MusicPitch

Architecture and Integration of Software Systems

Software Engineering Degree

2nd year

José Manuel Bellido Cuesta (josembell97@gmail.com)
José Antonio Carmona Fombella (joanca.carmona597@gmail.com)
José Luis Del Pino García (josedelpg@gmail.com)
Alicia Viñas Ordóñez (aliciavo97@gmail.com)

Tutor: Adela del Río Ortega
Group number: ADR-Grupo ING-MusicNess

Application link: http://www.music-pitch.appspot.com
Project link on GitHub: https://github.com/aliviaord/MusicPitch

HISTORY OF VERSIONS

Date	Version	Details	Participants
14/03/2014	1.0	- Includes an introduction, user interface mockups and UML component diagram and UML deployment diagram.	José Manuel José Antonio José Luís Alicia
		<Mention the most significant changes regarding the previous version>	

Index

1. Introduction

In our current society, we all have a fight against time. We want to extend our day to more than 24 hours, but it doesn't matter how much time we get, we need more. This is the reason why every process that wastes this invaluable time must be avoided as soon as possible.

On the other hand, for most of the population, music is a main pillar when it comes to free time, and, due to globalization, English music is usually the preferred option. However, for most of us, our level of English is far from perfect and we resort to the written lyrics to understand the meaning of our favorite artist's song.

Finally, we are also social beings and lyrics can be used as a way of expressing our frame of mind and sharing it with our circle of friends.

MusicPitch is a mashup for music lovers. In a single place, you can search for songs, albums and artists' profiles. Each of this element are brought to its maximum expression:

·Not only will you be able to listen to your favorite song, you will also be able to read its lyrics, watch its video clip and share your favorite extract from the lyrics on Facebook.

·Not only will you be able to see your favorite album, you will also have access to all its songs (bringing all the possibilities mentioned before), add it to your Spotify's playlist or purchasing it on Amazon.

·Not only will you be able to see the profile of your favorite artist but also viewing its Facebook's profile, reading its biography, accessing to all its albums and knowing when and where his next concerts will take place.

All of this lets us to save our time and energy having all these possibilities united in a single place. Consuming music becomes easier, more intuitive, more enriching and faster, and all of this, with a single click.

1.1. Integrated applications

We will now explain the way in which we use each API:

Facebook: It will be the way in which we show the artists' Facebook profiles and also where we will post extracts from the lyrics.

Spotify: Spotify will be used to search for songs, albums and artists. This will not only bring us the metadata but the cover of an album or a photo of an artist.

MusixMatch: This API will give us the lyrics of a song using the metadata obtained previously with Spotify.

Youtube: Youtube will let us display the video clip of a song next to its lyrics.

MediaWiki: The MediaWikiAPI will give us the possibility to show the biography of the artists.

SongKick: Songkick will help us to show the next concerts of an artist.

ImageColorExtraction: This API will use the cover of the album, previously obtained from Spotify, to find the most important color in the picture and we'll use this color as the main one for the album's webpage.

Nombre aplicación	URL documentación API
Facebook	https://developers.facebook.com/docs/
Spotify	https://developer.spotify.com/web-api/
MusixMatch	https://developer.musixmatch.com/documentation
Youtube	https://www.youtube.com/yt/dev/es/api-resources.html
MediaWiki	https://www.mediawiki.org/wiki/API:Main_page
SongKick	http://www.songkick.com/developer
ImageColorExtraction	https://market.mashape.com/nijikokun/image-color-extraction

TABLE 1. INTEGRATED APPLICATIONS

1.2. Project evolution

Es habitual que la aplicación final diste mucho de la idea inicial. Puede que la idea fuese muy compleja, no haya sido posible integrar alguna de las aplicaciones o alguno de los miembros del grupo haya abandonado. Explicar en esta sección cuál ha sido la evolución del proyecto, problemas, cambios, decisiones, etc.

2. User interface mockups

2.1. Main Page

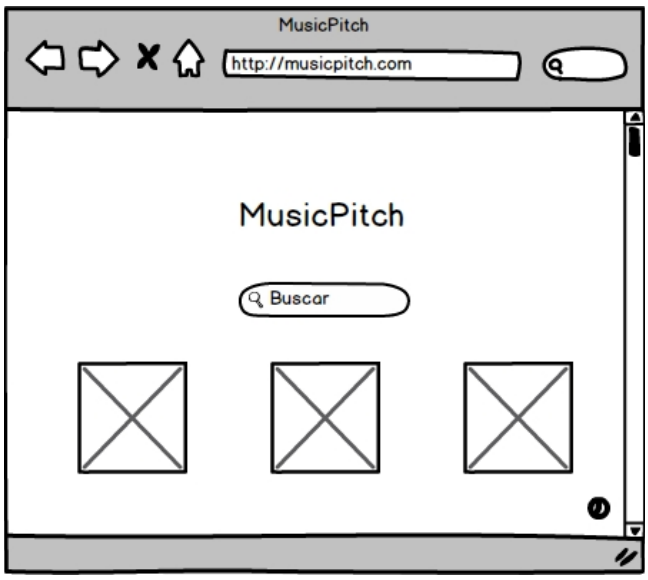


FIGURE 1. PROTOTYPE OF THE USER INTERFACE OF THE MAIN PAGE VIEW

This is the main page of MusicPitch. In this section, you will be able to search for songs, artists and albums in just one click. There are three pictures that represent the songs, the artists and the albums, which will be useful to distinguish the results in the following page.

