Data Eng Final Presentation: Music DB Project

Team:

Duygu Dalman Ciplak
Hakan Dogan

Context

Data related:

- There are many websites to reach music and song records. But, target data fields such as lyrics are not always easily available
- As an example, Spotify has long tried partnership with Musixmatch to link and sync song lyrics data. But, it still majorly lacks this information in its app.

Machine Learning related:

- Recommender systems are frequently used by technology firms, i.e. Amazon, Youtube, Spotify all use this method in their UI
- It can be based on user ratings, shopping habits, text content, image content, audio data and many more
- Here, we use language content in the track name data

METHODOLOGY & CONTENT

PROBLEM DEFINITION

- *DATA MODELLING*
- *DATA CONSUMPTION*

SERVICE ARCHITECTURE

- *MYSQL*
- *MACHINE LEARNING*
- *API*
- *DOCKER*

INTEGRATION / DEPLOYMENT

- *UNIT TESTS*
- *GITHUB*

CODE DEMONSTRATION AND OVERVIEW

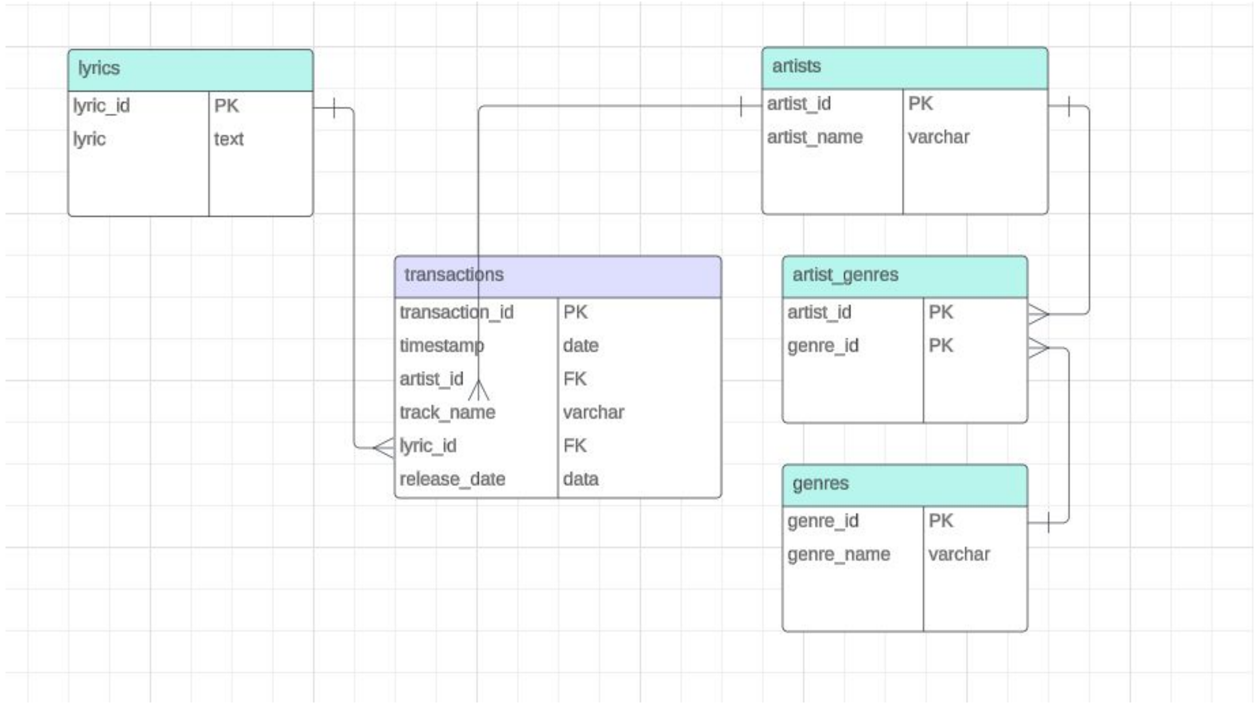
DATA MODELLING

Database Creation:

- Snowflake Schema in MySQL Database
- Efficient data organization for song metadata and user interactions.

