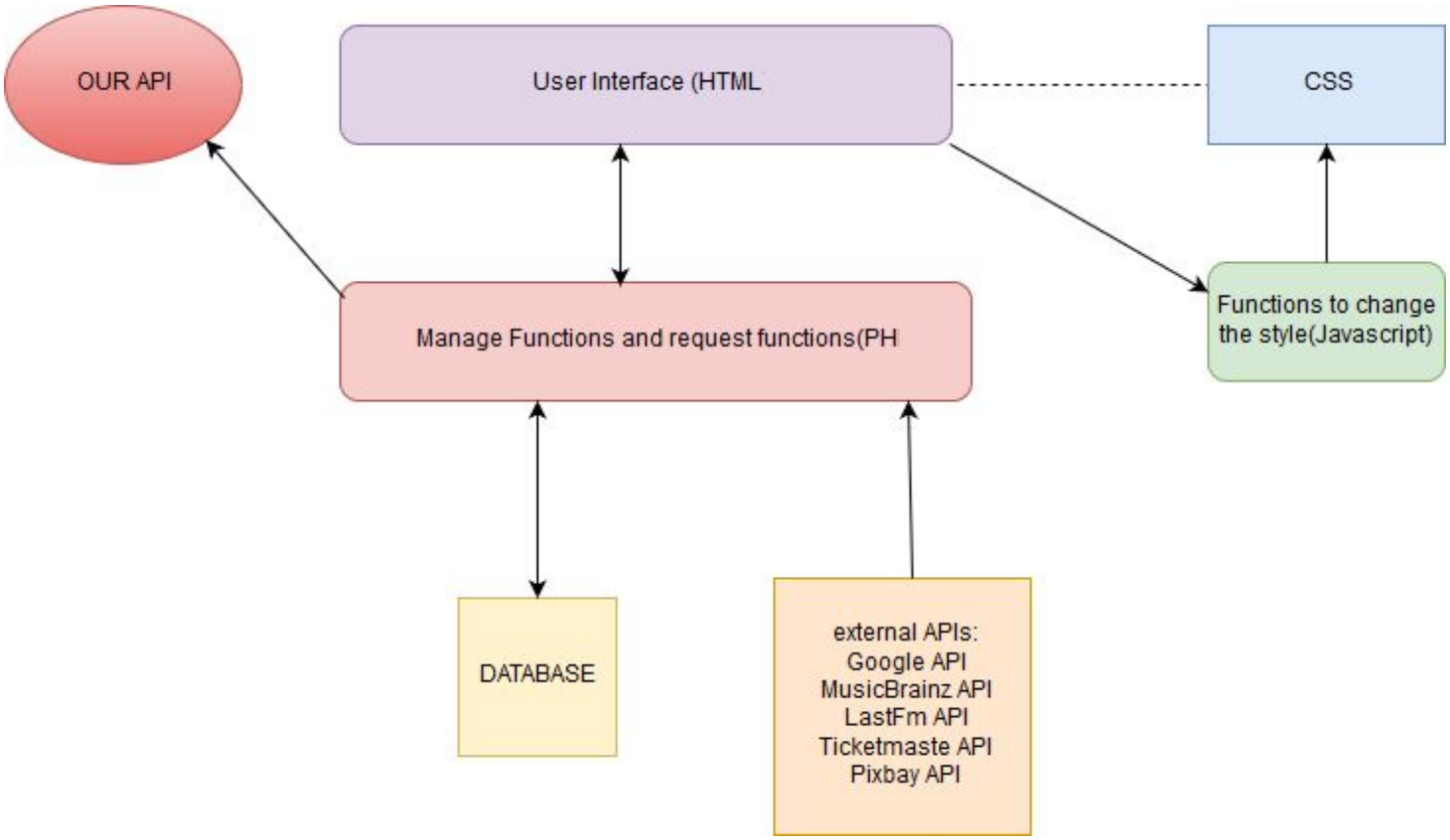
ARCHITECTURE OF ORPHEUS

José Manuel Rodríguez Calvo

Jesús Jiménez Sánchez

Table of contents

Plan for architecture

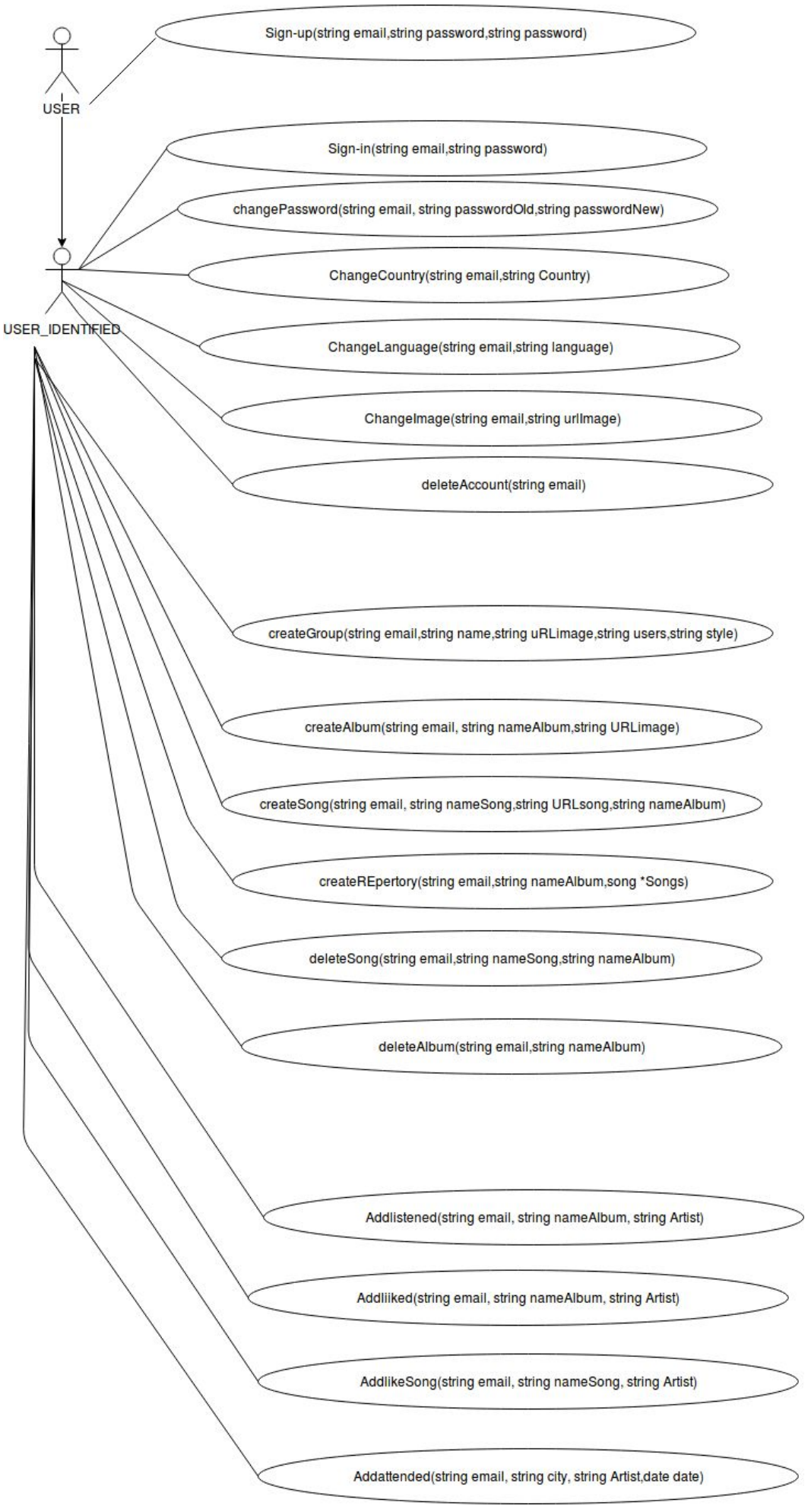


In this diagram, we show the behaviour of our web app and how it interacts with the user sending the right information in an organised way.

The first thing we can see is that the user will interact with the HTML, which has a CSS file to add the style and, therefore, be more organised and effective for the user.

The HTML file does also the task of getting information from the user, for example from a text input box or a button. When the user interacts with some of these elements, a request to the database to create, modify or delete a value.

