Faculty of Computer Science,
"Alexandru Ioan Cuza" University of Iaşi
Academic year 2015-2016

MUXar

Using web semantics to provide music suggestions

Technical report

Project coordinator : **Dr. Sabin-Corneliu Buraga**
Project members :

- **Alexandru Iulian Covaşă** - 2nd year Student, Master of Distributed Software Systems
- **Corina Gabriela Covaşă** - 2nd year Student, Master of Distributed Software Systems
- **Sebastian-Laurenţiu Plesciuc** - 1st year Student, Master of Computer Software Engineering

Abstract

With the increase of web semantic use around the Internet, more and more websites use linked data to provide a better user experience, or use semantic markup to make them more accessible to search engines.

This technical report presents an approach to building a website that uses the advantages of web semantics to present users with music suggestions based on their preferences. The application will gather preliminary data from existing music services (e.g. MixCloud) and will display connections between this data and other musical entities using knowledge offered by linked data repositories (e.g. DBpedia).

Keywords : web semantics, sparql, rdf, music

1. Introduction

The objective of this project is to create a system which will function as a web application that users will access from their PCs, laptops or mobile devices. This application will be able to run on any server or PaaS (e.g. *Amazon EC2*).

The end-user will have access to information and music suggestions based on their use of other musical services (e.g. *MixCloud*). The information presented will be enhanced by the MUXar application using linked data and semantic web technologies from external APIs.

In order to achieve the objective of building the **Musical UX Smart Enhancer** (*MUXar*) application, some software development considerations must be taken into account.

The development of this application aims to serve as an example that such applications that make use of web semantics, from an academical research standpoint, can be built. At most, this application will serve as prototype or reference for commercial applications. As such, this report will not cover a State-of-the-Art analysis.

However, the following sections of this document will discuss user requirements, the architecture of the application, the technologies and external services used.

2. User requirements

This section will address what operations we will expect the user to perform using the MUXar application.

At least the application must be able to provide the following functionality :

- Allow the user to log into the website with an existing account from an existing music service (at least *MixCloud* or *Google Play Music*).

- Allow the user to view suggestions based on playlist or history from their external accounts (songs/artists/albums that have connections to their listening habits).
- Allow the user to search for music.
- Allow the users to create and save a playlist on our website. (This might require the setup of an Openlink Virtuoso server to hold the data in RDF format).

Requirements that are influenced by limitations of external services :

- Allow the user to create playlist and synchronize it with their accounts on external services (e.g. save the playlist on their *MixCloud* account).

These requirements are reflected in the following Use-Case diagram with only *MixCloud* and *DBpedia* as external sources :

