies/web-api/lookup>

de, por exemplo, um artista, é necessário saber o seu identificador único. Esse identificador é um *URI* da forma:

`spotify:artist:<artist_id>`, onde *artist\_id* é um identificador único.

Exemplo:

`spotify:artist:65nZq8l5VZRG4X445F5kmN`, é o identificador único da fadista "Mariza".

Também existem identificadores únicos para álbuns:

`spotify:album:5d1LpIPmTTrvPltx26T1EU` (álbum "Fado Tradicional" de "Mariza")

e para faixas de música:

`spotify:track:2vqYasauhDLVjTt7CGWK6y` (música "Fado Vianinha" do mesmo álbum)

Para obter este *URI* é preciso interrogar a base de dados com um método de pesquisa. Para isso, usa-se o *search*.

### **Search**

O *URL* base de utilização é:

`http://ws.spotify.com/search/1/album`, para pesquisa de álbuns.

Se se pretender pesquisar Artistas, usa-se *artist*, se se pretender pesquisar Faixas de música, usa-se *track*.

Exemplos:

`http://ws.spotify.com/search/1/album?q=foo`

`http://ws.spotify.com/search/1/artist.json?q=red+hot`

O resultado da *query*, por defeito, tem o formato *XML*. No entanto, também se pode especificar o formato *JSON* (como no segundo exemplo).

Dada a *query*:

`http://ws.spotify.com/search/1/artist.json?q=camane` (fadista "Camané")

Obtém-se o resultado:

```
1 {
2   "info": {
3     "num_results": 2,
4     "limit": 100,
5     "offset": 0,
6     "query": "camane",
7     "type": "artist",
8     "page": 1
9   },
```



```

10     "artists": [
11         {
12             "href": "spotify:artist:3MLPFTe4BrpEV2e0VG0gLK",
13             "name": "Camane",
14             "popularity": "0.27"
15         },
16         {
17             "href": "spotify:artist:5Gwulm1LfURW7dbZD1V3zX",
18             "name": "Sergio Godinho/Camane/Carlos Do Carmo",
19             "popularity": "0"
20         }
21     ]
22 }

```

Listing 3.2: Os resultados são ordenados pelo atributo "popularity"

### Lookup

Depois de obtido o *URI* identificador, é possível obter mais informações de um conteúdo usando o *lookup*.

A seguinte *query*:

<http://ws.spotify.com/lookup/1/.json?uri=spotify:artist:3MLPFTe4BrpEV2e0VG0gLK>

Retorna:

```

1 {
2     "info": {
3         "type": "artist"
4     },
5     "artist": {
6         "href": "spotify:artist:3MLPFTe4BrpEV2e0VG0gLK",
7         "name": "Camane"
8     }
9 }

```

Listing 3.3: Resultado do *lookup* do fadista "Camané"

Esta API seria bastante útil para desenvolver qualquer um dos seis módulos propostos. Aliás, até complementa as *Widgets* e o *Libspotify SDK*.

As of the writing of this report, Spotify has a set of tools

### 3.1.2 Experimentações Feitas

Numa primeira experiência com as ferramentas, foi criado um pequeno *website* que permite pesquisar e ouvir Música do Spotify usando a *Metadata API* e *Spotify Widgets*:

<http://carsy.github.io/spotify-playground>

Na figura 3.5 é possível ver o resultado de uma pesquisa, e a *Widget Play Button* com o resultado selecionado da pesquisa.

Verificou-se que as duas ferramentas estão bem documentadas e em constante atualização.

Outra experiência foi realizada para verificar se é possível usar o elemento *canvas* numa Aplicação Spotify. Isto é necessário pois será a única forma de poder desenhar graficamente o grafo. Para isso foi apenas necessário criar uma aplicação com o seguinte código fonte:

```
1 <iframe src="http://rama.inescporto.pt/app" frameborder="0"></iframe>
```

Listing 3.4: Elemento *iframe* que embebe o *website* do RAMA na aplicação

Desta forma, é possível embeber o RAMA na Aplicação Spotify (que usa o elemento *canvas* para desenhar o grafo). Resultado final na figura 3.6.

Apesar de *iframes* serem suportadas, existem outros componentes que não o são. A aplicação não é usável, pois não permite, por exemplo, reproduzir automaticamente faixas de artistas.