Below we can see the functions that will control the information and how the different types of user can interact with them.


USER
USER_IDENTIFIED
Sign-up(string email,string password,string password)
Sign-in(string email,string password)
changePassword(string email, string passwordOld,string passwordNew)
ChangeCountry(string email,string Country)
ChangeLanguage(string email,string language)
ChangeImage(string email,string urlImage)
deleteAccount(string email)
createGroup(string email,string name,string uRLimage,string users,string style)
createAlbum(string email, string nameAlbum,string URLimage)
createSong(string email, string nameSong,string URLsong,string nameAlbum)
createREpertory(string email,string nameAlbum,song *Songs)
deleteSong(string email,string nameSong,string nameAlbum)
deleteAlbum(string email,string nameAlbum)
Addlistened(string email, string nameAlbum, string Artist)
Addliiked(string email, string nameAlbum, string Artist)
AddlikeSong(string email, string nameSong, string Artist)
Addattended(string email, string city, string Artist,date date)

All functions shown above are responsible for modifying the values in the database, for example, *changePassword(string email, string passwordOld, string passwordNew)* is responsible for updating the password, *createSong(string email, string nameSong, string URLsong, string nameAlbum)* will add a song to the database and *addAttended(string email, string city, string Artist, date date)* will add a concert to the list of concerts a user has attended.

In the next diagram we show the functions we use in Javascript, these functions are able to read the database and show the obtained information in the web app, with the help of the API in PHP. Here are some examples:

showRecommendationStyle(string email) will be in charge of consulting in the database which is the music style the user listens to the most, identifying him using his email. It will get the frequencies of each style and will show, from the top five, one chosen randomly.

showNearConcerts(string email) will be in charge of calling the PHP or Javascript which will return the nearer concerts depending on the location the user gave when he registered.

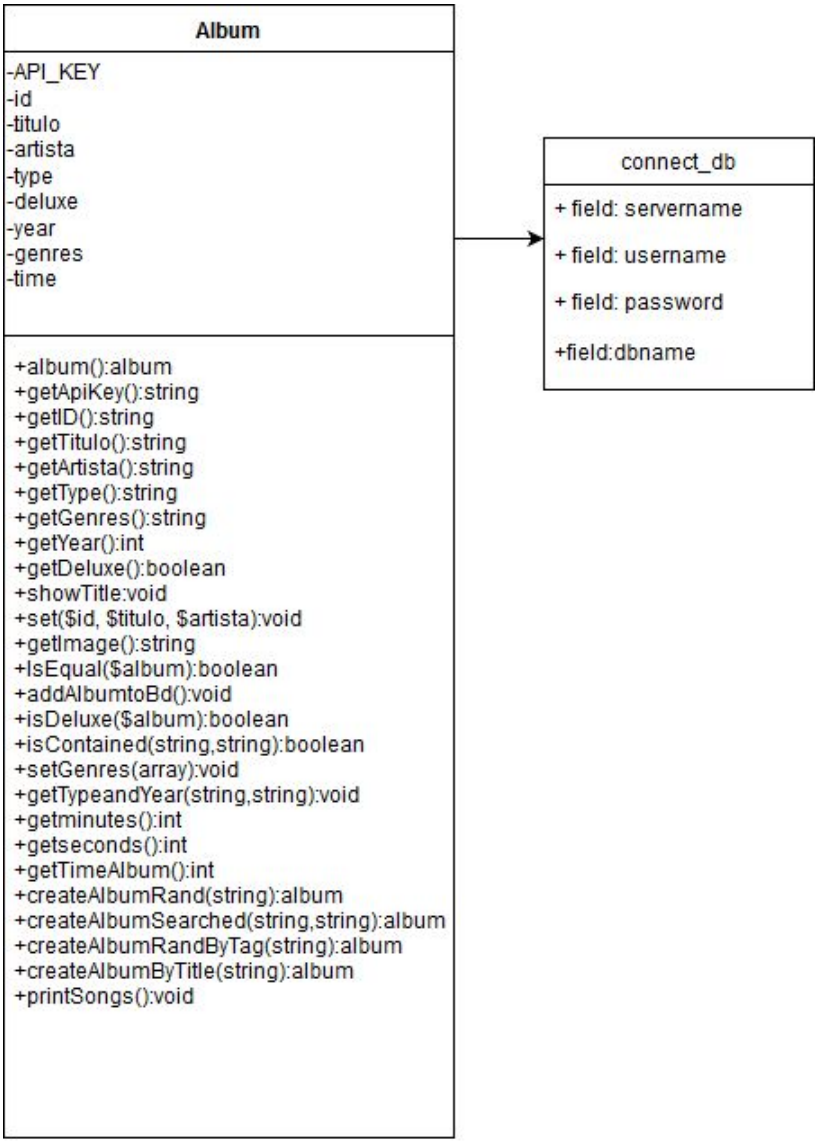
showRelatedArtist(string artist) will also call a PHP or Javascript function that, thanks to the API, is able to select all related artists and will show thirteen of them.

