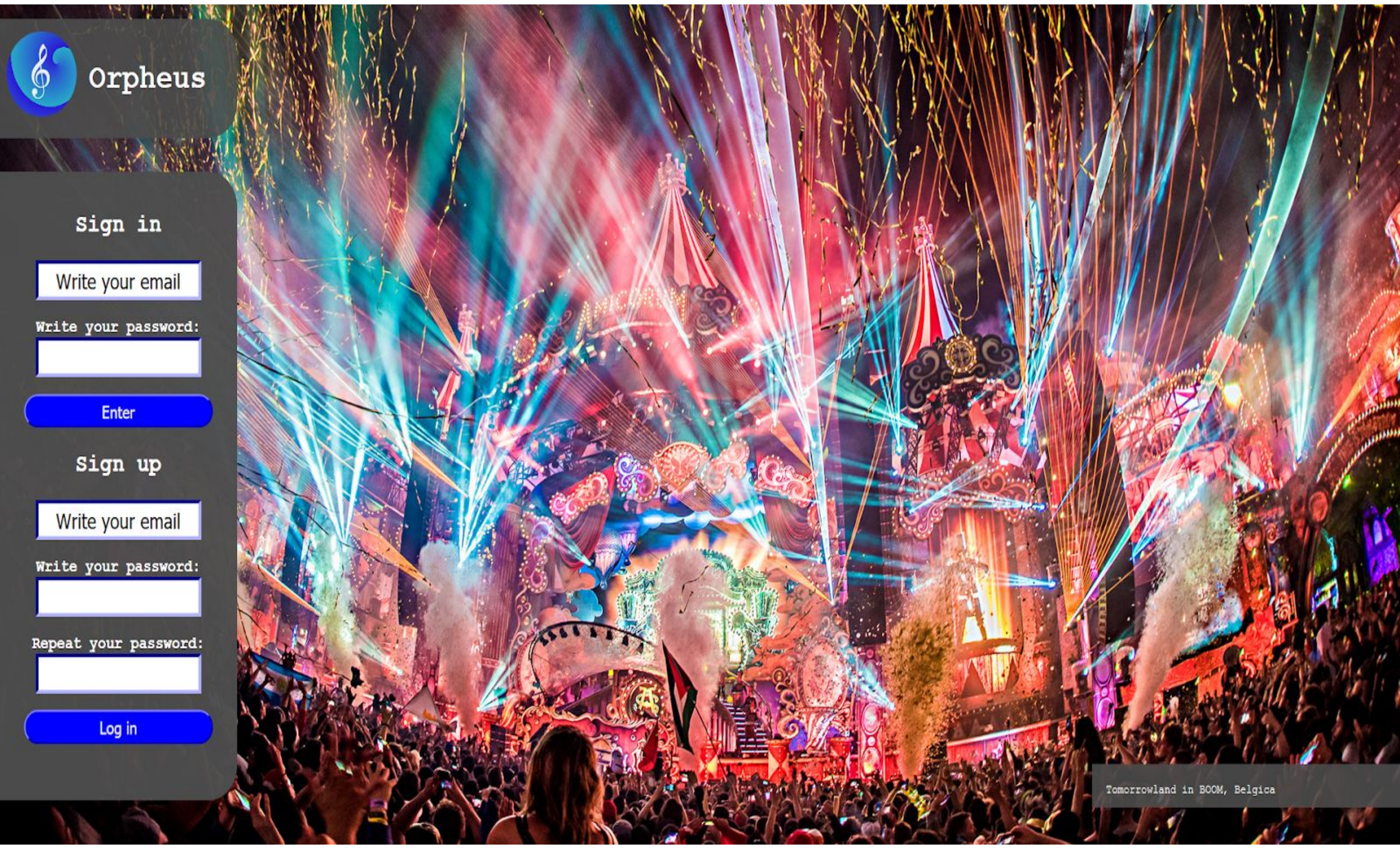
ARCHITECTURE OF ORPHEUS

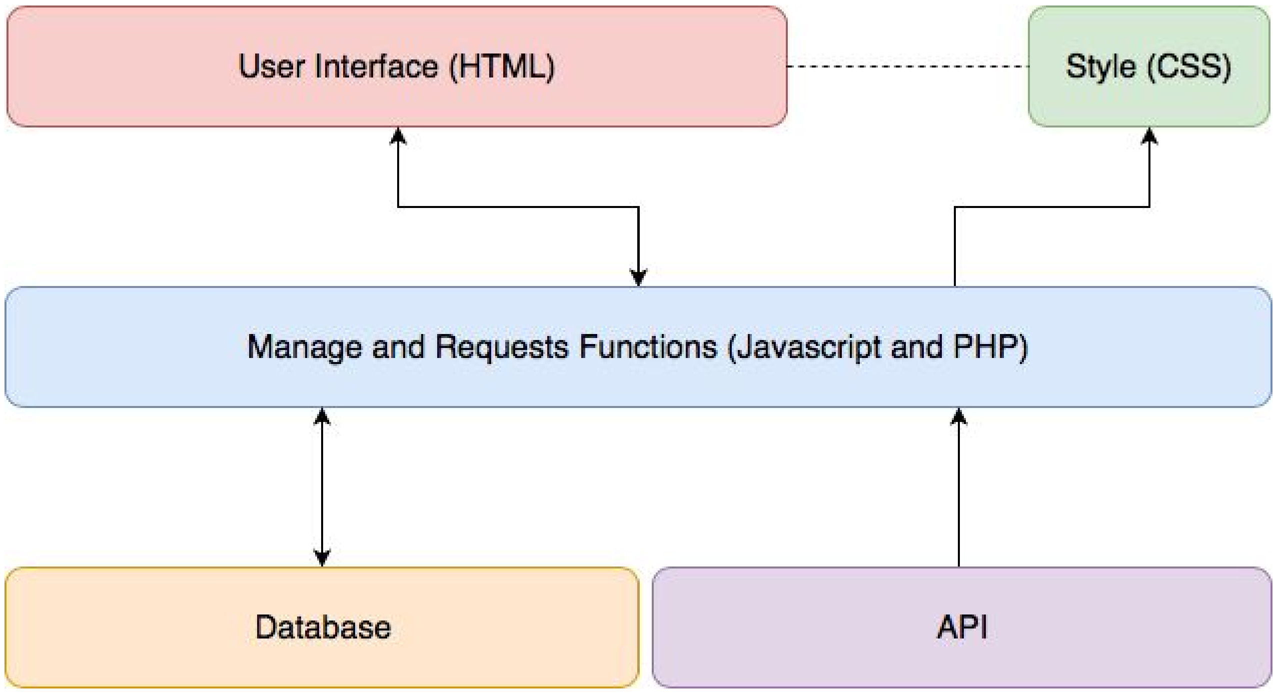


José Manuel Rodríguez Calvo

