EXTRACTION, TRANSFORMATION, LOADING AND VISUALIZATION OF COMBINED TWITTER AND SPOTIFY DATA IN A SCALABLE ARCHITECTURE

Thesis of the Master's degree in Business Intelligence and Big Data in Cyber-Secure Environments

July 2022

Adrián Riesco Valbuena

Tutor: Álvar Arnaiz González

Table of contents

- Introduction
- Objectives
- Techniques and tools
- Relevant aspects
- Conclusion and future work lines

INTRODUCTION

Introduction

Objective: Extraction, Transformation and Loading process to capture and visualize data from Twitter and Spotify APIs.

Using: Software tools from the Big Data domain and with a life cycle driven by agile methodologies.

Highlights:

- Service containerization.
- Data flow orchestration.
- ETL processing (data search, cleaning, merging and loading).
- Data storage.
- End user visualizations.

OBJECTIVES

Main objectives

- Ability to obtain data in real time.
- Combination of at least two different data sources.
- Potential to scale in both technology and data volume.
- Involvement of various technologies in the Big Data field.
- Use of open source tools
- Design and implementation of two visualizations for the end user.

Functional requirements

FR1 Data must be obtained from the Twitter hashtag #NowPlaying every 30 minutes.

FR2 There must be at least two different visualizations and one of them must provide the ability to view all of the stored data.

FR3 At least one of the visualizations must show last songs name, artist and audio features.

FR4 At least one of the visualizations must have a link to the source tweet.

FR5 At least one of the visualizations must have the ability to compare different metrics.

FR6 At least one of the visualizations must combine two different types of visualizations.

FR7 Both visualizations must provide sorting capabilities.

FR8 Both visualizations must be responsive to different screen sizes.

Technical requirements

TR1 The development must have the ability to be deployed in different environments with minimum effort.

TR2 The data flow must be automated, with the entire process orchestrated by a single tool.

TR3 The execution of the ETL process must be done with a tool that can scale and run in distributed environments.

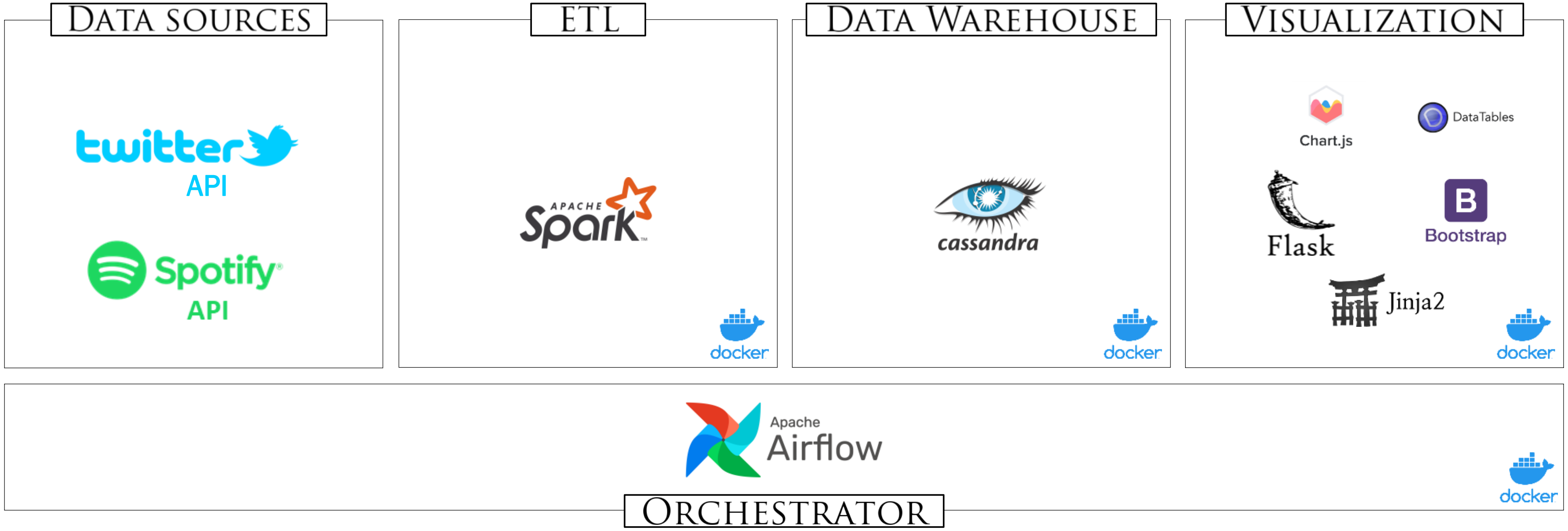
TR4 The data warehouse must have the ability to escalate in terms of a Big Data problem.