No entanto existe uma forma de testar quais os elementos de *HTML5* suportados, usando uma aplicação interna do Spotify. Na figura 3.7 é possível ver que o elemento *canvas* é suportado a cem por cento.

### 3.1.3 Conclusão

A prova de conceito desenvolvida (3.6) demonstrou-se a mais indicada para o objetivo final de criar um ambiente integrado entre o Spotify e o RAMA.

Assim, os módulos a serem desenvolvidos são 4 e 5.

## 3.2 Technologies used

As seguintes tecnologias serão utilizadas nas fase de desenvolvimento, testes e otimização da Aplicação Spotify.

### 3.2.1 Spotify Desktop Client

O desenvolvimento de Aplicações Spotify é feito de forma integrada no programa.

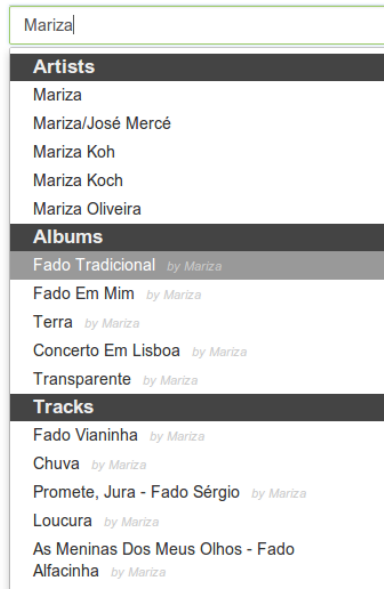
Para abrir uma Aplicação Spotify, localmente, escreve-se o seguinte na barra de pesquisa: spotify:app:rama

Onde *rama* deve ser o identificador da aplicação declarado no ficheiro *manifest.json*<sup>12</sup>.

Exemplo de ficheiro *manifest.json*:

<sup>12</sup>ficheiro situado na *root* da pasta do projeto

### Spotify Playground



(a) Resultado da pesquisa "Mariza"

### Spotify Playground



(b) Depois de selecionado o álbum "Fado Tradicional" aparece o *Play button* com as faixas do álbum.

Figure 3.5: Experiência com *Metadata API* e *Play Button Widget* (código fonte: [github.com/carsy/spotify-playground](https://github.com/carsy/spotify-playground))