Jesús Jiménez Sánchez

Table of contents

1. Plan for architecture

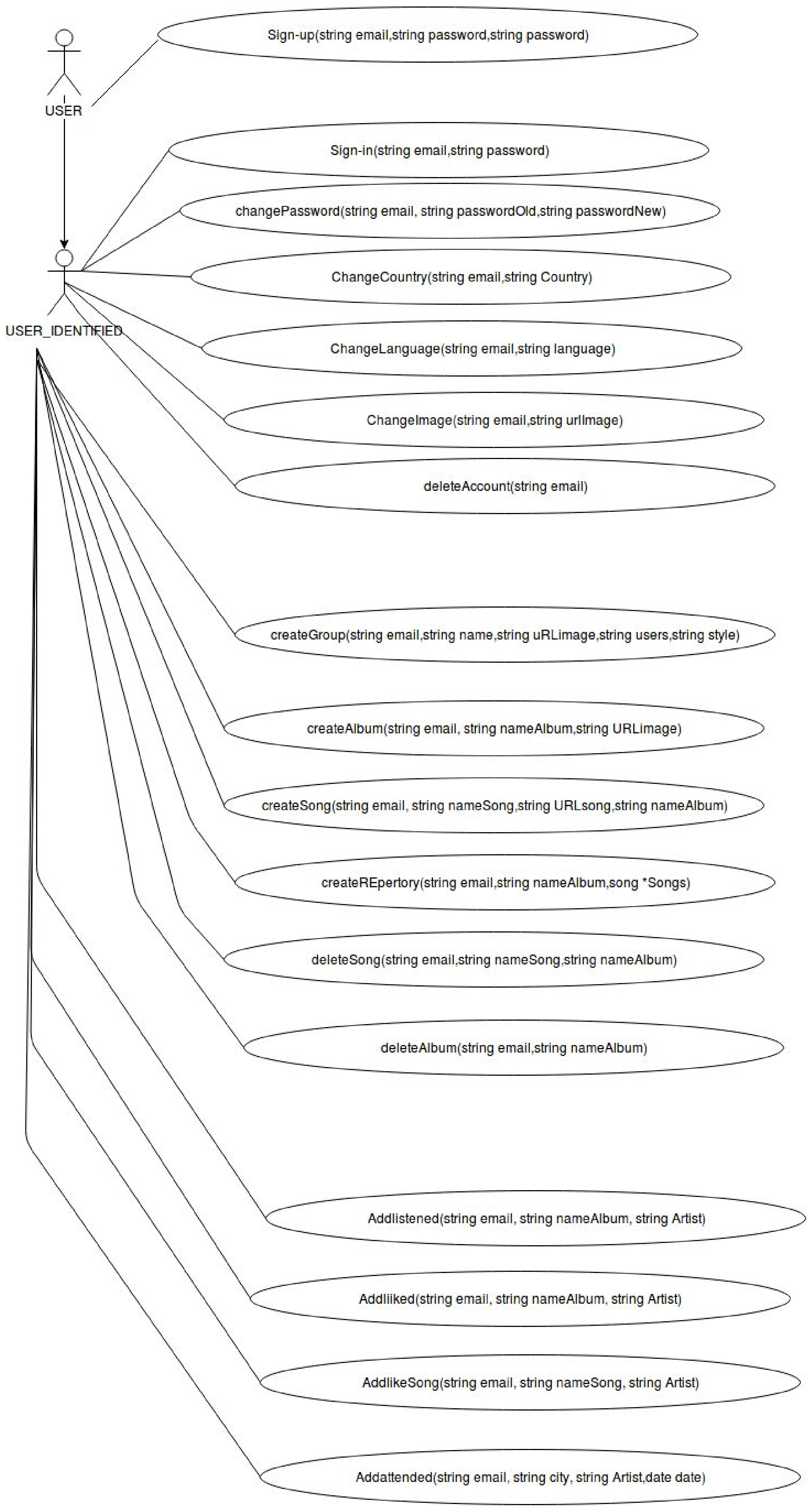


In this diagram, we show the behaviour of our web app and how it interacts with the user sending the right information in an organised way.

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.

