1. Introducción

1.1 Descripción del proyecto.

MusicStream es una plataforma de streaming que busca profundizar en las tendencias musicales y optimizar la experiencia de sus usuarios. Este proyecto se centrará en el análisis de la popularidad de canciones y álbumes desde el año 2000 hasta el presente, considerando factores como las calificaciones, el número de reproducciones y las opiniones de los usuarios.

1.2 Objetivos

La meta es identificar las canciones y álbumes más destacados en MusicStream. Para lograrlo, utilizaremos técnicas de extracción de datos de varias fuentes, incluyendo la API de Spotify, MusicBrainz y last.fm. La información recopilada se ha organizado en una base de datos para realizar consultas con las que obtendremos insights clave.

1.3 Audiencia

Esta documentación está destinada a futuros desarrolladores que participen en el proyecto, para que entiendan las dimensiones del mismo, los objetivos para los que se ha creado ,las especificaciones pedidas por el cliente y cómo se han llevado a cabo.

Sin embargo puede ser útil también para explicar a rasgos generales a otros miembros de la compañía como está estructurado el producto y el funcionamiento del mismo.

2. Requisitos

2.1 Requisitos funcionales

En primer lugar haremos una extracción de datos con conexión a APIs de Spotify, MusicBrainz y last.fm para obtener información sobre canciones, álbumes y artistas. Una vez tengamos esto, almacenaremos los datos en una base de datos estructurada para almacenar la información extraída evitando duplicados.

Una vez tengamos esta información distribuida en tablas hay que realizar consultas SQL para responder a preguntas clave sobre reproducciones, géneros y artistas.

2.2 Requisitos no funcionales

Para que el rendimiento sea óptimo el sistema debe ser capaz de procesar y almacenar datos en tiempo real o casi en tiempo real. Las consultas SQL deben ejecutarse en un tiempo razonable, idealmente en menos de 2 segundos para un conjunto de datos promedio

La base de datos debe ser escalable para manejar un aumento en la cantidad de datos y usuarios sin comprometer el rendimiento. El sistema debe poder integrarse fácilmente con futuras fuentes de datos y APIs adicionales.

Respecto a la seguridad, los datos sensibles deben estar protegidos mediante cifrado, tanto en tránsito como en reposo. Es muy importante el acceso a la base de datos y a la API debe estar restringido y controlado mediante autenticación adecuada.

Nuestro sistema debe contar con herramientas de monitoreo para rastrear el rendimiento del sistema y el uso de recursos.

2.3 Herramientas

Las herramientas y tecnologías necesarias para el proyecto son las siguientes, como lenguajes de programación hemos usado Python y SQL. Python para la generación del código necesario para crear el proyecto y SQL para las consultas a la BBDD una vez que tenemos los datos ahí.

El entorno de desarrollo que se usa es Visual Studio Code.

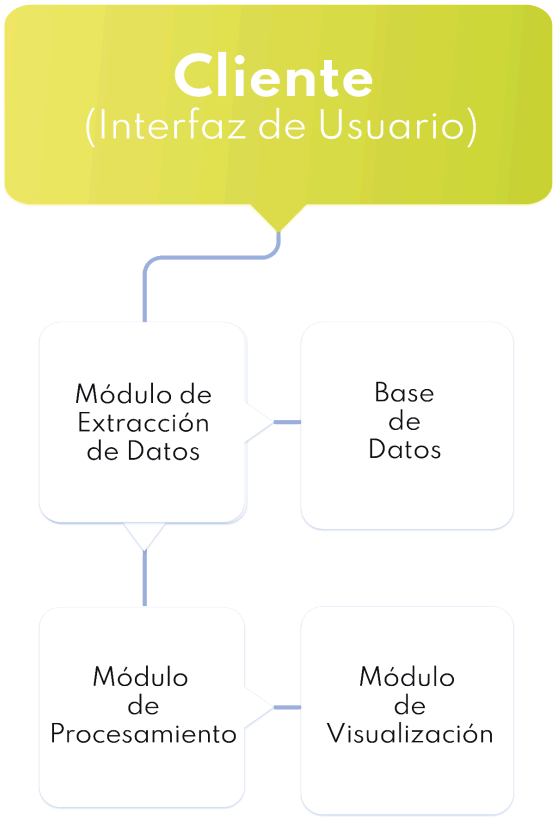
Respecto a la BBDD, para la gestión de bases de datos, diseño de esquemas y ejecución de consultas SQL. MySQL Workbench

Para extraer la información necesaria usaremos tres APIS diferentes. Spotify para obtener información sobre canciones, álbumes y artistas. MusicBrainz para detalles adicionales sobre artistas y álbumes y Lastfm para datos de popularidad y oyentes.

Hemos usado GitHub para alojar el repositorio del proyecto y facilitar la colaboración en equipo y Git para el control de versiones del código

En cuanto a frameworks y bibliotecas de Python hemos usado: Pandas para la manipulación y análisis de datos. Requests para realizar solicitudes HTTP a las apis. Glop para la optimización matemática en la que se necesiten gestionar decisiones bajo restricciones específicas.