TR5 The web application must be designed with widely recognized tools.

TR6 All the tools used must be open source.

TECHNIQUES AND TOOLS

Techniques and tools



DATA SOURCES

twitter API

Spotify API

ETL

APACHE Spark

docker

DATA WAREHOUSE

cassandra

docker

VISUALIZATION

Chart.js

DataTables

Flask

Bootstrap

Jinja2

docker

Apache Airflow

docker

ORCHESTRATOR

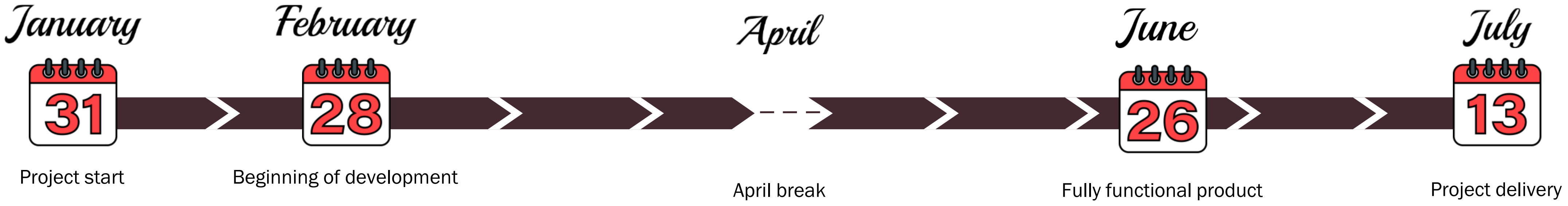
RELEVANT ASPECTS

Project plan

- ➢ Agile methodology with sprints of two weeks and POV and backlog refinement meetings.
- ➢ Stored in GitHub.

Sprint == Milestone.

Tasks == Issues.



January 31 – Project start

February 28 – Beginning of development

April – April break

June 26 – Fully functional product

July 13 – Project delivery

Data Twitter

Endpoint: Recent search

- **id**. Tweet id (integer), useful uniquely identify the tweet.
- **text**. Tweet text, useful to identify the song played.
- **entities**. Useful to clean the text and remove hashtasg, cashtags, mentions and urls.
- **created_at**. Tweet creation date.

Data Spotify

Endpoint: Search for Item

- id.
- name.
- popularity.
- artists' id.
- artists' name.

Endpoint: Get Tracks' Audio Features

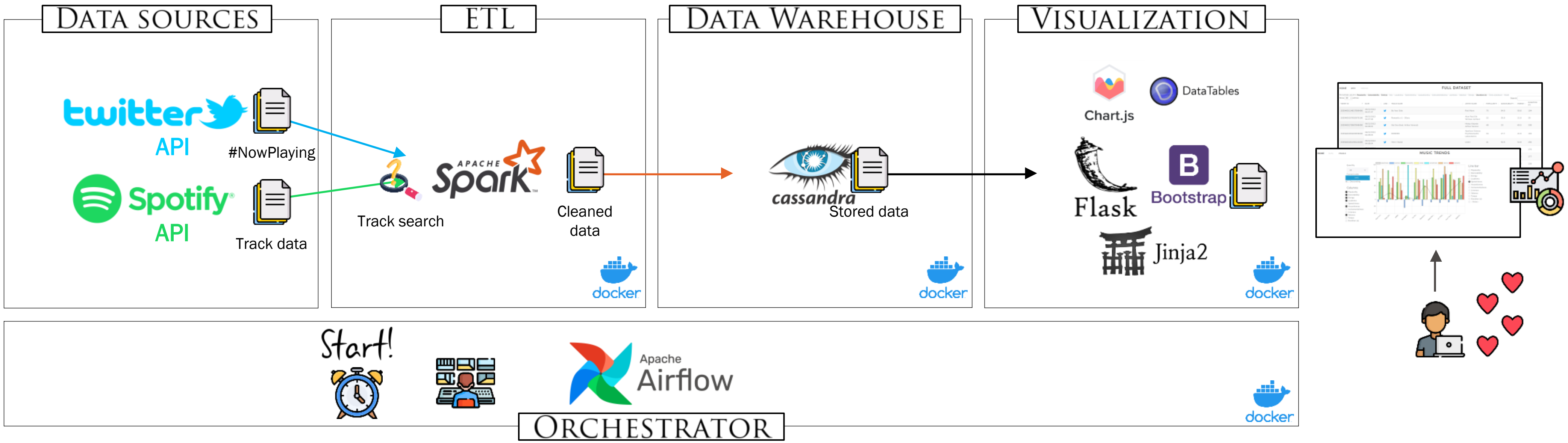
- id.
- danceability.
- energy.
- key.
- loudness.
- mode.
- speechiness.
- acousticness.
- instrumentalness.
- liveness.
- valence.
- tempo.
- duration_ms.
- time_signature.