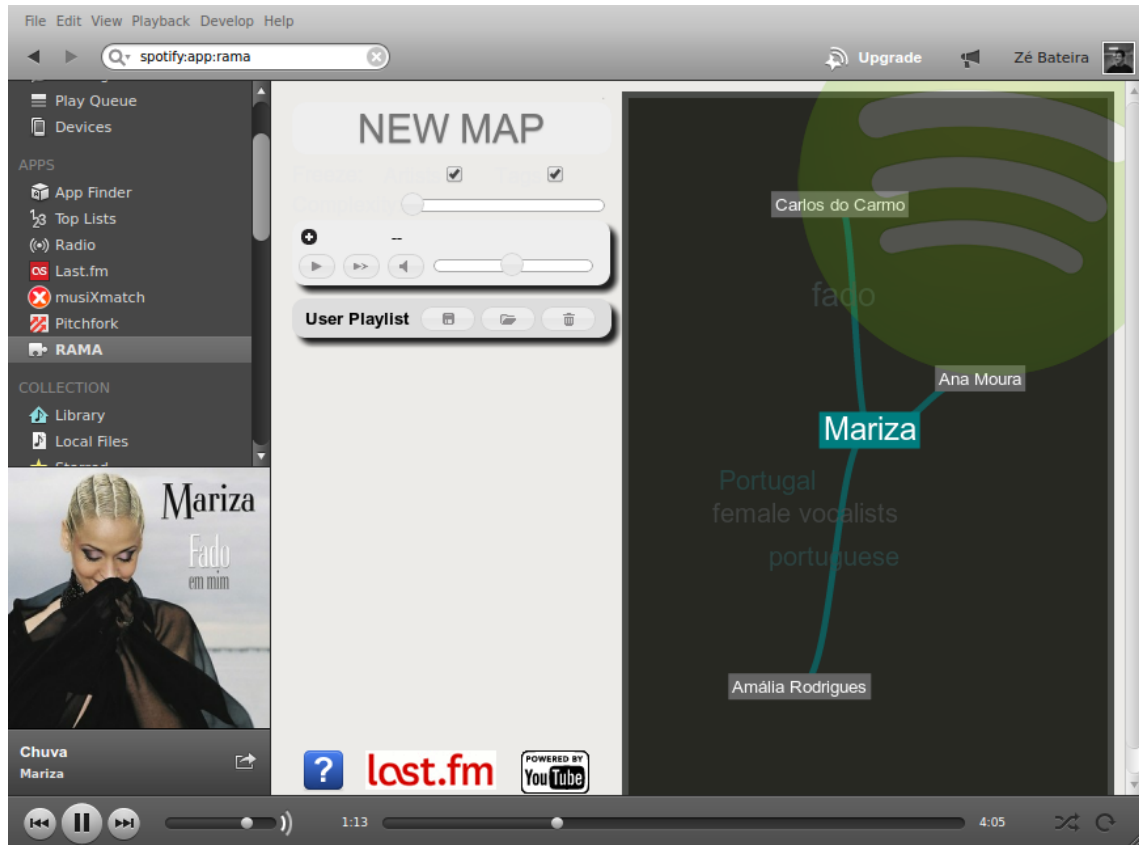


Figure 3.6: Website do RAMA embebido numa Aplicação Spotify

Canvas	20
canvas element <a href="#">?</a>	Yes <a href="#">?</a>
2D context <a href="#">?</a>	Yes <a href="#">?</a>
Text <a href="#">?</a>	Yes <a href="#">?</a>

Figure 3.7: Resultado do teste do elemento *canvas*

## Context and Methodologies

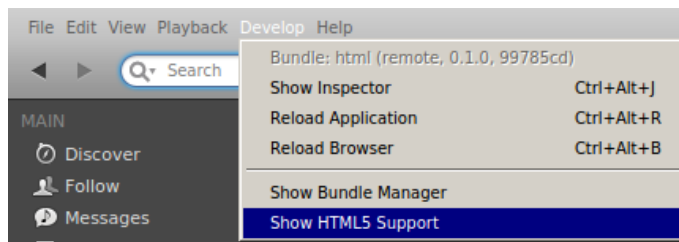


Figure 3.8: Menu *Develop*

```
1 {
2   "AppName": {
3     "en": "RAMA"
4   },
5   "BundleIdentifier": "rama",
6   "AppDescription": {
7     "en": "RAMA: Relational Artist MApps"
8   },
9   "AcceptedLinkTypes": [
10    "playlist"
11  ],
12  "BundleType": "Application",
13  "BundleVersion": "0.2",
14  "DefaultTabs": [
15    {
16      "arguments": "index",
17      "title": {
18        "en": "Home"
19      }
20    }
21  ],
22  "Dependencies": {
23    "api": "1.10.2",
24    "views": "1.18.1"
25  },
26  "SupportedDeviceClasses": ["Desktop"],
27  "SupportedLanguages": [
28    "en"
29  ],
30  "VendorIdentifier": "pt.inescporto"
31 }
```

Listing 3.5: manifest.json: *BundleIdentifier* é o identificador da aplicação; *Dependencies* declara as dependências das API's necessárias ao desenvolvimento.

Existem outras opções úteis a que se pode aceder usando a tab *Develop* (3.8). A opção "Show Inspector" abre a janela *Webkit Development Tools* (3.2.2)

