# Implementation of a big data system with Spotify Web API: relevant 'Covid-19' Playlists

Francesco Battista - 214224, Irene Parolin - 211591

January 2021

## 1 Introduction

The main objective of our project is to design and implement a big data system that retrieve Spotify data for analyze the spirit of songs that are relevant to "Covid-19" tag.

Spotify, like many other companies, releases the possibility to have access to its data through Web API. This is our starter point for implementing a big data system with which we can retrieve the Audio Features of all that songs that are in a playlist labeled with word 'Covid'.

The available information about playlist's owner did not allow us to analyze differences and similarities between countries ( we do not have owner's country data), therefore we can not know how are the Audio Features of songs in each European state but we analyze the differences/similarities between these last months marked by health emergency.

This report is organized with a first section where we describe the all the architecture of our systems and that we used to retrieve and store data. The following section is about how we have analyze data and visualize results found, and then the final conclusions of the project. This report does not discuss in detail how the code works or how it is structured. A better explanation can be found in GitHub's REDME.md.

## 2 Methodology

We present the technologies used to create this Big Data System that allow the analysis of the data extracted from the official Spotify API through Spotipy, the Python library that allows interaction with it.

### 2.1 Docker

Docker allows developers to be free to build, manage and secure applications without worrying about infrastructure where they are installed. For this project we are using a version of Docker Desktop for Linux. We chose to use Docker

Swarm to manage and deploy containers. According to Docker the cluster management is grafted into the Docker Engine built using Swarmkit. Swarmkit is a separate project that implements a layer that orchestrates Docker and is used directly within it.

For this project, Docker is perfect for simulating a horizontally scalable system. In case of need it will be possible to add a container and connect it to the same network to have one more element to be used.

## 2.2 PostgresSQL

PostgresSQL is an open source DBMS which has been used inside a dedicated docker container. Even knowing that connecting a database directly to the data source is a choice that makes the architecture weaker, we have chosen to implement this strategy since the data source was not multi-source but unique and standardized. In accordance with the principles of systems architecture and with the awareness described above, it was decided to create 3 database layers:

- Postgres: the first database used. It is the database where some of the filtered data that is loaded by the API is loaded.
- Postgres_backup: second Postgres_backup is the backup database, this is taken into account in case the first fails or falls;
- Postgres_dwh:third Postgres_dwh is the data warehouse that contains the data after carrying out the ETL operations with Spark.

## 2.3 PgAdmin

It is possible to access these 3 databases using the graphical client PgAdmin. It is initialized through commands and once inside you can query the databases and the tables inside them. The choice of PgAdmin is purely personal. In fact, in the stack.yml file it is possible to add another container with another client called Admire by uncommenting the relevant section.

## 2.4 Redis

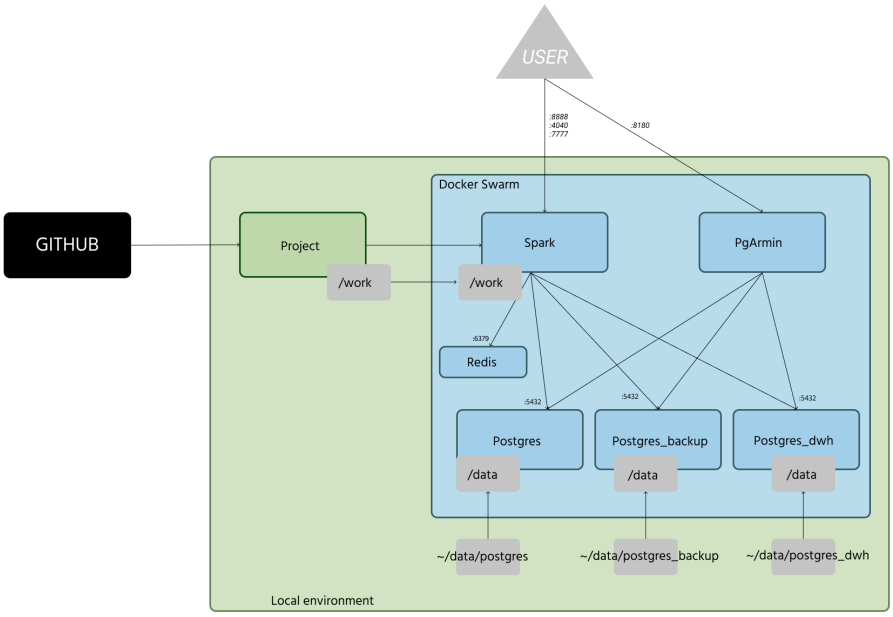
Redis is an in memory key-value store. It was used to manage data updates. It stores the ids of the playlists inserted in the main database with an indexed value of 1. Once the pipeline is started, a cronjob is launched which works in the background once every hour. This searches on the spot for playlists relevant to the Covid and compares them with those saved in Redis. If a playlist is not inside Redis then it is inserted in the main database in order to have the data always updated.