2.2. Results' View

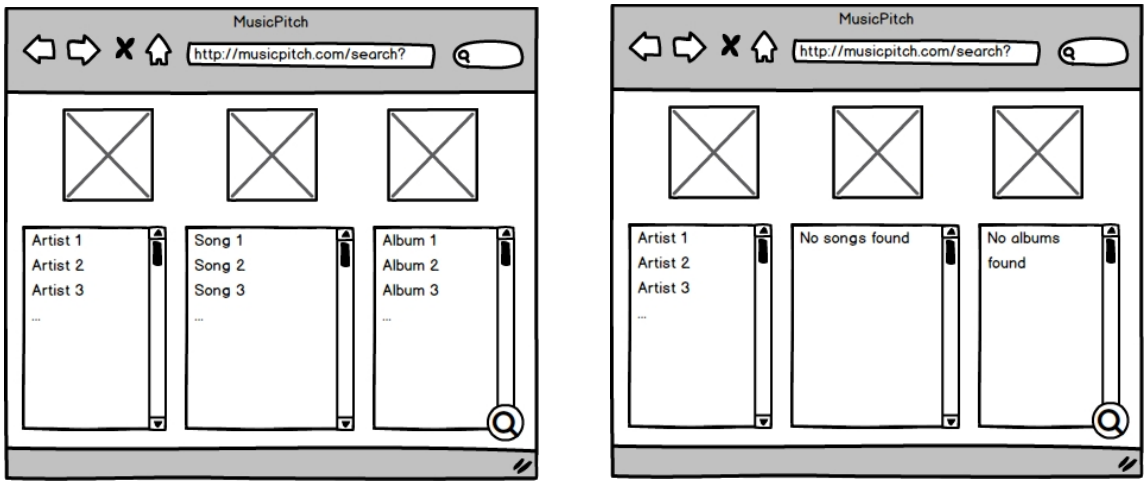


FIGURE 2. PROTOTYPE OF THE USER INTERFACE OF THE RESULTS' VIEW

This is the page where the results are shown. In the three columns you will be able to find the artists, songs and albums (in that order) you expected. Also, in case you need to make a new search, you will find a "Search" button in the bottom right corner, that will redirect you to a new results page. This button will appear in every following page in order to make faster searches.

The second mockup represents the view of the interface in case there weren't any results of a specific type.

2.3. Song's View

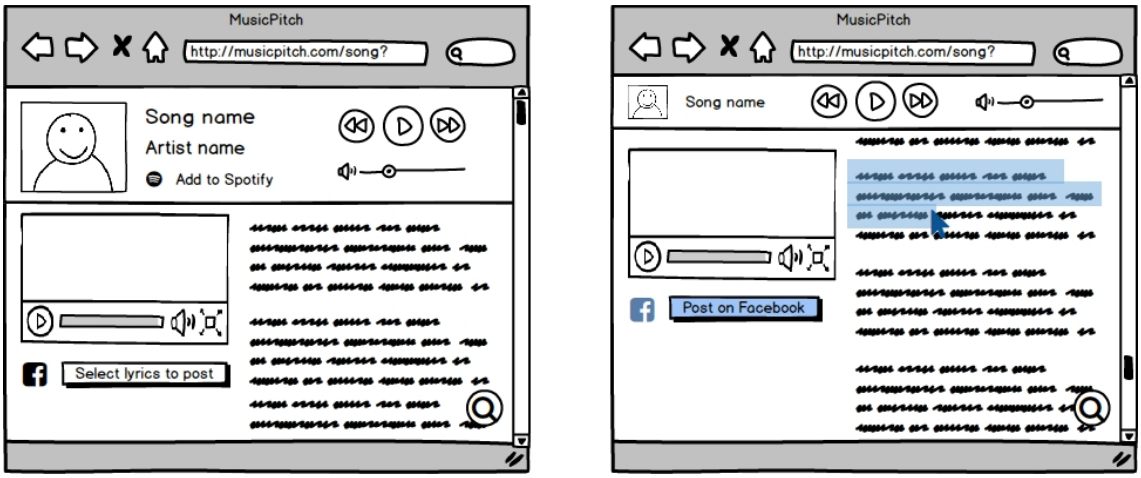


FIGURE 3. PROTOTYPE OF THE USER INTERFACE OF THE SONG'S VIEW

This is the song's interface. At the top of the page, you will see a widget that includes a player and some information about the song (name, artist and cover). Also, you will be able to add the song to your Spotify account in case you'd like it. This widget will always appear everywhere in the song's interface in order to have a better control of the song being played.