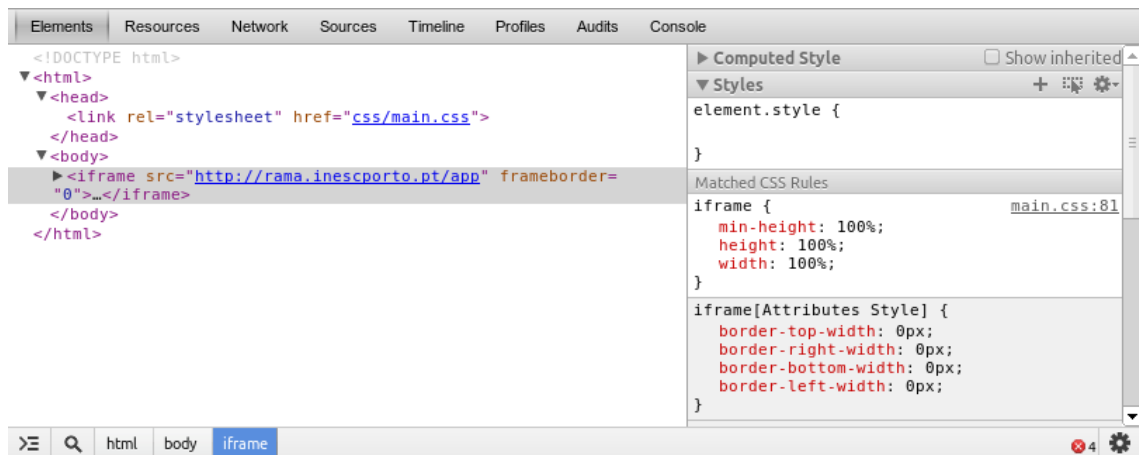


Figure 3.9: Webkit: Vista da tab *Inspector*. Outras ferramentas disponíveis (tabs): *Resources*, *Network*, *Sources*, *Timeline*, *Profiles*, *Audits* e *Console*.

### 3.2.2 Webkit Development Tools - webkit.org

A partir do *webkit*, tem-se acesso a várias ferramentas úteis para o desenvolvimento *web* (3.9).

A mais importantes são:

**Inspector** Permite inspecionar e editar o código *HTML* e *CSS* da aplicação diretamente (3.9).

**Network** Permite, por exemplo, ver o tempo que cada componente da aplicação demorou a carregar (uma imagem ou um ficheiro *css*) (3.10).

**Profile** Permite identificar que partes do código *javascript* são as mais frequentemente executadas (3.11).

**Audit** Ajuda a perceber quantos recursos estão a ser descarregados desnecessariamente, como por exemplo, regras de *CSS* que não estão a ser usadas (3.12).

**Console** Muito útil para *debug* de *javascript*.



### 3.2.3 Npmjs - npmjs.org

Gestor de pacotes de software e dependências. Para usar *npm* é necessário um ficheiro de configuração *package.json* que permite identificar quais os pacotes de que a aplicação depende, e as suas versões.

Exemplo:

```
1 {
2   "name": "RAMA",
3   "devDependencies": {
4     "grunt": "~0.4.2",
```

## Context and Methodologies

Elements		Resources		Network		Sources		Timeline		Profiles		Audits		Console	
Name	Path	Meth...	Status	Type	Initiator	Size	Content	Time	Latency	Timeline					
 <b>spotify-logo.png</b>	/img	GET	200 OK	imag...	Other	16.5 KB	16.3 KB	Pend...		-1477934 ms	-2216901 ms	-2955869 ms	-3694836 ms		
 <b>main.css</b>	/css	GET	200 OK	text/css	Other	1.6 KB	1.5 KB	Pend...							

2 requests | 18.1 KB transferred

≡ 🔍 📑 ● 🚫 All Documents Stylesheets Images Scripts XHR Fonts WebSockets Other ✖ 4 ⚙️

