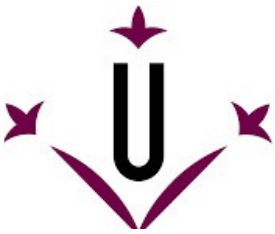


Universitat de Lleida
Escola Politècnica Superior

# Deliverable 1

**Names and Surnames: Armand Ciutat Camps, Joel Aumedes Serrano, Joel Farré Cortés, Marc Cervera Rosell, Moises Bernaus Lechosa, Roger Castellví Rubinat**

**DNI's: 48056684R, 48051307Y, 36984053R, 47980320C, 47903568L, 48056998Q**

**Data: 10 – 04 – 2021**

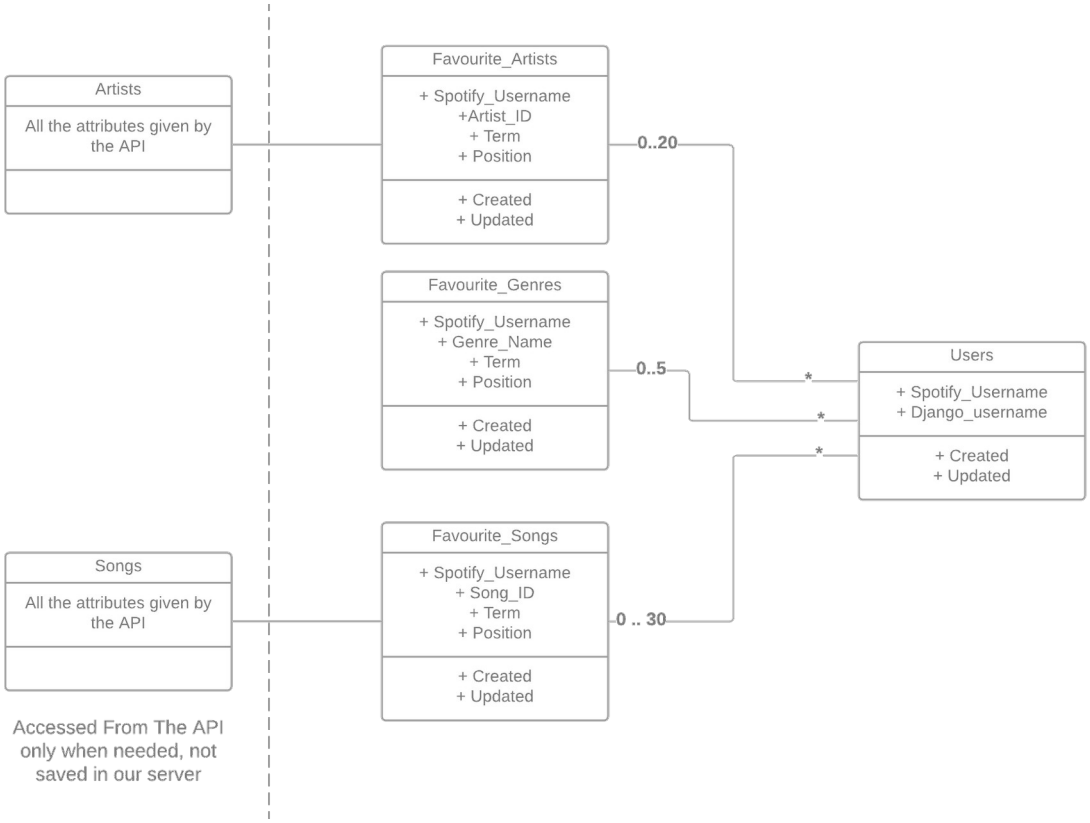
**Subject: Web Project**

**Curs: 2020 – 2021**

To develop this deliverable, we split our work in 2 branches; the development branch, in which we implemented everything related to the MVC architecture and tested via *python3 manage.py runserver* in our own computers, and the deploy branch, which was the main branch, to try and deploy the website to the world. This is also useful to avoid accidentally deploying not correct code to the web by splitting the code that is correct and the code that is being worked on.

The deploying we have used is Heroku. By using a Procfile and a requirements.txt file, we can push our changes to Heroku, which will automatically deploy our Django project using gunicorn on [web-project-galajat-2021.herokuapp.com](). By splitting our work in two branches, we can assure the web that is online is working perfectly and with the Debug flag set to False to avoid security issues.

We modified the data model that was presented in the project proposal to fit better what we wanted our website to be and to be more accurate:



The files in the Project Folder were barely modified. The vast majority of developing was done in the web interface app we created, called ‘webspotify’. The most important modification is in the project’s *urls.py* file, in which it delegates URL responses to a *urls.py* file in the app, and in the *settings.py* file, in which we changed the logging details to detect errors easily.

We used this app to implement our web interface. The Django Models that were implemented in this app’s *models.py* file and saved in our project are:

- The **users**, whose most important attributes are its Spotify Username and its username used to register in the web, and to build relations between tables. An object’s ID, be it a user, an artist or a song, will use the same ID that it’s referenced with inside the Spotify API, to facilitate the communication between our web and said API.
- The user's **favourite artists** in particular, which will contain the artist_id, its position and the term. The latter two attributes are used to save in which position is the object and in which time frame (the last 4 weeks, the last 6 months and of all time), and will be shared by other models such as those mentioned next.
- The **favourite musical genre** which consists of the name of the genre among other attributes mentioned previously.
- The **favourite song**, which consists of its id, again in addition to those shared attributes mentioned above.

The favourite artists, favourite songs and favourite musical genres are all linked to the user through its ID, and also have created and updated fields to know when were they created or last modified.

All models created in the *models.py* file are imported into the *admin.py* file and are registered on the admin site.

The *views.py* file maps the links of which the website is composed of to the template that must be rendered as an answer to the user’s request. Inside of the document we have the functions used to respond to the URLs of: main, dashboard, and register. Within the directory 'templates' we find the templates to be rendered separated in different directories. The registration directory has all the templates that the user needs to login, register, change its password, or log out. The webspotify directory contains the *index.html* file, which is the main URL and the page a non-logged in user sees, and the *dashboard.html* file, which is the URL a logged-in user sees. We managed to control this flow by changing the LOGIN_REDIRECT_URL and LOGOUT_REDIRECT_URL variables in the project’s *settings.py* file.

To deploy the web, simply push the repository to a Heroku app, add its hostname to the *ALLOWED_HOSTS* variable in *settings.py*, and the app should be up. The *db.sqlite3* file has some created users to test out the app, but not many relations in the other Models since the logic to obtain data from the API is not yet built.