Right under the widget, you will see the lyrics of the song being played and a YouTube videoclip, in case you want to watch it. Also, you will be able to select a fragment of the lyrics and share it on Facebook.

2.4. Artist's View

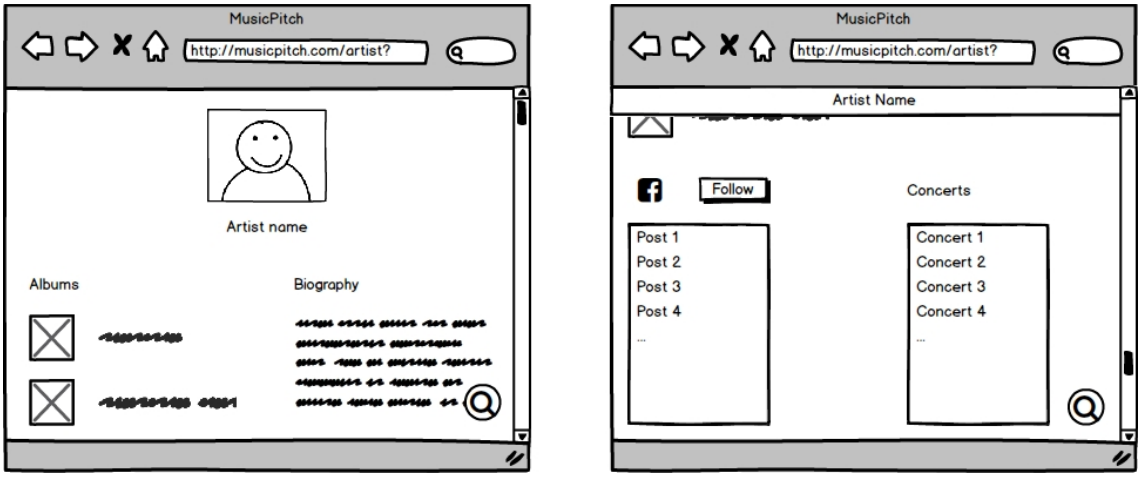


FIGURE 4. PROTOTYPE OF THE USER INTERFACE OF THE ARTIST'S VIEW

This is the artist's interface, which is divided in five parts. The first one, is a main picture of the artist, which is displayed at the top of the page. Under the picture, you will see the artist's recent albums and a short biography extracted from Wikipedia.

Below the albums and the biography, you will see the latest posts from the artist on Facebook and a small section where you will be able to see his/her upcoming concerts from SongKick.

2.5. Album's View

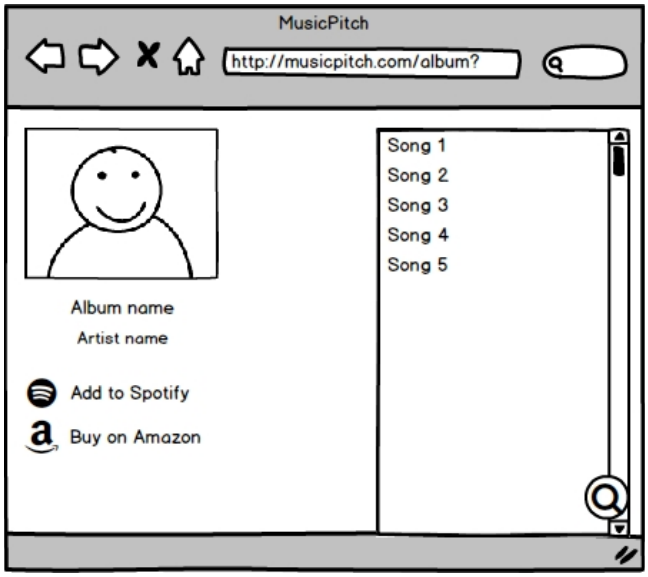


FIGURE 5. PROTOTYPE OF THE USER INTERFACE OF THE ALBUM'S VIEW

This is the album's interface. It shows the main information about the album (name, artist and the cover) and the list of the songs included. Also, you will be able to add it in your Spotify account or buy the physical disc in Amazon in just one click.