search(string key) will help the user search the entire database so it will have to access the *Song, Album* and *Concert* tables.


USER_IDENTIFIED
shopwLateral(string option)
goToDetailledCollection(string option)
showCover(string style,string email)
showRecomendationSong(string email)
showRecomendationStyle(string email)
close(string email,string nameAlbum, string artist)
showInfoAlbum(string email,string nameAlbum,string artist)
showmylistAlbum(string email)
showmyConcerts(string email)
showRecommendationConcerts(string email)
showCountries(string artist,string continent)
showCities(string artist,string country)
showRelatedArtist(string artist)
showNearConcerts(string email)
showNearConcerts(string email)
showmyAlbumsCreated(string email)
showGroupsCreated(string email)
search(string key)
getAlbum()
getCover()
getSongs()
getInfoAlbum()
getConcert()
getCountries()
getCity()
getNearConcert()

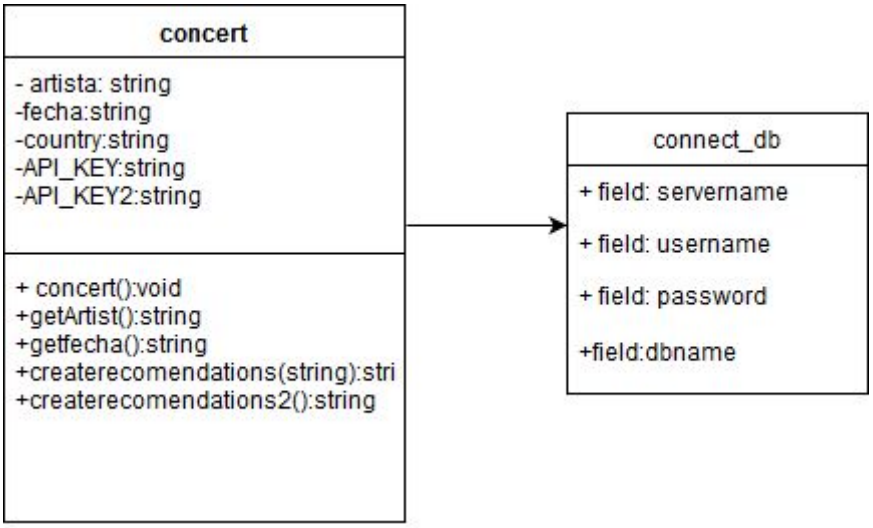
CLASS DIAGRAM

The specifications for our web app have made us create several classes to manage the data. In our web app, we can find two main classes which help us manage the data easily. First, we will explain Album class, the first to be created in our system. We use this class to show most of the information of an album in this project.



As we can see, there are not many relations between the different classes, this is because of the great amount of information we get from the APIs. For this reason, the information saved in the cache is relatively low.

To manage concerts available concerts, we use a class as well. In this case, it happens the same as with albums and we get a lot of information from the API.



A lot of the information provided by the APIs is also stored in the database to be used in the future.

EXTERNAL APIs USE/ OURSELF-API

In our web application, we use several APIs for the correct functioning of our system. For the development of this project it is essential the use of some, in this document, I will cite those that we have used throughout the development:

MusicBrainz: At the beginning it was one of the solutions that we found best for the implementation of our web application although we could see that it was not updated and that some features were not entirely correct, so we stopped using it and have only kept it to collect the information about the type of album that we are looking for and about its year of publication, this gives us a very valuable information to show the users.

