DOCUMENTACIÓN DEL PROCESO: EXTRACCIÓN DE DATOS DE APIs

En esta fase se cumplieron los criterios de aceptación:

- Configurar y realizar solicitudes a la API de Spotify, MusicBrainz y last.fm para obtener datos iniciales.
- Almacenar los datos extraídos en una estructura temporal para posteriormente insertarlos en la base de datos CSV.

A continuación, se describen los pasos que nos llevaron, desde el planteamiento de la idea inicial hasta la implementación final del código, para extraer los datos en las 3 APIS's.

Extracción de Datos desde la API de Spotify

Usamos la función load_dotenv(), de la biblioteca python-dotenv, para ocultar nuestra Key al subir el archivo a Git.

PASO 1

Revisamos y analizamos la documentación suministrada por Spotify for Developers y la librería Spotipy. Este paso incluyó lo siguiente:

- Registro con la cuenta de Spotify para conseguir la API Key.
- Creación de un dashboard para registrar el proyecto.
- Posteriormente, en la documentación, navegar por el punto "search for item" donde observamos el endpoint (q, type, limit, offset) con el "search".
- Importamos la librería Spotipy para poder autentificar los requests que nos daba la documentación de las APIs de Spotify (q, type, limit, offset).

La búsqueda de los datos se realizó utilizando la autenticación de la librería Spotipy (sp) y .search que nos proporcionaba la documentación de la API de Spotify.

PASO 2

Definimos la función "extraccion_datos" para buscar y recopilar datos de pistas musicales basadas en años (2021 y 2022) y géneros (pop, rock, indie y rap).

- **Inicialización:** Se crea una lista vacía datos para almacenar los resultados de las búsquedas.

- **Iteración Anidada:**
  Se recorren todos los años proporcionados en la lista años.
  Para cada año, se recorren todos los géneros en la lista géneros.

- **Paginación:**

Se utiliza un bucle para gestionar la paginación de resultados, con un rango de 0 a 1000 en pasos de 50. Esto permite obtener hasta 1000 pistas por cada combinación de año y género (que es el máximo que ofrece el *offset*) .
Se hace una llamada a la API usando sp.search para buscar pistas musicales que coincidan con el género y año especificados, y se limitan los resultados a 50 por llamada.

- Almacenamiento: Los datos obtenidos se añaden a la lista "datos".

PASO 3

Definimos la función "filtrado_datos" para organizar y filtrar los datos de canciones, para extraer información relevante.

- **Inicialización:** Se crea un diccionario datos_filtrados con listas vacías para almacenar información como artista, tipo, título de canción, nombre del álbum, año y género.

- **Iteración:** Se recorre cada elemento en datos, que contiene información sobre canciones. Para cada una de éstas, se extraen y almacenan los datos necesarios (artista, tipo, título, álbum y año de lanzamiento).

- **Género:** Se verifica si cada género de la lista generos está presente en el campo 'href' de los datos. Si hay coincidencia, se añade el género al diccionario.

- **Creación de DataFrame:** Se convierte el diccionario datos_filtrados en un DataFrame.

- **Exportación a CSV:** El DataFrame se guarda como 'spotify_df_concatenado.csv' para usarlo en la fase posterior.

- **Lista de Artistas Únicos:**Usamos el CSV creado previamente, para obtener una lista de artistas únicos.
  Esta lista se convierte en un DataFrame y se guarda como 'lista_artistas_spotify.csv' para futuras extracciones.
  Retorno: La función devuelve el DataFrame con los datos filtrados.

Extracción de detalles de artistas con la API de MusicBrainz

PASO 1

A través de la documentación consultada en la Api de MusicBrainz, construimos la URL para realizar una consulta. La consulta incluye el nombre del artista y especifica que la respuesta debe estar en formato JSON.

"url = f'https://musicbrainz.org/ws/2/artist/?query={artista}&fmt=json'"

PASO 2

Definimos una función llamada extraccion_datos, este recibe un argumento lista_csv.

- Se inicializa una lista vacía llamada datos_artistas, que se usará para almacenar la información de los artistas obtenida de la API.
- Utiliza pandas para leer el archivo CSV obtenido de la fase anterior, de Spotipy y cargarlo en un DataFrame llamado df_lista_artistas.
- Se crea una lista llamada artistas a partir de la columna '0' del DataFrame.
- Se inicia un bucle for que itera sobre cada nombre de artista en la lista artistas.
- Se utiliza time.sleep(1) para pausar la ejecución del bucle durante 1 segundo entre cada llamada a la API. Lo usamos para evitar sobrecargar la API, limitando las llamadas a 1 por segundo.
- Se realiza una llamada HTTP GET a la URL construida anteriormente, usando la librería requests.
- Se extrae el contenido de la respuesta en formato JSON y se almacena en la variable datos.
- Finalmente, la función devuelve la lista datos_artistas, que contiene los datos de todos los artistas consultados.