2.6. Error's View

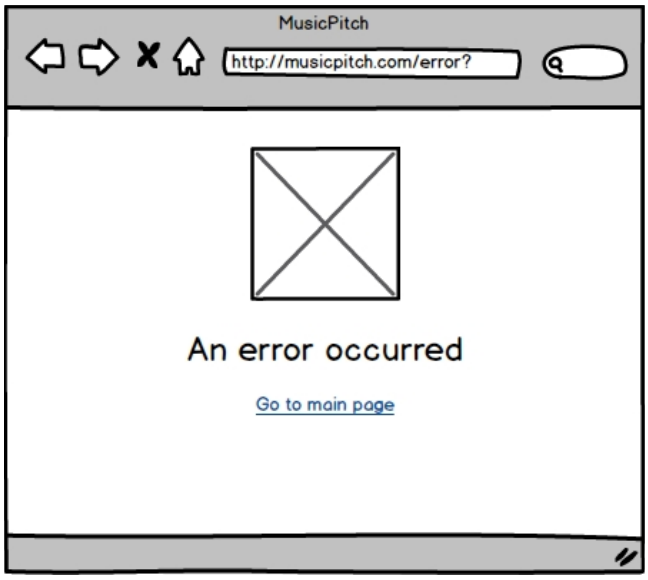


FIGURE 6. PROTOTYPE OF THE USER INTERFACE OF THE ERROR'S VIEW

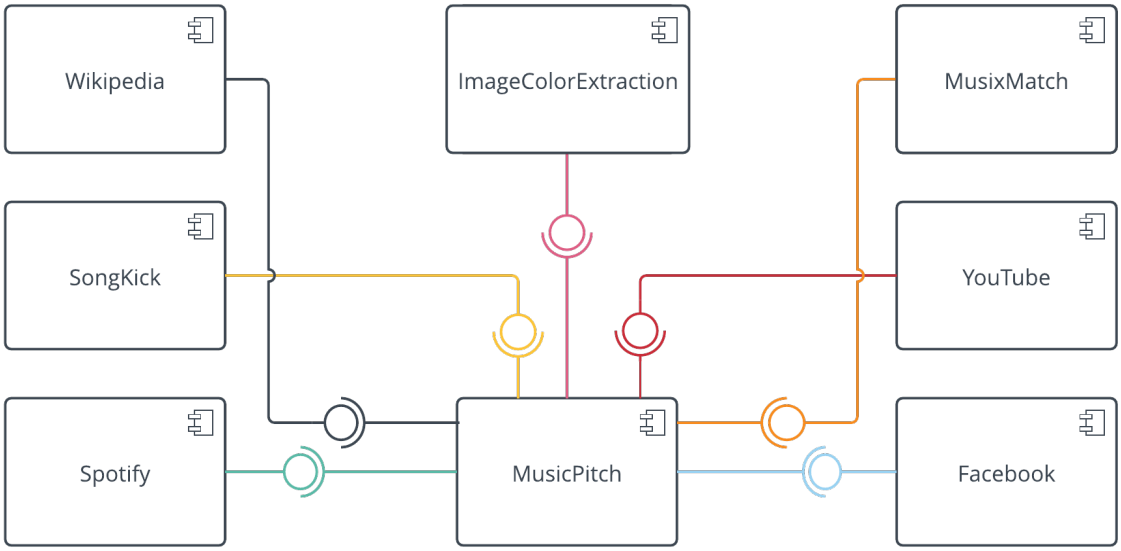
This page will be used in case an error occurred. Below the specification of the error, there will be a link that will redirect you to the main page in order to make a new search.

3. Architecture

Insertar los diagramas UML de componentes y de despliegue de la aplicación. Describir textualmente

3.1. Component diagram

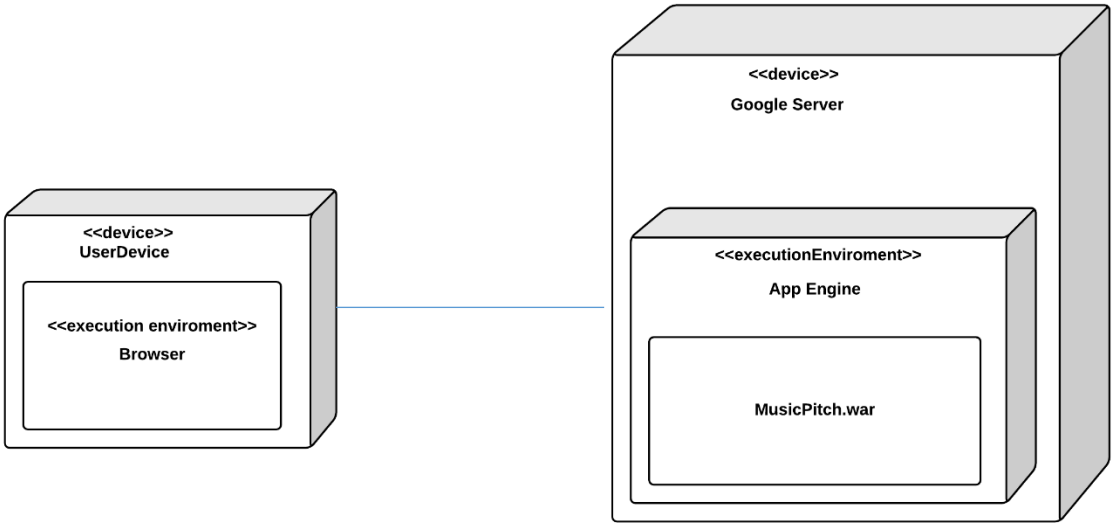
High level UML component diagram. Includes the integrated applications and our own application as an independent component.