Figure 3.10: Webkit Network

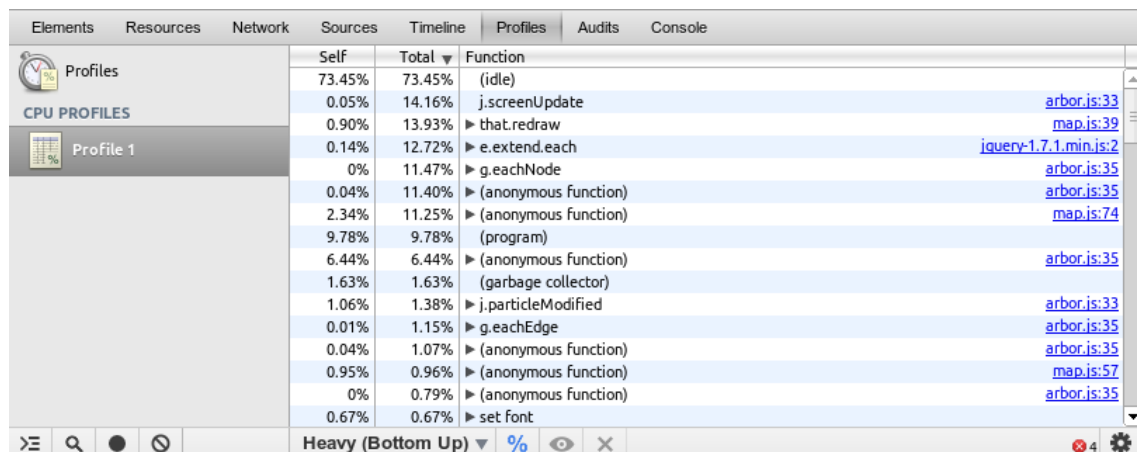










Elements	Resources	Network	Sources	Timeline	Profiles	Audits	Console
 <b>Profiles</b>  <b>CPU PROFILES</b>   <b>Profile 1</b>			Self	Total ▼	Function		
			73.45%	73.45%	(idle)		
			0.05%	14.16%	j.screenUpdate	<a href="#">arbor.js:33</a>	
			0.90%	13.93%	▶ that.redraw	<a href="#">map.js:39</a>	
			0.14%	12.72%	▶ e.extend.each	<a href="#">jquery-1.7.1.min.js:2</a>	
			0%	11.47%	▶ g.eachNode	<a href="#">arbor.js:35</a>	
			0.04%	11.40%	▶ (anonymous function)	<a href="#">arbor.js:35</a>	
			2.34%	11.25%	▶ (anonymous function)	<a href="#">map.js:74</a>	
			9.78%	9.78%	(program)		
			6.44%	6.44%	▶ (anonymous function)	<a href="#">arbor.js:35</a>	
			1.63%	1.63%	(garbage collector)		
			1.06%	1.38%	▶ j.particleModified	<a href="#">arbor.js:33</a>	
			0.01%	1.15%	▶ g.eachEdge	<a href="#">arbor.js:35</a>	
			0.04%	1.07%	▶ (anonymous function)	<a href="#">arbor.js:35</a>	
			0.95%	0.96%	▶ (anonymous function)	<a href="#">map.js:57</a>	
			0%	0.79%	▶ (anonymous function)	<a href="#">arbor.js:35</a>	
			0.67%	0.67%	▶ set font		
   			<b>Heavy (Bottom Up) ▼</b>   				 

Figure 3.11: Webkit Profile: É possível ver que a renderização do grafo é o que ocupa mais tempo de processamento como esperado. No entanto, existe uma parte de *jQuery* que ocupa 12.72% do tempo de processamento, o que pode indicar um possível ponto de melhoria de performance.

## Context and Methodologies

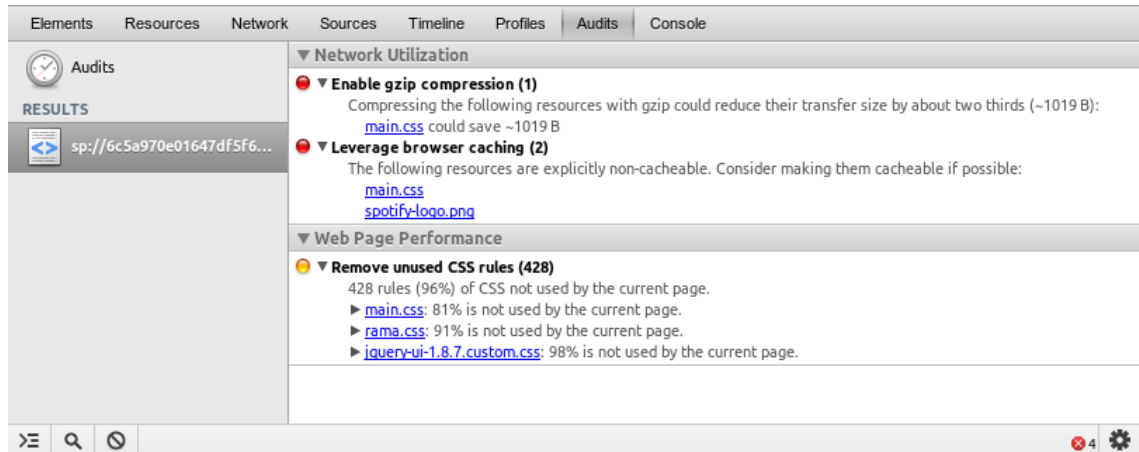


Figure 3.12: Webkit Audit: 96% do código CSS não está a ser usado, sendo por isso, um ponto de melhoria reduzir a quantidade de informação descarregada.

