musicl: Music Information Platform

Mariana Aguiar — up201605904@fe.up.pt

Tiago Castro — up201606186@fe.up.pt

Tiago Fragoso — up201606040@fe.up.pt

March 11, 2020

Abstract

*The proposed project — **musicl** — is a full-stack application capable of merging and displaying complementary information on Music. It is divided in two components: a RESTful API and a User Interface. This API will merge data from 3 different APIS: Spotify API, providing unique ids for Albums, Artists and Tracks, Wikipedia API, providing text extracts and Vagalume API, providing music lyrics. All in all, **musicl** seems to be a relevant and approachable project.*

1 Introduction

This article intends to define the proposal of the project to be developed in the Markup Languages and Document Processing course. In order to accomplish this, the article will approach the following topics: **Proposal Definition**, **Data sources and Models**, **State of the Art**, **Related Work** and the **Schedule** for the development of the project.

2 Proposal

The project proposed is an application capable of merging and displaying complementary data from different information sources on the same topic: **Music**.

This application is divided in two components:

- **RESTful API** capable of dynamically fetching complementary information from different public APIs
- **User Interface** which displays the gathered information in an intuitive and user-friendly way

The API will accept and produce **JSON content** and will provide a querying interface as well as details about each entity — **Albums**, **Artists** and **Tracks**.

The content provided by the API will be supported by a User Interface which will help to link the different entities.

3 Data sources and models

The data provided by the platform originates from three different APIs — **Spotify** [1], **Wikipedia** [2] and **Vagalume** [3].

3.1 Spotify API

The Spotify API [1] is the entry point to the platform and also provides the *unique ids* for the **Album**, **Artist** and **Track** entities to be used throughout the **musicl API**. Through the search capabilities of the Spotify API, our platform will also offer this functionality. Finally, detailed information about each entity will also be provided through this API.

The output from the Spotify API is in JSON format and the relevant entities are Album, Artist and Track.

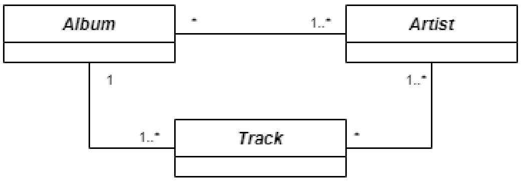


Figure 1: *Relevant Spotify API Data Model*

3.2 Wikipedia API

The Wikipedia API [2] will provide text extracts which, in turn, will accompany the the album and artist pages.

The output from the Wikipedia API is in JSON format but offers no relevant entities for the platform other than a plain text extract from the page in question.

3.3 Vagalume API

The Vagalume API [3] will provide lyrics to be shown on each track page.

The output from the Vagalume API is in JSON format and also offers no relevant entities for the platform other than lyrics in plain text format.

3.4 YouTube API

The possibility of gathering data from a fourth API — **Youtube** [4] — will also be assessed at a later stage.

4 State of the Art

Many music platforms exist, but none of them contain all of the information provided by our platform. For example, **Spotify** (one of our data providers) contains over 30 million songs and 1.2 million artists, but it does not supply the song's lyrics or information about albums and artists (except for metadata).

There are, however, more complete solutions such as **Genius** [5] which offers all our features in their website, through a crowd-sourcing solution where users add lyrics, annotations and information about albums, artists and tracks. On the other hand, their API does not provide the lyrics and the annotations are provided in a *hard-to-process* structure, being tied to song verses (which are not provided).

5 Related Work

The extent of related work is very limited. Although there is a platform — **MusicBrainz** [6] — that combines their musical database with the Wikipedia API to provide other information, it is geared towards music professionals, thus becoming hard to use for regular users.

6 Schedule

The project schedule is represented in the following Gantt chart (fig. 2).

7 Conclusions

All in all, the project is relevant and the proposed solution seems to be approachable and fit to solve the problem at hand. The team feels comfortable with the proposed schedule and is motivated to develop this project.

References

[1] Spotify. Web API — Spotify for Developers. https://developer.spotify.com/documentation/web-api/. Last visited on March 2020.