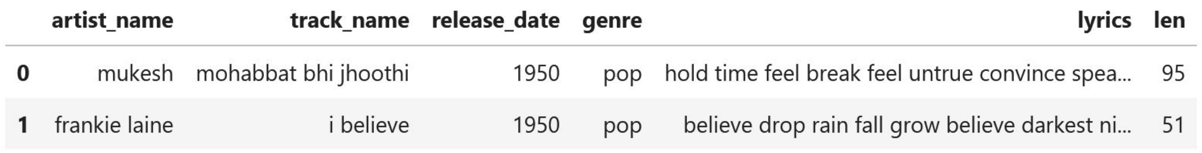
DATA RESOURCES:

- Mendeley Dataset
- SPOTIFY API
- Genius Website & API
- Kworb Website



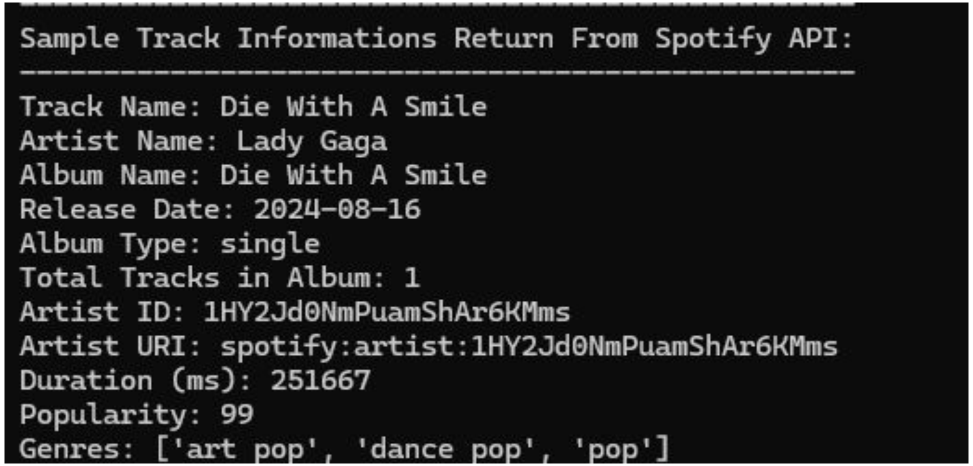
DATA INGESTION

Mendeley Dataset:

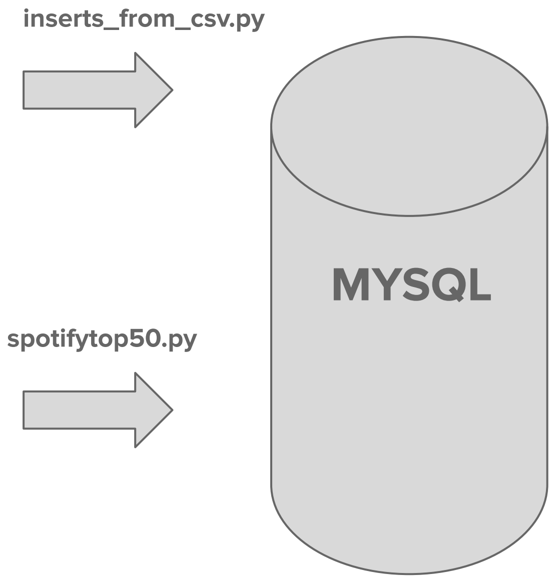


	artist_name	track_name	release_date	genre	lyrics	len
0	mukesh	mohabbat bhi jhoothi	1950	pop	hold time feel break feel untrue convince spea...	95
1	frankie laine	i believe	1950	pop	believe drop rain fall grow believe darkest ni...	51

Spotify API:



```
Sample Track Informations Return From Spotify API:
--------------------------------------------------
Track Name: Die With A Smile
Artist Name: Lady Gaga
Album Name: Die With A Smile
Release Date: 2024-08-16
Album Type: single
Total Tracks in Album: 1
Artist ID: 1HY2Jd0NmPuamShAr6KMms
Artist URI: spotify:artist:1HY2Jd0NmPuamShAr6KMms
Duration (ms): 251667
Popularity: 99
Genres: ['art pop', 'dance pop', 'pop']
```



Container Architecture

Docker Compose Configuration:

- MySQL container (mysql)
- Music Data API container (music-db-api)
- ML Recommender System API container (music-recsys-api)

Functionality:

- Each container is responsible for distinct functionalities, promoting modularity and scalability.
- Enables easy deployment and management of services.