## 2.5 Spark

Apache Spark is an open source framework for distributed computing. It is an integral part of the main container and allows you to launch an instance of

Spark on the node. Only one container has been initialized with Spark so it opens the session and closes on a single Docker container. It was used to take data from the main database and carry out various transformations in order to create 3 tables that could be saved within the data warehouse. You can see the statistics of Spark and how the various calculations were made at localhost: 4040 once finished launching the data upload pipeline.

## 2.6 Workflow




Once the containers have been instantiated and entered into the main one (jupyter _spark), the environment is bootstrapped where the main libraries are installed and the cronjob is initialized once an hour. Once this is done it is possible to use two pipelines.

- load data api: this pipeline loads data directly from the api
- load data backup: loads data from a backup file, updated to the latest instance of the pipeline.

Even if the data loading changes, the two pipelines follow the same workflow:

- initialization of the tables within the 3 postgres servers,
- load data,

- backup of the main database inside the backup one, copy the data into a csv file that will be used to load the files into the backup pipeline,
- ETL operations on main data and copy of data in the data warehouse,
- data visualization via Web App generated with Dash.

Once the pipeline is finished, you can go to the address 127.0.0.1:7777 to view the dashboard.

# 3 Data description

We retrieve with a series of GET operations all data of playlists that are tagged with label 'Covid', both in the title of playlist and in description of this. In this way we have almost 2000 playlists in our database. Since there is not the exact information of the creation of the playlist, but having the information on the insertion of the single tracks, it was decided to take into consideration the month with the greatest number of songs added. A bit of a surprise, we found that there are almost a hundred of playlists that have years prior to 2020 as their creation date. This is due to subsequent changes in the title or description of the playlists in which they have inserted the word 'Covid' and therefore linked to our reference tag. For the subsequent analysis, we have decided not to take these playlists into consideration and focus on the playlists created since January 2020. What represents our greater interest is the analysis on the Audio Features to discover the spirit of songs: among all we have chosen energy, valence and danceability. For each playlist we made the average of these three variables, and then we calculated the average of the playlists according to the month of creation. We also calculated the standard deviation for each Feature, with which we can investigate the variability within the playlists and therefore understand if the songs belongs to the same genre or not.

## 3.1 Visualization

From there we made our visualisation in the dashboard using dash and plotly libraries, and that represents our final product.

We first plotted the distribution of playlists in the last few months, and a barplot in which we grouped the number of tracks in the playlists into 6 categories. Range of tracks goes from one single song until nearly 10000 songs.

We have also created line graphs to display the average trend for each Features and them standard deviation. Thanks to the graphs we can see if there have been important differences over the months: this has not happened because the values of both the mean and the standard deviation have not suffered large changes in this months of health crisis.

# 4 Conclusions

In conclusion, this project uses technologies that allow to scale horizontally a project of this size but also to improve it with a few steps in the event of a growing amount of data. The aim of this project was to bring together these types of technologies by considering the best way they could coexist. One of the improvements that can be made is the use of Redis not only in the update phase but also in the data upload phase, with multiple containers managing the download it can easily be used to divide playlists for each container. Spark can be used with multiple containers facilitating ETL operations and better managing resources.

As results we can assert that there are big differences in the month of creation of the playlists, noting a peak on December, while there are no wide variations in the spirit of the songs: in these 12 months the variations for the averages of the Audio Features (energy, valence and danceability) are minimal, so we can say that the spirit of the songs has not undergone major changes by time.