twitter Spotify

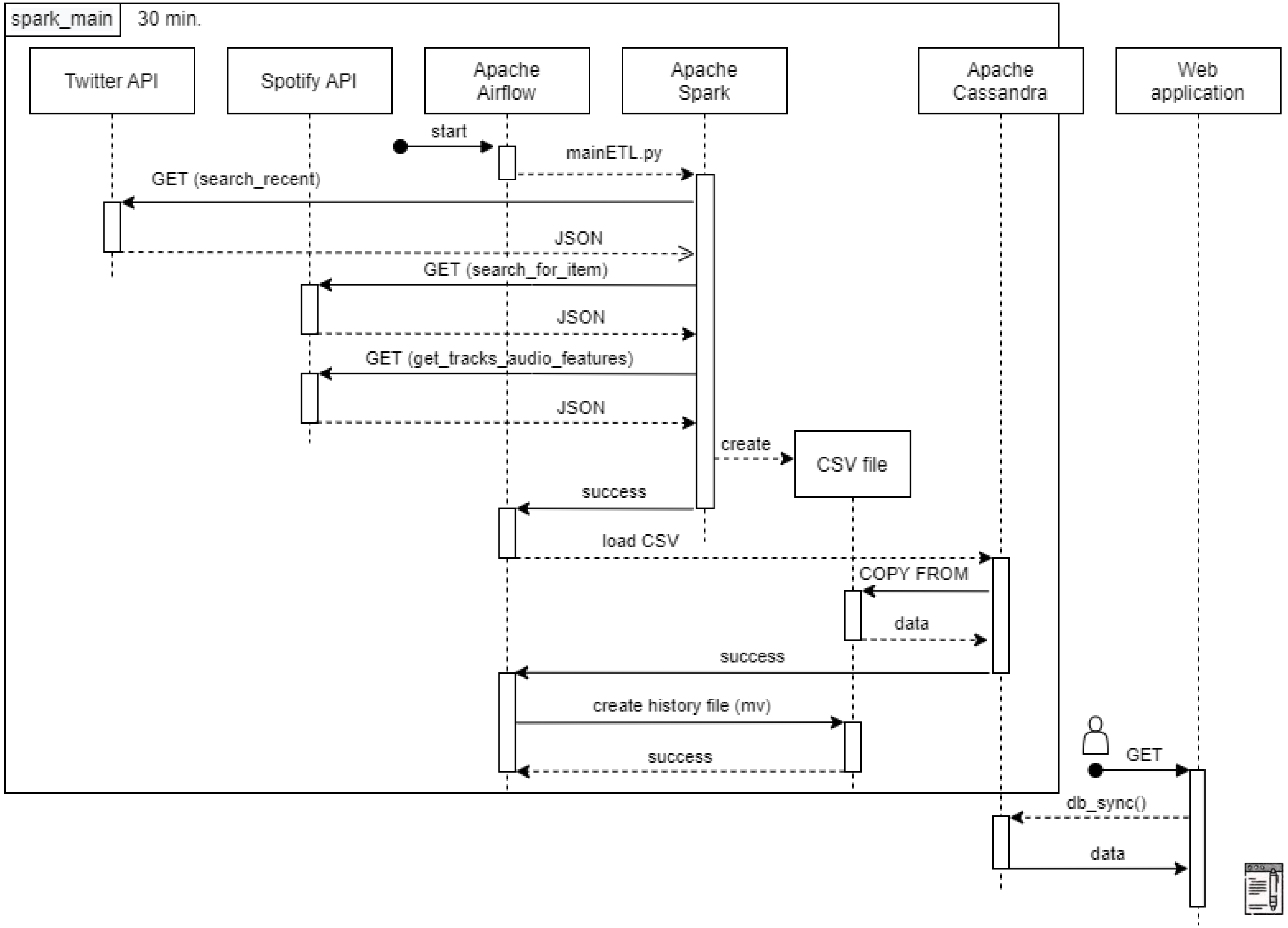
Cleaned data

🐦	id_tweet	bigint	🟢	artists_name	text	🟢	Instrumentalness	float
🐦	text	text	🟢	danceability	float	🟢	liveness	float
🐦	created_at	timestamp	🟢	energy	float	🟢	valence	float
🐦	url_tweet	text	🟢	key	int	🟢	tempo	float
🟢	id_track	text	🟢	loudness	float	🟢	duration_ms	int
🟢	name	text	🟢	mode	float	🟢	time_signatura	float
🟢	popularity	int	🟢	speechiness	float			
🟢	artists_id	text	🟢	Acousticness	float			

Data flow


Data sources
twitter
API
#NowPlaying
Spotify
API
Track data
ETL
Track search
Apache Spark