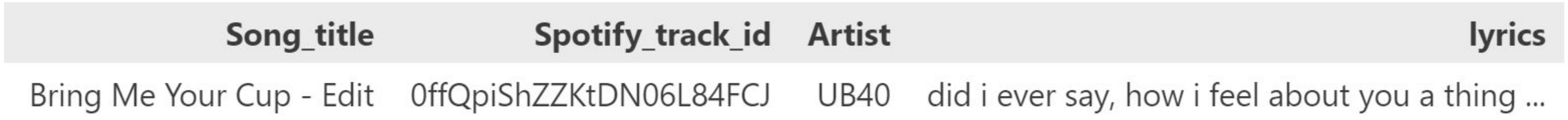
Data Acquisition

- **Kworb Website**

500k tracks

- **Genius API & Website**

1000 tracks



Song_title	Spotify_track_id	Artist	lyrics
Bring Me Your Cup - Edit	0ffQpiShZZKtDN06L84FCJ	UB40	did i ever say, how i feel about you a thing ...

- **Mendeley Data** **29000 tracks**
- **Kworb Data** **100000 tracks**

Recommender System

Training Dataset

Recommender System

FEATURE EXTRACTION

- Content-based filtering
- Count Vectorizer for the *track_name*

MODEL TRAINING

- Count matrix
- Cosine similarity

	I	sun	and	travel	rain	nice	you
track1	0	1	0	0	1	1	0
track2	1	0	1	1	0	1	0
track3	0	0	0	1	0	0	1
track4	1	1	0	0	0	1	0

Ref:

DataScientest

Unit Tests - Data

Test Scripts:

- **test_create_tables.py:** Validates the creation of database tables.
- **test_inserts_from_csv.py:** Ensures data from CSV files is inserted correctly.
- **test_spotifytop50.py:** Verifies data retrieval and insertion from Spotify API.

Testing Approach:

- **Utilizes pytest framework for reliability.**
- **Mock objects to simulate database interactions, enhancing test speed and accuracy.**

Unit Tests - ML

Test Scripts:

- **test_recsys_track_name_inference.py:**

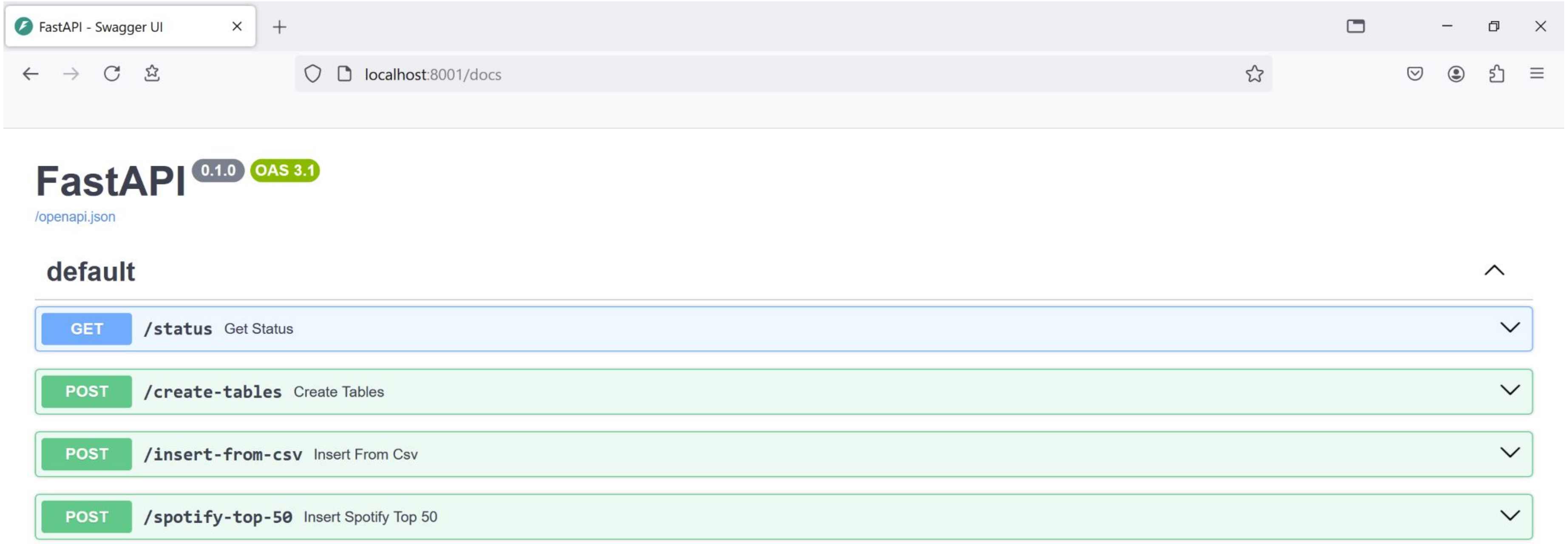
  Tests the corresponding Python module and function.