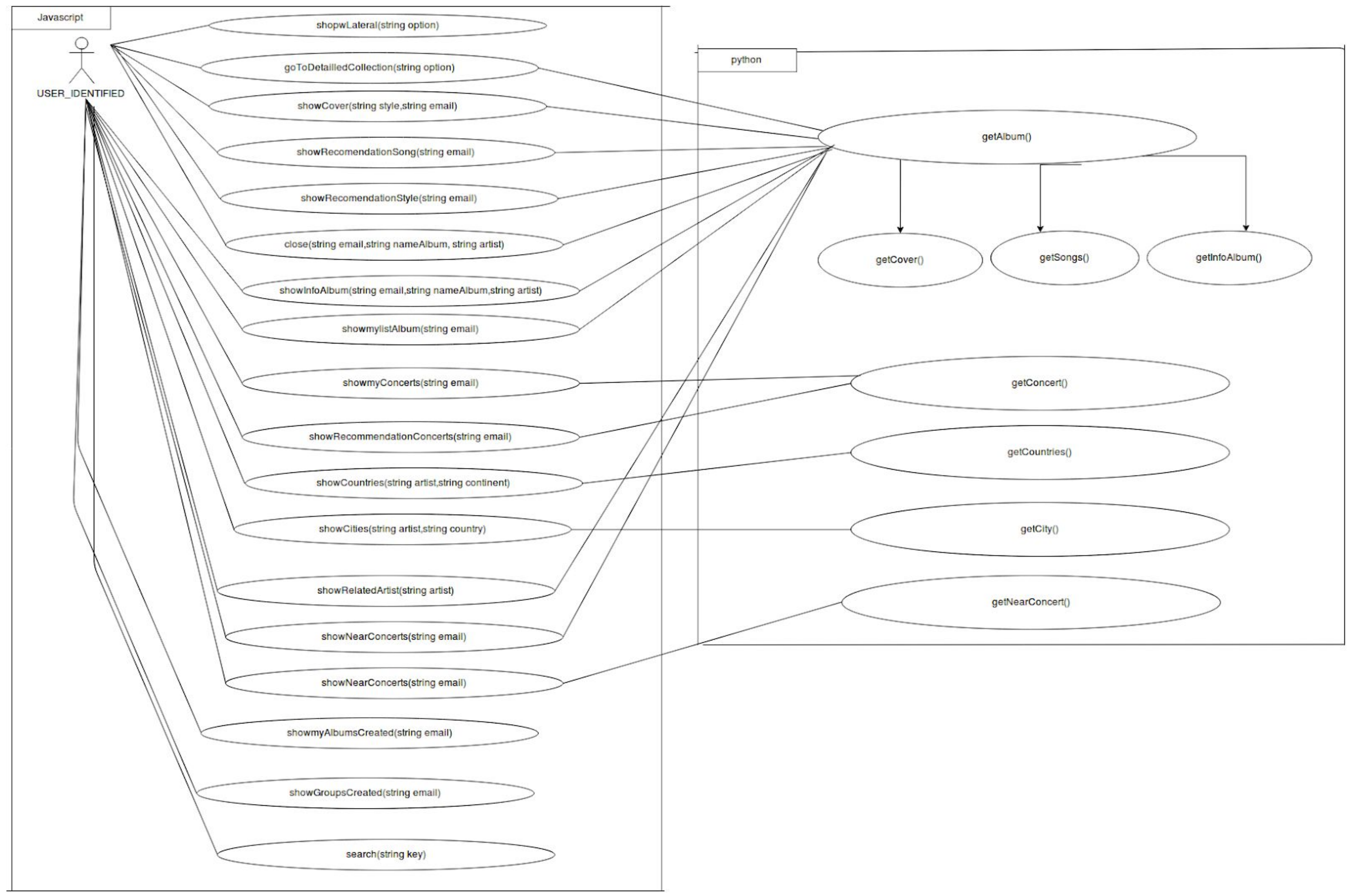
In the next diagram we show the functions we use in Javascript, these functions are able to read the database and show the obtained information in the web app, with the help of the API in Javascript and PHP. Here are some examples:

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.


Javascript
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)
python
getAlbum()
getCover()
getSongs()
getInfoAlbum()
getConcert()
getCountries()
getCity()
getNearConcert()

APIs USE

Our web app uses some APIs for some of the tasks that have to be done. Some of these tasks are managing the countries, the albums, the songs and the concerts.