Cleaned data
docker
Data Warehouse
cassandra
Stored data
docker
Visualization
Chart.js
DataTables
Flask
Bootstrap
Jinja2
docker
FULL DATASET
MUSIC TRENDS
Start!
Apache Airflow
Orchestrator
docker

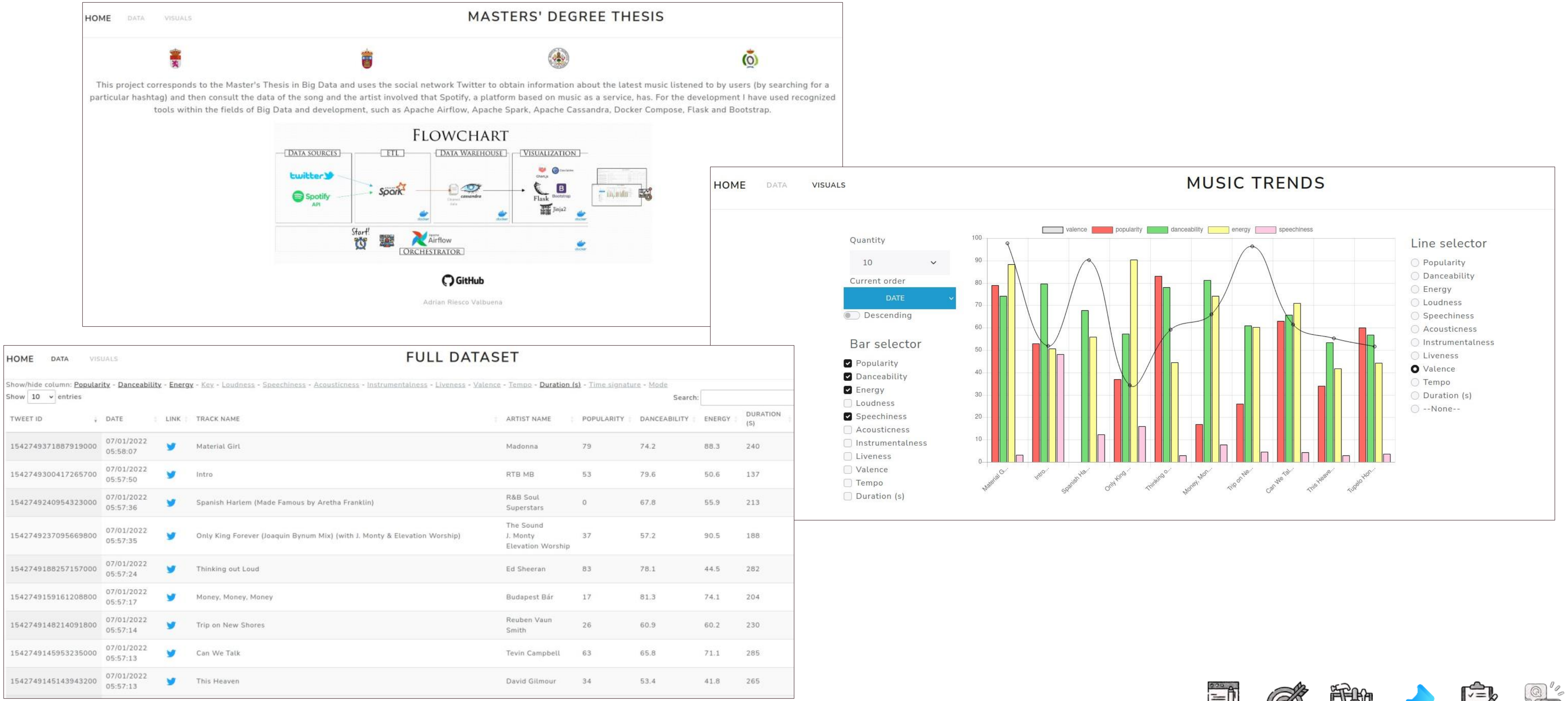
Data flow


spark_main
30 min.
Twitter API
Spotify API
Apache Airflow
Apache Spark
Apache Cassandra
Web application
start
mainETL.py
GET (search_recent)
JSON
GET (search_for_item)
JSON
GET (get_tracks_audio_features)
JSON
create
CSV file
success
load CSV
COPY FROM
data
success
create history file (mv)
success
GET
db_sync()
data