3. Arquitectura del sistema



4. Guía para desarrolladores

4.1 Estructura de código

Importaciones:

- Se importan las bibliotecas necesarias para realizar solicitudes `requests`, manipular datos (`pandas`), y acceder a las APIs de Spotify y MusicBrainz.
- import mysql.connector es una biblioteca de Python que permite conectar y trabajar con bases de datos MySQL
- import glob para importar todos los archivos CSV

Clase:

- Esta clase `MusicStream` en Python está diseñada para interactuar con tres APIs relacionadas con música (Spotify, MusicBrainz y Last.fm) para obtener información sobre canciones y artistas, basándose en un género y año específico.

Método `__init__`:

- Inicializa la clase con los atributos `género` (género) y `año`.

Método `spotify`:

- Se conecta a la API de Spotify usando `SpotifyClientCredentials`.
- **Búsqueda en Spotify**: Este método busca canciones en Spotify utilizando el género y el año proporcionados.
- **Estructura de Datos**: Crea un diccionario llamado `tracks` que almacena las canciones, artistas, género, tipo de pista y año de lanzamiento.
- **Iteración y Búsqueda**:
    - Utiliza un bucle `for` para iterar con un `offset` que permite paginar los resultados, ya que Spotify limita la cantidad de resultados devueltos en una sola solicitud.
    - Realiza una búsqueda de pistas con `self.spotify_client.search()`, que devuelve una lista de pistas que coinciden con los criterios de búsqueda.
    - Por cada pista, verifica si ya se ha agregado a la lista de canciones para evitar duplicados y almacena la información relevante.
- **Manejo de Errores**: Se utiliza un bloque `try-except` para manejar errores que puedan ocurrir durante la búsqueda, imprimiendo un mensaje de error y rompiendo el ciclo si hay problemas..

Método `musicbrainz`:

- **Información de Artistas**: Este método utiliza los nombres de los artistas recuperados de Spotify para buscar información más detallada en MusicBrainz.
- **Eliminación de Duplicados**: Convierte la lista de artistas en un conjunto (set) para eliminar duplicados antes de la búsqueda.
- **Estructura de Datos**: Crea un diccionario `artists_info` para almacenar la información del artista, como nombre, país de origen, área de origen, fechas de inicio y final de actividad.

- **Búsqueda y Almacenamiento**:
    - Realiza una búsqueda de artistas en MusicBrainz y almacena la información en el diccionario. Utiliza el método `get()` para evitar errores si las claves no están presentes.
    - También maneja casos en los que no hay datos disponibles usando un `try-except`.
- **Manejo de Errores**: Utiliza un `except IndexError` para manejar situaciones en las que no se encuentra el artista.

Método `lastfm`:

- **Información de Artistas en Last.fm**: Este método busca información adicional sobre los artistas usando la API de Last.fm.
- **Configuración de la API**: Obtiene la clave de API desde una variable de entorno para mayor seguridad.
- **Estructura de Datos**: Crea un diccionario `artists_lfm` que almacena información sobre los artistas, incluyendo biografía, oyentes, recuento de reproducciones y artistas similares.

Después de sacar toda la información hacemos los DataFrame que se hacen con la librería pandas.

Cuando tengamos todos los DataFrame se pasan a archivo csv con la función .to_csv()

Creamos una variable para buscar todos los csv con la librería glob y con la librería pandas se concatena todos los archivos.

Hacemos una lista de tuplas para poder subir los datos a la BBDD creada en SQL Workbench.

Para subir los datos a la BBDD necesitamos las librerías mysql.connector y errorcode.En el código hemos puesto un ignore para que no se puedan subir los valores duplicados.

4.2 MySQL Workbench.

Creamos el schema con las tres tablas de las APIs, que son las siguientes:

- Spotify
- MusicBrainz
- lastFM

Primero creamos la tabla de musicbrainz para que la Primary Key sea artists con valores únicos. La Foreign Key de la tabla spotify es artist con referencia a MusicBrainz (artist), la tabla de lastFM la Foreign Key es artist con referencia a MusicBrainz (artist).

5.Pruebas

5.1 Fase 1: Extracción de datos.

- **API SPOTIFY:**

  Primero hicimos la prueba de extraer 50 canciones por cada género y año. Analizamos la estructura del json para extraer los campos necesarios para la BBDD. Cuando conseguimos sacar las 50 canciones de cada género y año, reformulamos el código para extraer 500 solicitadas por el cliente.

- **API MUSICBRAINZ:**

  Elegimos un artista para analizar la estructura y así poder extraer los datos para la BBDD.

- **API LASTFM:**

  Hicimos las mismas pruebas que en la api de musicbrainz

5.2 Fase 2: Organización y Almacenamiento de la BBDD.

Empezamos a crear una base de datos de prueba para subir unos datos limitados, de esta manera pudimos ver que la primary key no puede tener datos repetidos. Reorganizamos las tablas y cambiamos las primary key y foreign key.

Hicimos unas listas de tuplas con unos datos limitados a 10 y hubo errores en la denominación de las columnas.

Cargamos todos los datos en la BBDD y observamos que la manera más eficiente es crear primero la tabla de musicbrainz y relacionarlas con las otras dos tablas restantes.

5.3 Fase 3: Análisis y consultas de datos.

Realizamos diferentes consultas en las tablas para ver si están bien estructuradas.