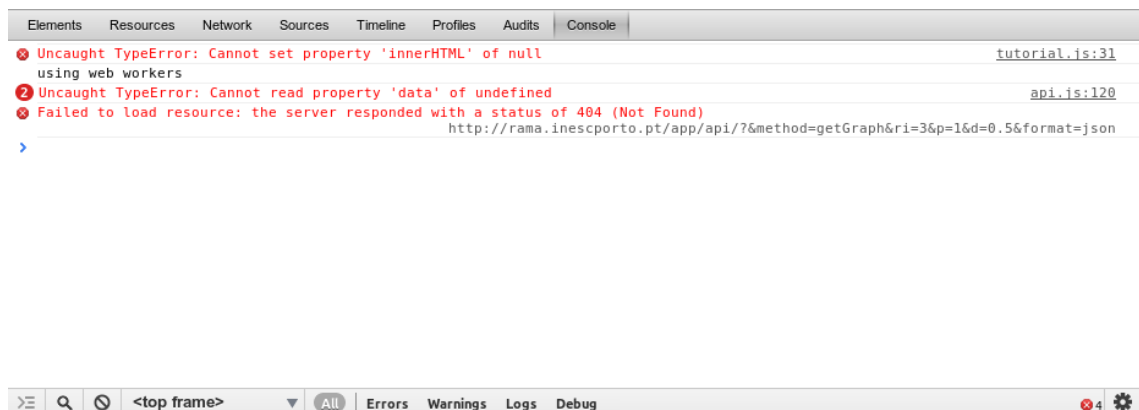


Figure 3.13: Webkit Console: Erros de *Javascript* aparecem destacados para chamar a atenção.



```
5   "grunt-contrib-jshint": "*",
6   "grunt-contrib-jasmine": "*",
7   "grunt-contrib-watch": "*"
8 },
9   "version": "0.1.0"
10 }
```

Listing 3.6: *package.json*: ao indicar a versão com "\*", significa que se deve usar sempre a mais recente.

### 3.2.4 Gruntjs - gruntjs.com

Programa de gestão de tarefas automatizadas. Muito útil para testes, compilação e otimização de código. É possível por exemplo, quando qualquer parte do código mudar, a aplicação automaticamente atualiza com as mudanças mais recentes, sem ser preciso refrescar manualmente a aplicação.

### 3.2.5 Arborjs - arborjs.org

Framework de javascript para desenho de grafos. Foi já utilizada no desenvolvimento do RAMA (existe sempre a possibilidade de se usar outra ferramenta substituta caso esta não for adequada).

## 3.3 Conclusions

A escolha final do módulo a desenvolver é a Aplicação Spotify.

Apesar de as outras opções serem também viáveis, a possibilidade de poder integrar uma interface estilo RAMA num ambiente que os utilizadores já se sentem confortáveis (Spotify), é muito favorável a que seja melhor aceite pelos mesmos.

É esperado que as tecnologias a usar ajudem no desenvolvimento desta dissertação.

Em suma, será desenvolvida uma Aplicação Spotify que implemente os módulos 4 e 5.



# References

- [1] P. Lamere. Creating transparent, steerable recommendations. 2008.
- [2] BG Costa, Fabien Gouyon, e L Sarmiento. A Prototype for Visualizing Music Artist Networks. 2008. URL: [http://www.inescporto.pt/~fgouyon/docs/CostaGouyonSarmiento\\_ARTECH2008.pdf](http://www.inescporto.pt/~fgouyon/docs/CostaGouyonSarmiento_ARTECH2008.pdf).
- [3] L Sarmiento e EC Oliveira. Visualizing networks of music artists with rama. *International Conference on Web ...*, 2009. URL: <http://repositorio-aberto.up.pt/bitstream/10216/15194/2/18675.pdf>.
- [4] Diogo Costa, Luis Sarmiento, e Fabien Gouyon. RAMA : An Interactive Artist Network Visualization Tool. (i):2, 2009. URL: <http://ismir2009.ismir.net/proceedings/LBD-2.pdf>.
- [5] Fabien Gouyon, Nuno Cruz, e Luis Sarmiento. A last.fm and youtube mash-up for music browsing and playlist edition. 2011.