This diagram shows a basic representation of our system's logic architecture. It consists of a serie of eight component: seven of these component are applications integrated in our application, and the eighth component is our own application. These components are related by their interfaces. In our case, Wikipedia, SongKick, Spotify, ImageColorExtraction, MusixMatch, YouTube and Facebook provide their interfaces to MusicPitch.

3.2. Deployment diagram

UML deployment diagram of the application.



This diagram shows the physical architecture on which our software system is deployed, it describes both physical devices and software elements. It consists of an artifact and four nodes: two devices and two execution environments.

On the left there is the general device that the user is using (smartphone, PC, …), a hardware node. It contains inside another node, in this case, an execution environment. That execution environment is the web browser from where the user will be able to access to our software system.

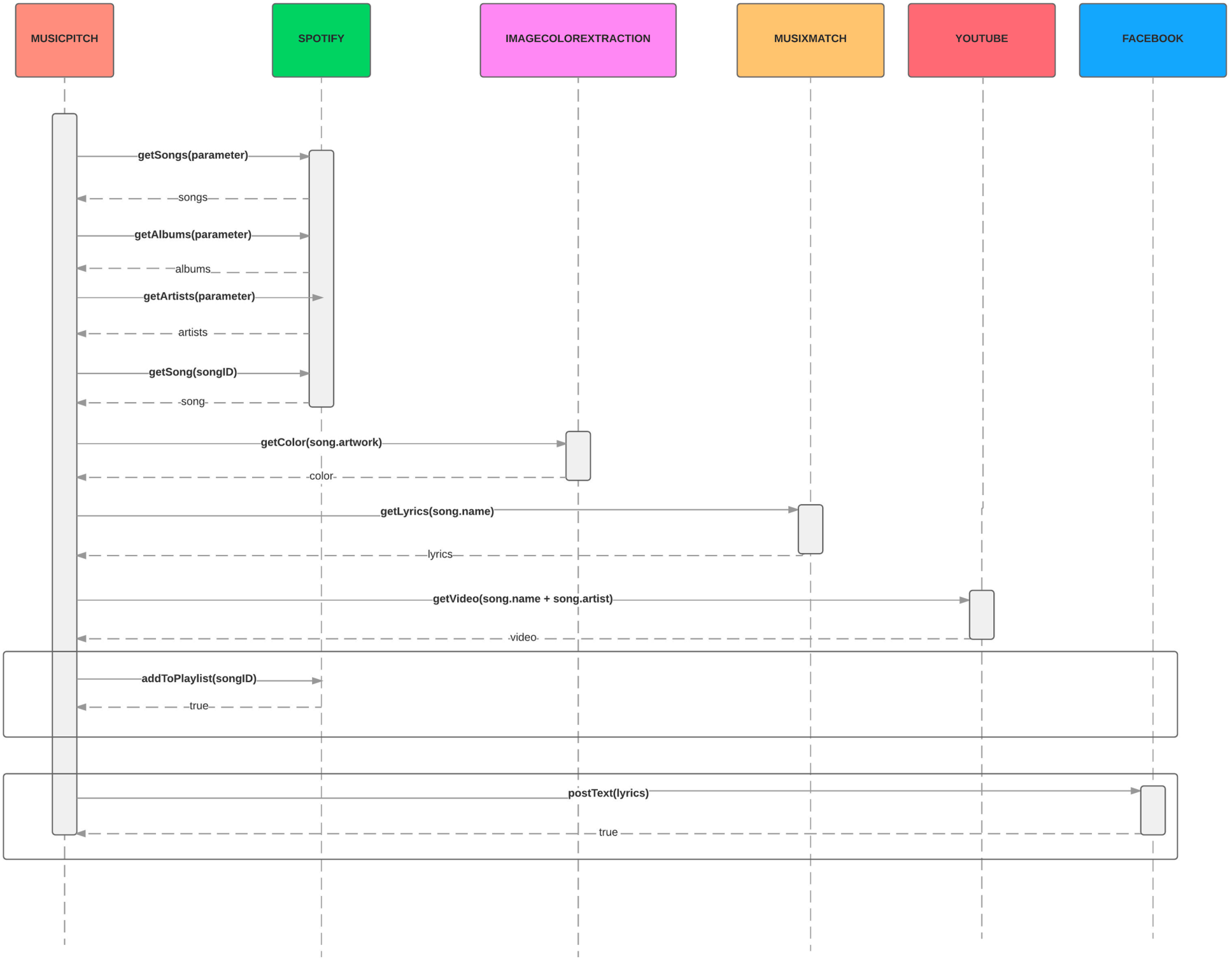
The path that link both sides is the connection being used, for example, Wi-Fi connection or Ethernet connection between both devices.