LastFM: this is our main API, it is powerful and also shows all the necessary information like the songs that make up an album, this has made us decide to trust this provider in front of others, besides it is a powerful search engine that is able to search not only by artist or album but also by tags, in which we can specify anything that the album may have.

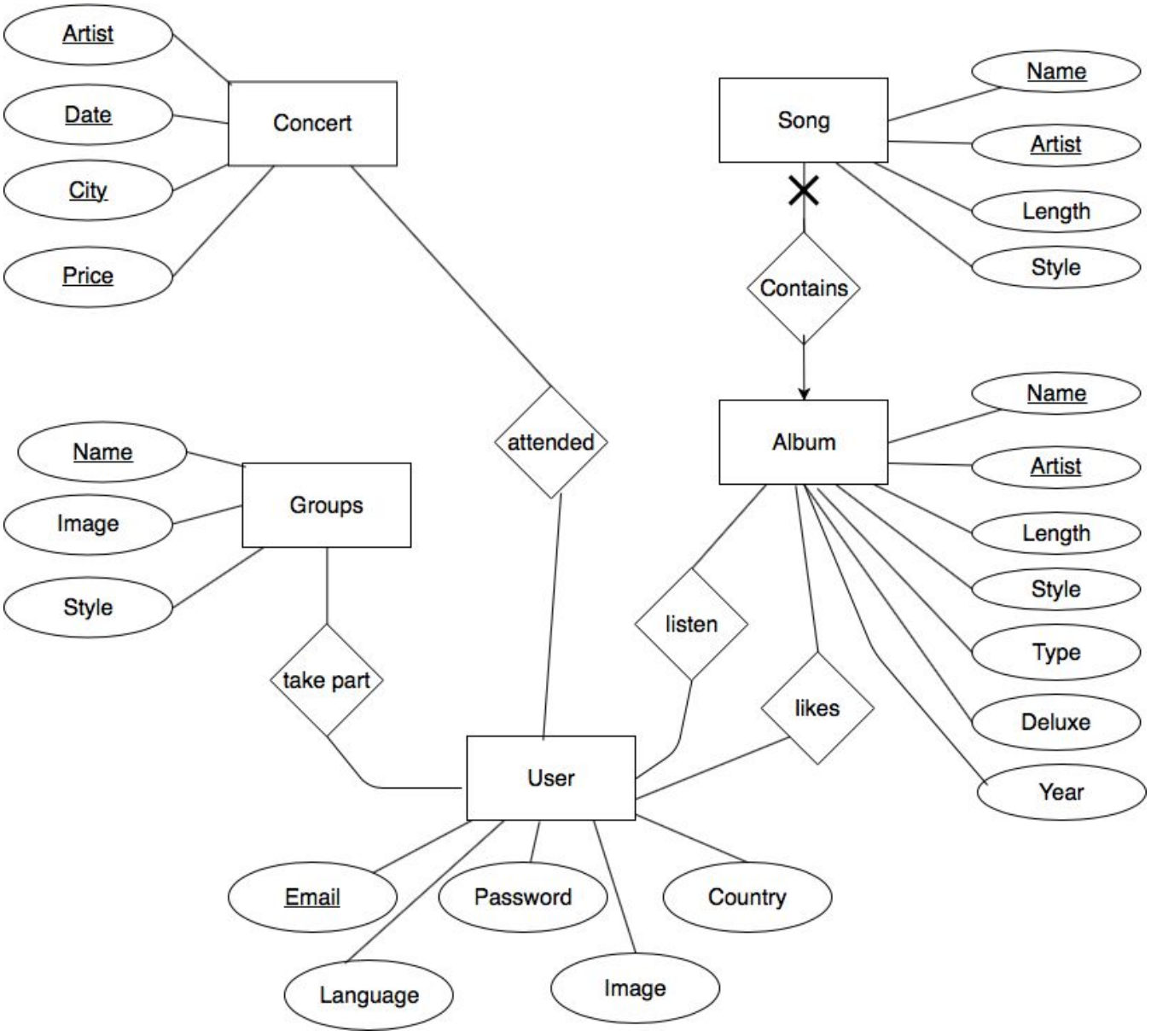
Bing: this API is used to search and display images of the Internet, its main and primary use is to show the cities in the view of the concerts offered by an artist.