Implementation highlights and challenges

- **Developer keys.** Twitter and Spotify APIs required developer keys.
- **APIs' rate limits.** Maximum of 500.000 tweets per month.
- **Airflow operators.** There is no specific operator for Apache Cassandra.
- **Airflow and Spark connection.** Environmental variable to set connection.
- **Database schema configuration at launch.** Additional container to create schema.
- **Data representation.** Use of Datatables and Chart.js.
- **Mismatched tracks.** Several languages and only the first result collected from Spotify.

User interface



MASTERS' DEGREE THESIS

This project corresponds to the Master's Thesis in Big Data and uses the social network Twitter to obtain information about the latest music listened to by users (by searching for a particular hashtag) and then consult the data of the song and the artist involved that Spotify, a platform based on music as a service, has. For the development I have used recognized tools within the fields of Big Data and development, such as Apache Airflow, Apache Spark, Apache Cassandra, Docker Compose, Flask and Bootstrap.

Adrian Riesco Valbuena

MUSIC TRENDS

Quantity: 10

Current order: DATE

○ Descending

Bar selector

- ☑ Popularity
- ☑ Danceability
- ☑ Energy
- □ Loudness
- ☑ Speechiness
- □ Acousticness
- □ Instrumentalness
- □ Liveness
- □ Valence
- □ Tempo
- □ Duration (s)

Line selector

- ○ Popularity
- ○ Danceability
- ○ Energy
- ○ Loudness
- ○ Speechiness
- ○ Acousticness
- ○ Instrumentalness
- ○ Liveness
- ● Valence
- ○ Tempo
- ○ Duration (s)
- ○ --None--

FULL DATASET

Show/hide column: Popularity - Danceability - Energy - Key - Loudness - Speechiness - Acousticness - Instrumentalness - Liveness - Valence - Tempo - Duration (s) - Time signature - Mode

Show 10 entries

Search:

TWEET ID	DATE	LINK	TRACK NAME	ARTIST NAME	POPULARITY	DANCEABILITY	ENERGY	DURATION (S)
1542749371887919000	07/01/2022 05:58:07	🐦	Material Girl	Madonna	79	74.2	88.3	240
1542749300417265700	07/01/2022 05:57:50	🐦	Intro	RTB MB	53	79.6	50.6	137
1542749240954323000	07/01/2022 05:57:36	🐦	Spanish Harlem (Made Famous by Aretha Franklin)	R&B Soul Superstars	0	67.8	55.9	213
1542749237095669800	07/01/2022 05:57:35	🐦	Only King Forever (Joaquin Bynum Mix) (with J. Monty & Elevation Worship)	The Sound J. Monty Elevation Worship	37	57.2	90.5	188
1542749188257157000	07/01/2022 05:57:24	🐦	Thinking out Loud	Ed Sheeran	83	78.1	44.5	282
1542749159161208800	07/01/2022 05:57:17	🐦	Money, Money, Money	Budapest Bár	17	81.3	74.1	204
1542749148214091800	07/01/2022 05:57:14	🐦	Trip on New Shores	Reuben Vaun Smith	26	60.9	60.2	230
1542749145953235000	07/01/2022 05:57:13	🐦	Can We Talk	Tevin Campbell	63	65.8	71.1	285
1542749145143943200	07/01/2022 05:57:13	🐦	This Heaven	David Gilmour	34	53.4	41.8	265

CONCLUSION AND FUTURE WORK LINES

Conclusion and future work lines

Final result is considered a success

Future work lines:

- Improve the percentage of correctly identified tracks.
- Ensure that we capture as much data as possible.
- Add visualizations in the front-end layer.
- Upload from history functionality.
- Replace Docker Compose with a more suitable tool.
- Refactor the code.
- Partial or total migration to the cloud.

Questions?

Thank you for your time!

Adrián Riesco Valbuena