On the right there is the Google server which contains the Google AppEngine execution environment, and from where our system's artifact or file is deployed.

3.3. High level sequence diagram

UML sequence diagram which indicates the message flow between the different integrated applications.

In our case, we decided to divide the high level sequence diagram in three, so we can represent clearly and appropriately the message flow that happens according to each of the three base cases that our system allows.


MUSICPITCH
SPOTIFY
IMAGECOLOREXTRACTION
MUSIXMATCH
YOUTUBE
FACEBOOK
getSongs(parameter)
songs
getAlbums(parameter)
albums
getArtists(parameter)
artists
getSong(songID)
song
getColor(song.artwork)
color
getLyrics(song.name)
lyrics
getVideo(song.name + song.artist)
video
addToPlaylist(songID)
true
postText(lyrics)
true

SONG DIAGRAM

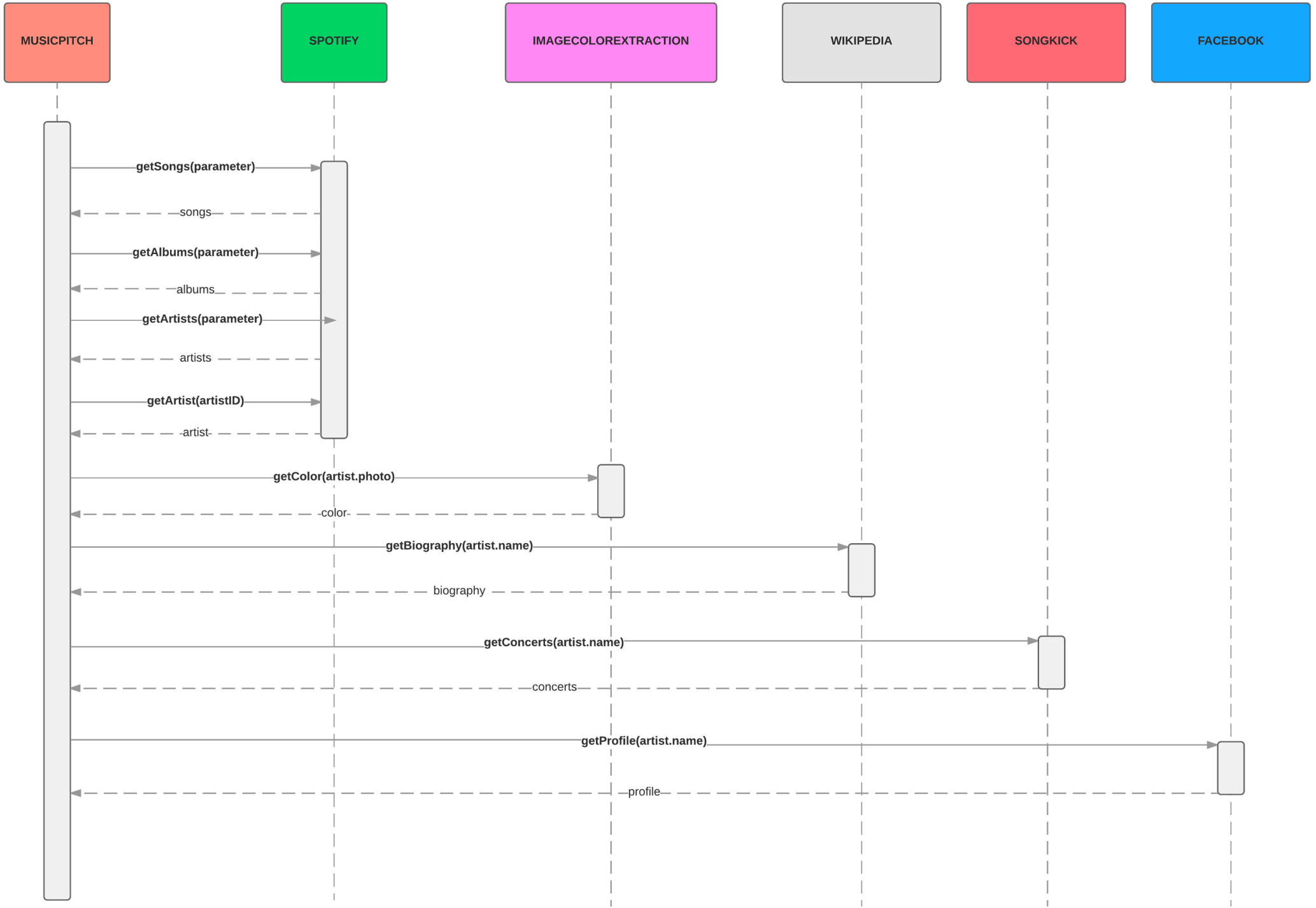
Firstly, this is the diagram in case that the user is interested in a song search.