PASO 3

Creamos la función "filtrado_datos", que se encarga de limpiar y estructurar los datos de los artistas, a partir de un conjunto de datos (listado de diccionarios).

- Inicialización: Se crea un diccionario "info_artistas" para almacenar información clave de los artistas (nombre, tipo, país, fechas de actividad, etc.).

- Iniciamos una iteración:

  Se intenta extraer información de cada artista, manejando excepciones para evitar errores por datos faltantes:

  Nombre: Se obtiene el nombre del artista.
  Tipo: Se identifica si es un grupo o una persona.
  País y Área de origen: Se extraen estos datos.
  Fechas: Dependiendo del tipo (grupo o persona), se almacenan fechas de actividad o nacimiento/fallecimiento.

- Manejo de Errores: Se utilizan excepciones para agregar valores por defecto ('None' o '-') cuando los datos no están disponibles.

- Creación de DataFrame: Al final, se convierte el diccionario en un DataFrame de pandas y se guarda como un archivo CSV llamado 'musicbrainz_df_concatenado.csv'.

- Retorno: Devuelve el DataFrame con los datos limpios.

Extracción de detalles de artistas con la API de last.fm.

PASO 1

Nos registramos en la página de API Last.fm, y así obtuvimos la ClaveApi (Key), la cual es requerida para hacer la llamada.

Usamos la función load_dotenv(), de la biblioteca python-dotenv, para ocultar nuestra Key al subir el archivo a Git.

PASO 2

Revisamos y analizamos la documentación proporcionada por la Api, y allí obtuvimos la URL para hacer la llamada. Encontramos que La URL raíz de la API se encuentra en "http://ws.audioscrobbler.com/2.0/" y para realizar la busqueda de la información de los artistas por nombre, la completamos con la URL JSON: "/2.0/?method=artist.search&artist=cher&api_key=SU_CLAVE_API&format=jso n". Encontrar la URL correcta se nos hizo fácil, ya que la documentación no es compleja.

La documentación también nos indicaba los parámetros: limit, page, artist y Api_key, sin embargo, solo artist y Api_key son los obligatorios y fue los que usamos en nuestra URL.

URL ESTRUCTURADA:
http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist={artista}&api_ke y={API_KEY}&format=json

PASO 3

Creamos una función "extraccion_datos" que tiene como objetivo extraer información sobre artistas desde esta API, utilizando el archivo CSV que obtuvimos de la fase de Spotipy.

- **Inicialización:** Se crea una lista vacía "datos_artistas" para almacenar los datos obtenidos.

- **Lectura del CSV:** Se lee el archivo CSV (que contiene nombres de artistas) y se convierte la primera columna en una lista llamada "artistas".

- **Iteración y Llamadas a la API:**

Se itera sobre cada artista en la lista, usando la URL previamente construida.
Se realiza la llamada a la API y se obtienen los datos en formato JSON.

- **Almacenamiento de Datos:** Los datos obtenidos se añaden a la lista datos_artistas.

- **Retorno:** Finalmente, la función devuelve la lista con todos los datos solicitados de los artistas.

PASO 4

También creamos una función "filtrado_datos" que se encarga de limpiar y estructurar los datos, partir de una lista de diccionarios.

- **Inicialización:** Se crea un diccionario "info_artistas", con listas vacías para almacenar campos específicos como nombre, biografía, oyentes, reproducciones y artistas similares.

- **Iteración:** Se recorre cada elemento en datos para extraer información relevante:

Se intenta obtener el nombre del artista, manejando excepciones si la clave no existe.
Se extraen la biografía, el número de oyentes y el conteo de reproducciones.
Para los artistas similares, se recopilan todos los nombres en una lista.
Manejo de Errores: Se utilizan excepciones para ignorar errores en caso de que falten datos.

- **Creación de DataFrame:** Se convierte el diccionario en un DataFrame de pandas y se guarda como un archivo CSV llamado 'lastfm_df_concatenado.csv'.

- **Retorno:** La función devuelve el DataFrame con los datos limpios.