Testing Approach:

- **Ensures the right choice of test split in data**
- **Ensures the proper input type (string) for the track name**
- **Ensures the return type, i.e. the recommendations as a DataFrame**

DATABASE API

- **Overview of Endpoints**



FastAPI - Swagger UI

localhost:8001/docs

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

GET /status Get Status

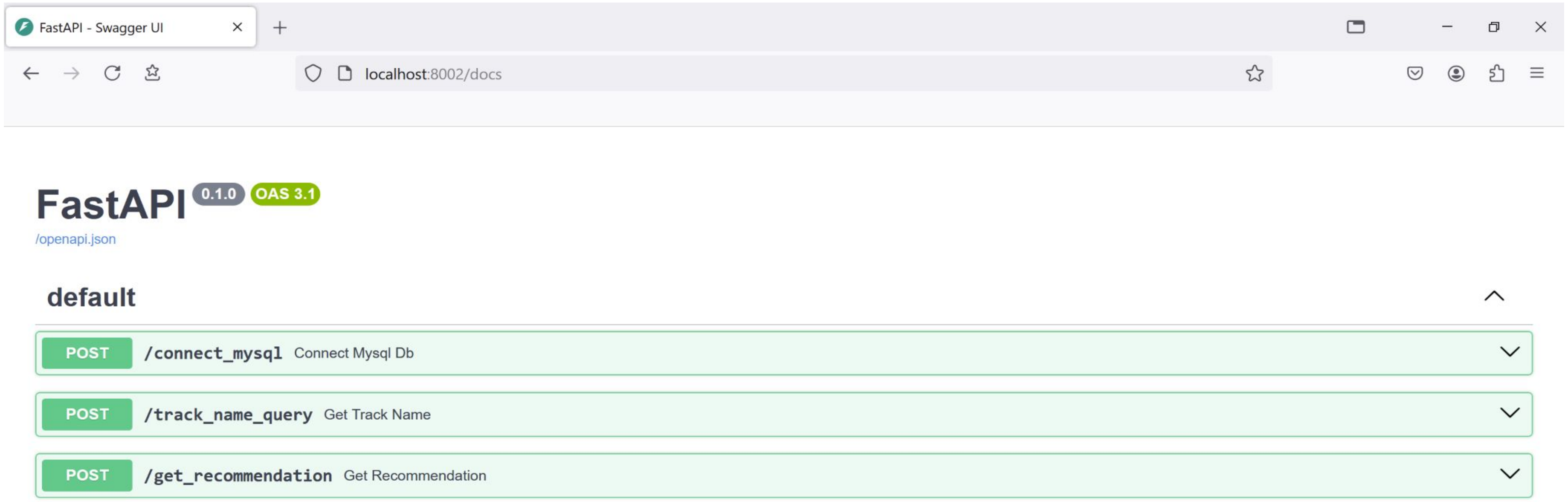
POST /create-tables Create Tables

POST /insert-from-csv Insert From Csv

POST /spotify-top-50 Insert Spotify Top 50

RECSYS API

- Overview of Endpoints



FastAPI - Swagger UI

localhost:8002/docs

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

POST /connect_mysql Connect Mysql Db

POST /track_name_query Get Track Name

POST /get_recommendation Get Recommendation

LIVE DEMO

Live demonstration will cover:

PART 1

- **Creating the Database**
- **Data Ingestion from CSV and Spotify API**

PART2

- **Query for obtaining artist name and track name**
- **API request for getting recommendations for specific track**

CI Pipeline

CI: ci_workflow.yml file is executed by Github Actions on the master branch

CI:

- **Set-up Ubuntu and python**
- **Check syntax errors with flake8**
- **Launch Docker compose**
- **Create Database and Ingest Data through API endpoints**
- **Query a track name and get recommendations through API endpoints**

CONCLUSIONS

Microservice Database-API-Docker Architecture with CI

Implemented different microservice architectures:

- API with Database and Containerization
- MySQL server for data ingestion and queries
- Monitoring with a unified log file
- Machine learning algorithm as Data Consumption method
- Testing methods

Potential improvements:

- AIRFLOW to schedule Spotify and Genius API requests. So that
  - Monitor the changes to the Top 50 lists
  - Fetch more song information to expand the database

Thank you for your attention