The user searches a term that will later be the parameter of three petitions from MusicPitch to Spotify in order to obtain songs, albums and artists results related to that parameter.

When the search results are shown to the user, in this first case, he or she decides to click on some song from the ones that are shown on the screen. This song will be obtained from Spotify using its ID, which was already obtained in the first general search.

Then, we make a petition to ImageColorExtraction with the chosen song's álbum artwork, in order to obtain its primary color and use it as a background color of the web page. We also make a petition to MusixMatch in order to obtain the chosen song's lyrics and a last petition is made to YouTube in order to obtain the video that best suits these parameters.

Furthermore, there exists two optional actions that the user can make and that would require additional messages between the applications. If he or she is interested in adding the chosen song to his or her Spotify playlist, we will need to send another petition to Spotify. Again, if what he or she wants is to share a lyrics fragment on his or her Facebook profile, a petition to Facebook will be required.


MUSICPITCH
SPOTIFY
IMAGECOLOREXTRACTION
WIKIPEDIA
SONGKICK
FACEBOOK
getSongs(parameter)
songs
getAlbums(parameter)
albums
getArtists(parameter)
artists
getArtist(artistID)
artist
getColor(artist.photo)
color
getBiography(artist.name)
biography
getConcerts(artist.name)
concerts
getProfile(artist.name)
profile