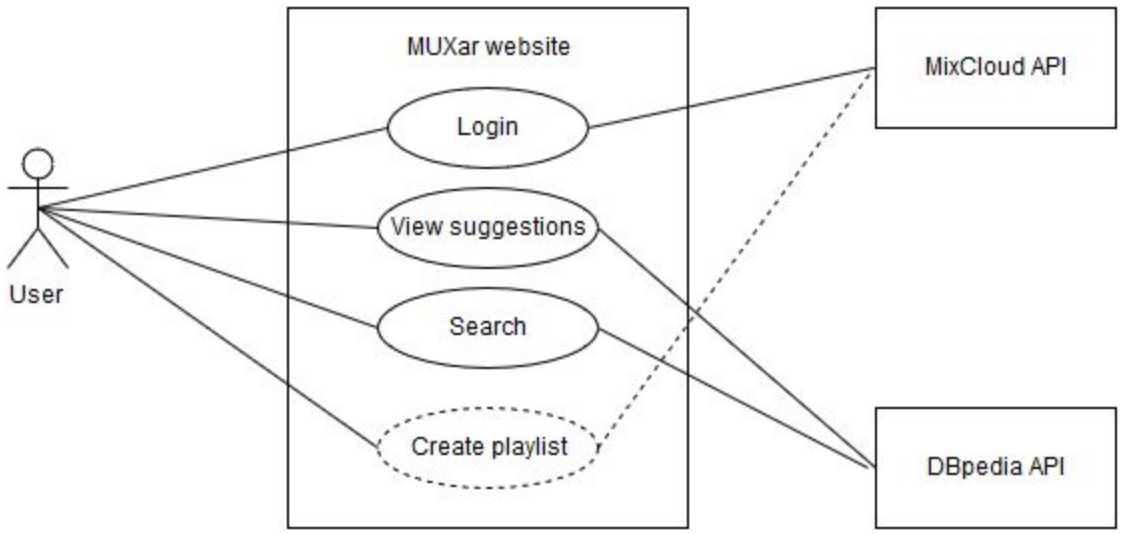

Fig. 2.1. Use-Case Diagram

The diagram shown in figure 2.1 describes which use case interacts with which entity. As such, the *login* and *create playlist* functions interact with the APIs that provide **user** information via simple *HTTP* requests. The *search* and *suggestions* functions provide **music** information via semantic web technologies (e.g.*SPARQL*). The *Create playlist* function is represented with dotted lines since the playlists are synchronized, if possible, with the user account on that particular website. Otherwise, the playlists are stored in *RDF* format in the projects *Virtuoso* server.

3. Technologies

In order to accommodate the user requirements described in the previous sections, the technologies used must be chosen accordingly. From a software development standpoint, considerations regarding the time required to develop or other such criteria might be taken into

account. However, for the purpose of this project, we will only focus on technologies that have been proven as suitable for web application development and that support the protocols required.

The front-end of this application will consist of an *Angular.js* controller that will handle user interactions and the communication with the MUXar web service. On this controller the user interface will be built with *Angular Material.*

On the back-end, the main controller of the application will be built using *Node.js* which has native server support for HTTP/HTTPS connections. Modules for *Node.js* that facilitate the development of this application will be used, some of the more important ones are *Passport.js* middleware for user authentication and *Express.js* middleware for routing. Other modules will be used as needed for *SPARQL* and *Redis* communication. A *Redis* store will be used to store user sessions and other data. Depending on how the requirements might change during development, the possibility of using an RDF store cannot be excluded. For this *Openlink Virtuoso* might be used.

4. Architecture

Using the technologies described in section 3, the following components diagram describes the application architecture :