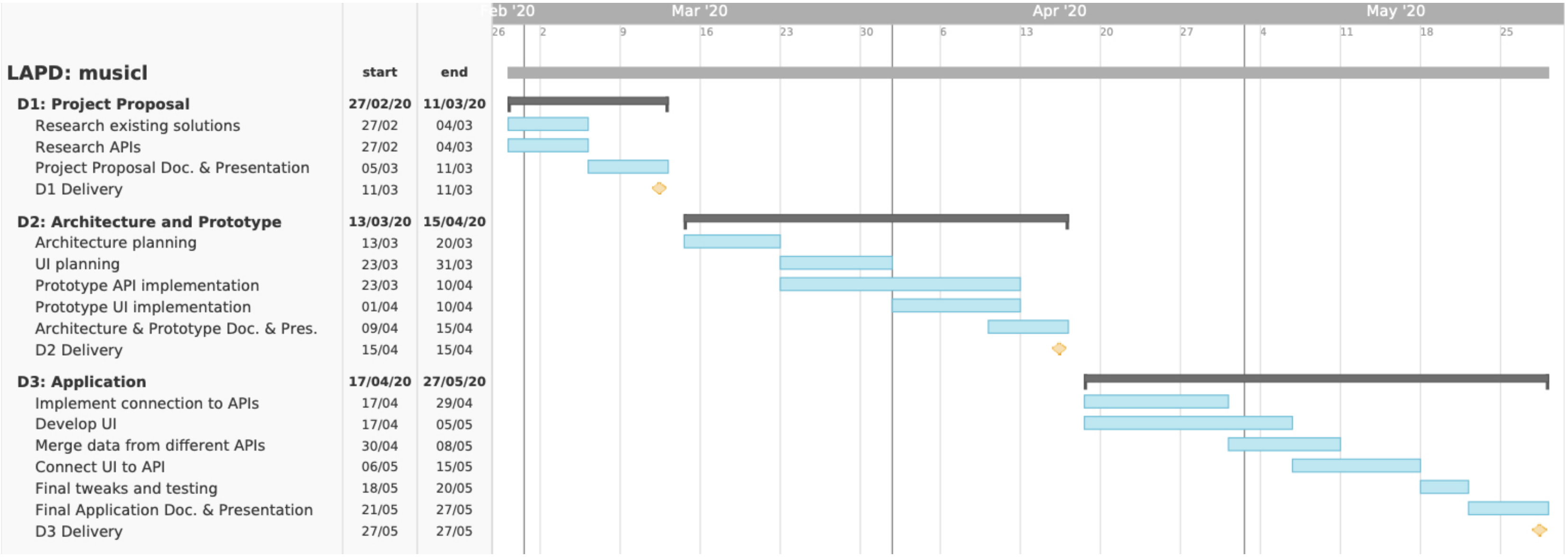
[2] MediaWiki. API: Get contents of a page. https://www.mediawiki.org/wiki/API:Get_the_contents_of_a_page. Last visited on March 2020.

[3] Vagalume. Vagalume API. https://api.vagalume.com.br. Last visited on March 2020.

[4] Google. Youtube Data API — Google Developers. https://developers.google.com/youtube/v3. Last visited on March 2020.

[5] Genius. Genius — Songs Lyrics & Knowledge. https://genius.com. Last visited on March 2020.

[6] MusicBrainz. MusicBrainz — The Open Music Encyclopedia. https://musicbrainz.org. Last visited on March 2020.



LAPD: musicl	start	end
D1: Project Proposal	**27/02/20**	**11/03/20**
Research existing solutions	27/02	04/03
Research APIs	27/02	04/03
Project Proposal Doc. & Presentation	05/03	11/03
D1 Delivery	11/03	11/03
D2: Architecture and Prototype	**13/03/20**	**15/04/20**
Architecture planning	13/03	20/03
UI planning	23/03	31/03
Prototype API implementation	23/03	10/04
Prototype UI implementation	01/04	10/04
Architecture & Prototype Doc. & Pres.	09/04	15/04
D2 Delivery	15/04	15/04
D3: Application	**17/04/20**	**27/05/20**
Implement connection to APIs	17/04	29/04
Develop UI	17/04	05/05
Merge data from different APIs	30/04	08/05
Connect UI to API	06/05	15/05
Final tweaks and testing	18/05	20/05
Final Application Doc. & Presentation	21/05	27/05
D3 Delivery	27/05	27/05

Figure 2: *Project schedule*