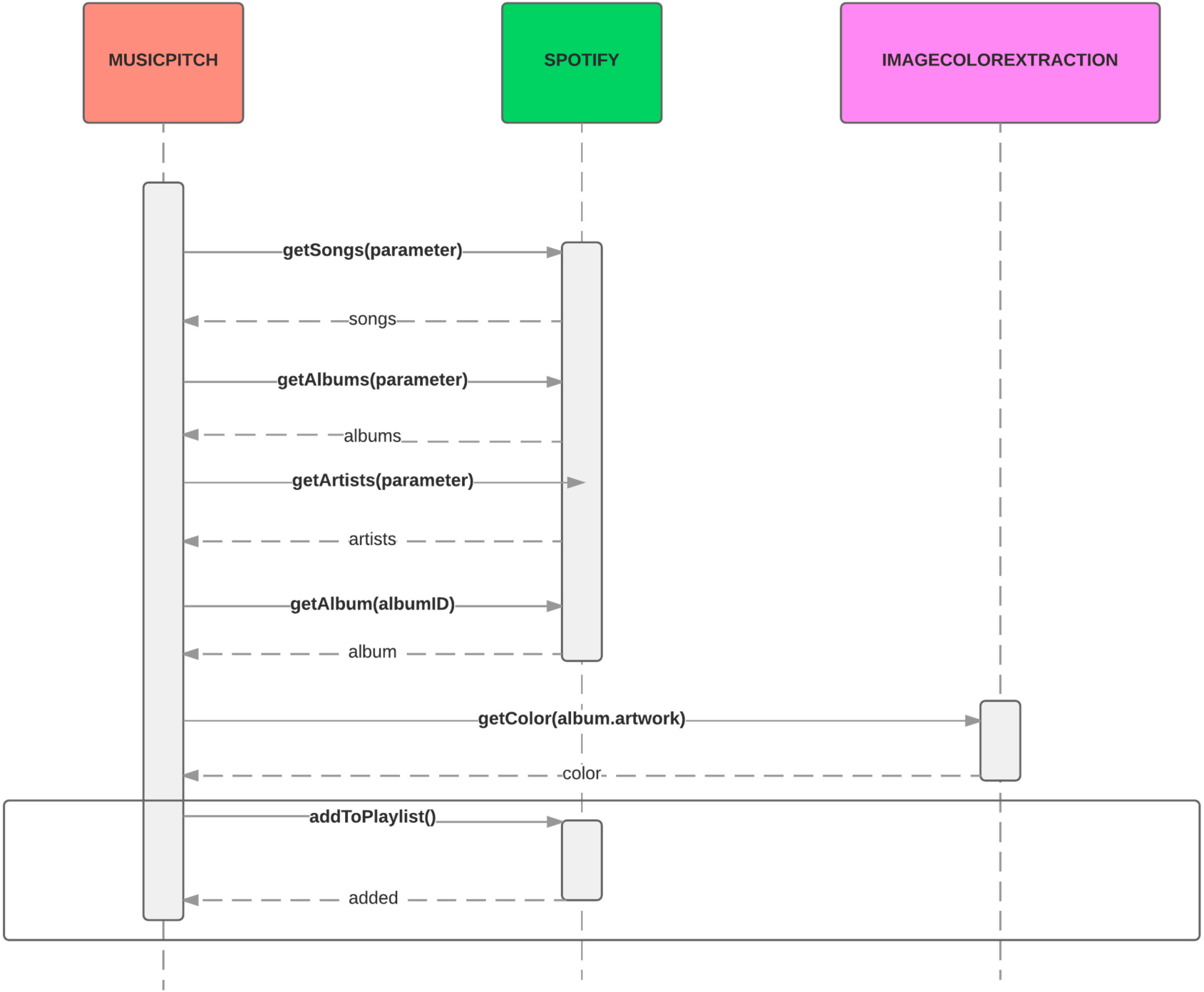
ARTIST DIAGRAM

Secondly, this is the diagram in case that the user is interested in an artist search.

The user searches a term that will later be the parameter of three petitions from MusicPitch to Spotify in order to obtain songs, albums and artists results related to that parameter.

When the search results are shown to the user, in this second case, he or she decides to click on some artist from the ones that are shown on the screen. This artist will be obtained from Spotify using its ID, which was already obtained in the first general search.

Then, we make a petition to ImageColorExtraction with the chosen artist's profile picture, in order to obtain the primary color and use it as a background color of the web page. A petition to Wikipedia is made with the artist name in order to obtain his or her biography, another petition is made to SongKick in order to obtain information about the artist's latest concerts and a last petition is made to Facebook in order to obtain the chosen artist's profile on this social network.


MUSICPITCH
SPOTIFY
IMAGECOLOREXTRACTION
getSongs(parameter)
songs
getAlbums(parameter)
albums
getArtists(parameter)
artists
getAlbum(albumID)
album
getColor(album.artwork)
color
addToPlaylist()
added

ALBUM DIAGRAM

Thirdly and in last place, this is the diagram in case that the user is interested in an album search.

The user searches a term that will later be the parameter of three petitions from MusicPitch to Spotify in order to obtain songs, albums and artists results related to that parameter.

When the search results are shown to the user, in this third case, he or she decides to click on some album from the ones that are shown on the screen. This album will be obtained from Spotify using its ID, which was already obtained in the first general search.

Then, we make a petition to ImageColorExtraction with the chosen album's artwork, in order to obtain the primary color and use it as a background color of the web page. This is the last petition needed in this case, due to the fact that all the information needed about the álbum was already obtained previously.

Furthermore, there exists an optional action that the user can make and that would require additional messages between the applications. If he or she is interested in adding the chosen album to his or her Spotify playlist, we will need to send another petition to Spotify.

3.4. Class diagram

Diagrama UML de clases indicando la distribución de las clases entre las distintas capas, según el patrón MVC.

3.5. Sequence diagrams

Diagramas UML de secuencia ilustrando la comunicación entre vistas, controladores y clases del modelo.

4. Implementation

Describir brevemente los aspectos de la implementación que creen da más mérito al trabajo. Añadir algún fragmento de código si se considera oportuno.

5. Testing

Documentar las pruebas realizadas a la aplicación. Justificar textualmente la estrategia de pruebas seguida y por qué (ej. pruebas incrementales ascendentes).

Indicar el número total de pruebas realizadas y cuáles de ellas han sido automatizadas mediante JUnit.

Resumen	
Número total de pruebas realizadas	25
Número de pruebas automatizadas	20 (80%)

ID	**Prueba 1**
Descripción	Prueba para la detección de errores al implementar búsquedas en Spotify usando servicios RESTful.
Entrada	Se hace uso de la librería XXX para invocar al servicio usando la URI YYY desde nuestra aplicación.
Salida esperada	Los datos devueltos en formato JSON son mapeados a una clase Java y a continuación se muestran por pantalla.
Resultado	EXITO
Automatizada	Sí

6. User guide

6.1. Mashup

Indique textualmente e incluyendo capturas de pantalla el manual de uso del mashup.

6.2. API REST

Indique la documentación de la API REST (contrato) implementada [2]. Cómo mínimo, la API debería incluir:

1. Protocolo de aplicación empleado por el servicio.
2. URIs para invocar a las operaciones del servicio.
3. Formato empleado para las representaciones de los recursos.
4. Códigos de estado empleados por el servicio.
5. Ejemplos de uso.

Está información también debe facilitarse en formato HTML como parte de la aplicación.

References

[1] Balsamiq. http://balsamiq.com/. Accedido en Enero 2014.

[2] J. Webber, S. Parastatidis y I. Robinson. REST in Practice: Hypermedia and Systems Architecture. O'Reilly Media. 2010.