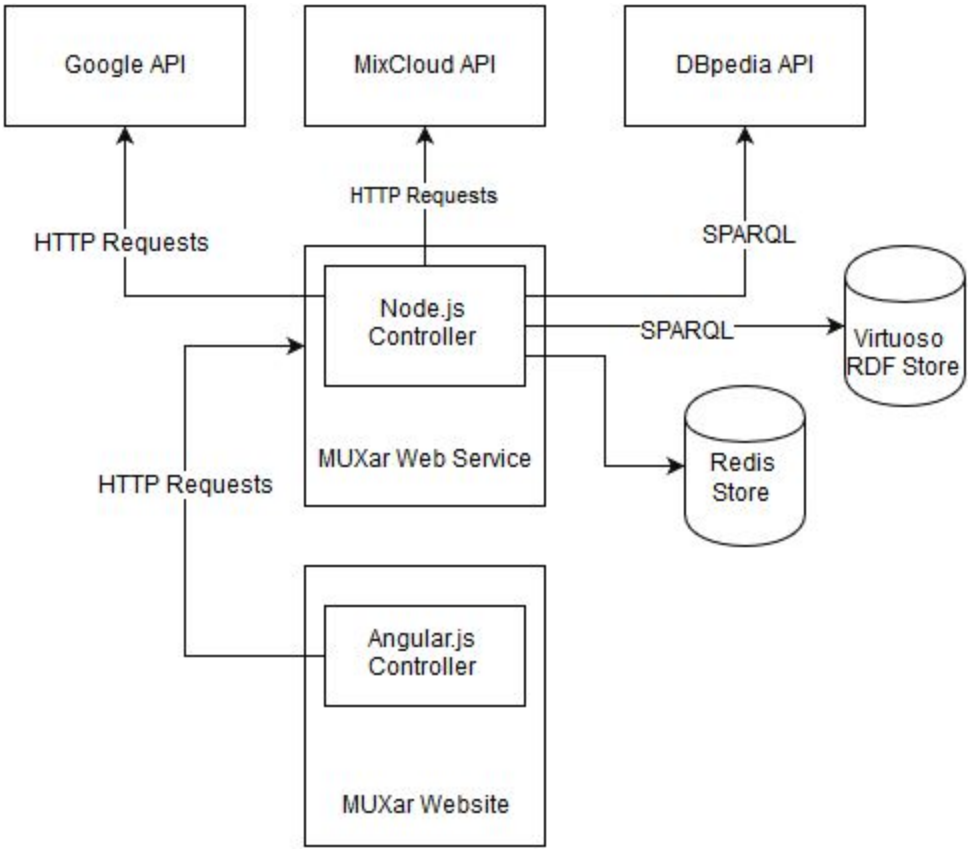

Fig. 4.1. Components diagram

4. External services

For this project to satisfy the requirements specified in section 2 of this document, the application will have to rely on a number of external services in order to work.

When this document was written, the most reliable sources were :

- **MixCloud** - external service that allows users to listen to music created by independent artists and create playlists. Provides a web API that is documented and has good uptime.
- **DBpedia** - external service that provides linked data sets.
- **Google Play Music** - external service that allows user to listen to music, buy albums and create playlists. There is no official API but several unofficial APIs exist.

Previously considered sources :

- **Last.fm** - external beta service that allows users to listen to music and create playlists. It no longer provides API keys.
- **Seevl.fm** - external service that provides music discovery and personalization. Suffers from frequent downtime and very high response times.
- **Freebase** - external service that provides linked data sets. Has announced that it has become read-only since March 2015 and the API will be retired.

5. Entities and relationships

In order to link the data used in our application, a standard model for data representation must be chosen. The first step is to identify the main entities related to music :

- **Song**
- **Artist**
- **Album**
- **Concert**
- **Composer**
- **Event**
- **Band**

For each of these entities, an appropriate model from *schema.org* will be identified and used. The markup displayed by the website for each of the entities will identify with the semantic meaning associated for them and their properties as described by the *schema.org* vocabulary.

The properties of these entities that will be taken into account when providing suggestions are :

Song entity:
- name
- artist
- duration
- genre
- album
- release date
- composer

Artist entity:
- name
- age
- place of birth
- no of albums
- no of songs
- no of concerts he attended

Album entity:
- title
- release date
- artist
- [songs]
- has received a special award

Composer entity:
- name
- age
- place of birth
- is he also a singer
- no of sings written
- [songs written]In

Concert entity:
- location
- date
- [artists/bands]
- duration
- no of participants

6. Conclusion

This preliminary technical report serves as a reference for the requirements needed to undertake this project.

The details of this project and the code which implements it can be viewed at the following Github page : https://github.com/wadeproject/muxar_wade

Preliminary tests using technologies mentioned in this report can be found here : https://github.com/wadeproject/muxar_prototypes_test

7. References

- DBpedia Wiki : http://wiki.dbpedia.org/
- MixCloud Developer Documentation : https://www.mixcloud.com/developers/
- Node.js Documentation : https://nodejs.org/en/docs/
- Angular.js Documentation : https://docs.angularjs.org/guide
- Unofficial Google Play Music API : https://github.com/jamon/playmusic
- OpenLink Virtuoso Documentation : http://docs.openlinksw.com/virtuoso/
- Schema.org Documentation : http://schema.org/docs/documents.html