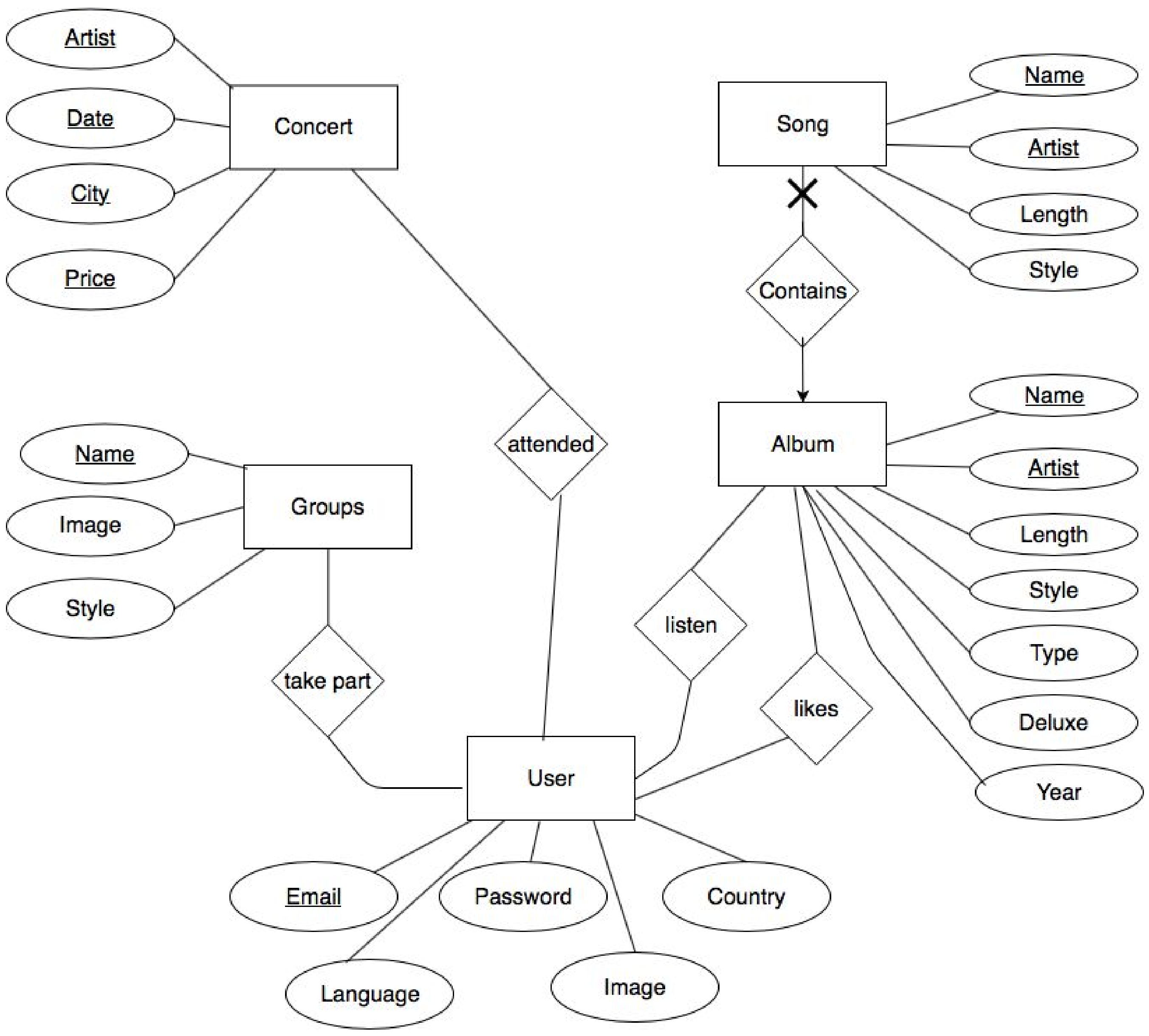
The countries have to be managed through the API for using images of the city, the country's flag, its initials and the continent where it belongs in a more beautiful concert view.

The API used for the albums and the songs will be in charge of getting the songs belonging to an album and the information of this album.

The API in charge of the concerts will do a similar job than the songs' and albums' API, it will get the details of the concerts. It will be also able to notify the user well in advance of the nearer concerts.

The creation of the API would be solved using the REST technology, this will be implemented using PHP or Javascript and it will be one of the last parts to be done in the project.

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 Javascript and Python 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.