Ticketmaster: this is one of the most complete APIs to get information about available concerts. Thanks to this API we can show information such as price, poster, country...

Google Maps: in our web app, we use this API to look for the coordinates of a city or a country. These coordinates are used by the Ticketmaster API to look for concerts in that area.

The development of our API was done using PHP defining the GET, POST, PUT and DELETE methods to create a useful REST API. These methods are able of extracting, inserting, modifying and deleting data from our database. To check its correct performance we used the Insomnia plugin.

2. Scheme of the database



This entity-relation diagram shows our database and its tables, attributes and their relations:

- **Song** has as primary keys *Name* and *Artist* because there cannot be a song with the same name and artist. Other attributes are *Style* and *Length*, which can't be null or zero.
- **Album** has as primary keys *Name* and *Artist,* because of the same reason as **Song**. Other attributes are *Length*, *Style*, *Type*, *Year* and *Deluxe*.

- **Concert** has *Artist, Date* and *City* as primary keys while the last attribute is Price.
- **Groups** has *Name* as the primary key while the other attributes are *Image* and *Style*.
- **User** has *Email* as the primary key and the other attributes are *Password, Country, Language* and *Image*. In addition to these attributes, **Song** has relations with **Concert**, **Groups** and **Album.**

  This database will be updated using PHP as explained before.

3. User relation with Orpheus

- Unidentified malicious users: their IP will be added to a list and will block him from connecting again to our website.
- Identified malicious users: we will ban their IP and email so they won't be able to connect or register using the same email.
- Unidentified well-meaning users: we will check their IP and, if it's not on the banned IPs' list, they will be able to register.
- Identified well-intended users: their anonymity will be assured and will have access to all of our functionalities.

4. Progress & conclusion

The progress in this project is being done in a very collaborative way because we are two people and we have been working together in many aspects of it. Our plan is to keep it like this and work together for the rest of the project.

As managers and developers of MuCr, we have changed some of the interfaces throughout the development of the web application, this is undoubtedly one of the procedures that happen when the front and then the back part first develops. In addition to being a project of the university we have to follow some guidelines, which are difficult to achieve according to the method that is applied. Due to this and the limited time that we have, we have

decided to comply with the requirements that are required, by changing the interface. The changes are constant, and varied along the MuCr principle:

Updates carried out on May 22, 2018:

We have appreciated that people do not add simple songs, but just albums, in the specifications that we gave at the beginning the user could be able to add songs, this has disappeared in this update due to the little use that it would obtain, and limited time, maybe in a future update the specification is included again, due to this ambiguity we will allow our structure of the database to be conserved, without producing any change.

Updates carried out on May 20, 2018:

The first idea for the search of nearby concerts was the development of a locator, but this did not seem right to maintain the privacy of users, we have decided that users should voluntarily enter the location they want, being possible the search of the same city in which it is located, or the country.

Updates carried out on May 24, 2018:

The user can only add albums to the list he likes or to the listener list, only in the window in which the album is explained in detail.

Updates carried out on May 26, 2018:

In the specifications that we had at the beginning, in which we explained how there was a search engine that looked for everything in our system, that has been changed and is directed to the specifications in which the implementation of a search engine by the artist is required, which has been added, in the sidebar.

Updates carried out on June 1, 2018:

To complete the update about the artist search engine, we created a detailed artist page.

Updates carried out on June 6, 2018:

Due to the use of our database and the possibility that any user can put misinformation in the database and this can lead to conflicts, as developers and heads of the application we have decided to add two more tables to our database, that will be albums_NV and concerts_NV in which users can upload concerts and albums that are not in the web application, in the initial plan the selection and correct decision of these, will be done by ourselves. The plan for the future is for an AI to be able to properly catalogue this information.

Tasks

Jesús	José Manuel	Both
- view-controller of the nearby concerts - the recommendation of the concerts - the detailed concerts - the collection of artists shown according to a tag - the search by artists - the development of the API of MuCr	- view-controller of adjustments - the section so that the users can add new albums and concerts - the detailed album of an artist - the related artist of another given artist - the recommendation of concerts	- the main page - the view of the albums that I have listened to and that I have liked - adjustments - the documentation