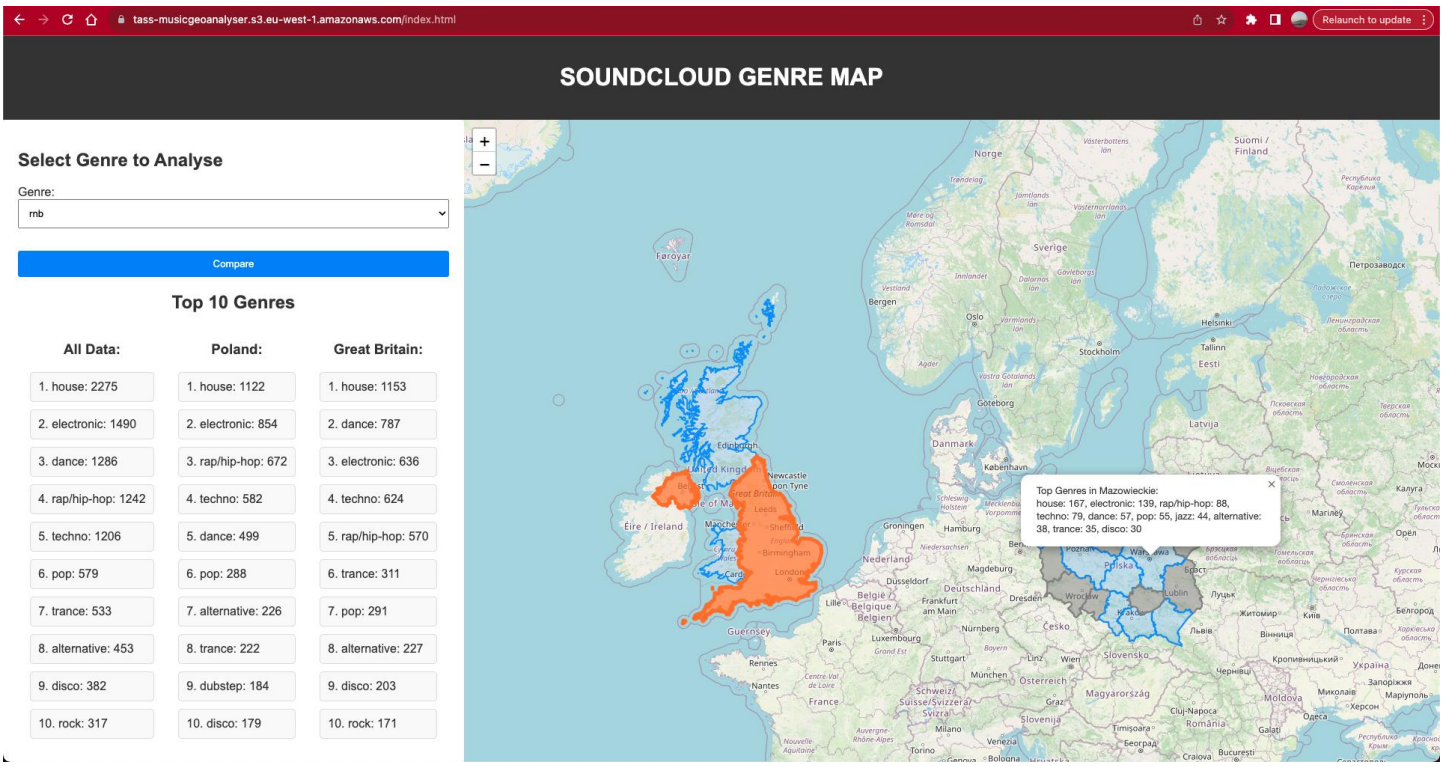
TASS – Soundcloud Genre Map

Maciej Groszyk, Ignacy Kleszczyński

Opis aplikacji

Opis funkcjonalności:

Prezentacja aplikacji: https://tass-musicgeoanalyser.s3.eu-west-1.amazonaws.com/index.html



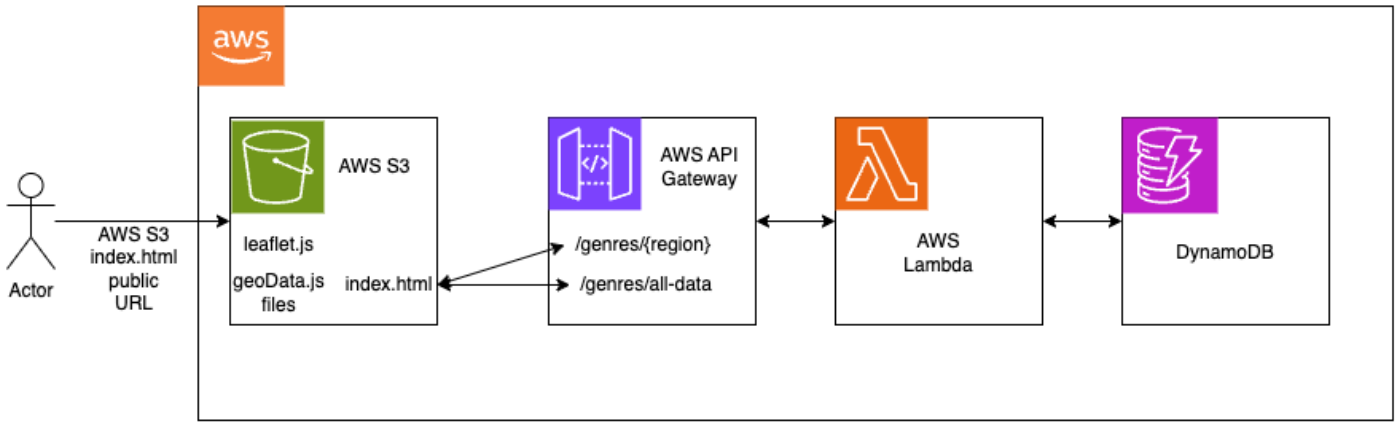
Aplikacja jest podzielona na dwa główne elementy panel po lewej stronie i interaktywną mape.

Główny panel aplikacja sklada sie z 3 elementów.

1. Rozwijana lista która pozwala nam wybrać gatunek muzyczny, który chcemy przeanalizować na mapie.
2. Przycisk compare który podświetla na pomarańczowo regiony w których wybrany gatunek znajduje sie w top 10 gatunków dla danego regionu
3. Ostatni element, to listy które podsumowują wyniki dla wszystkich danych i poszczególnych krajów

Interaktywna mapa oprócz funkcji podświetlania regionów, posiada możliwość do sprawdzenia każdego regionu z osobna. Po kliknięciu w region wyskakuje okienko z wypisanymi top 10 gatunkami muzycznymi wraz z ilością słuchaczy.

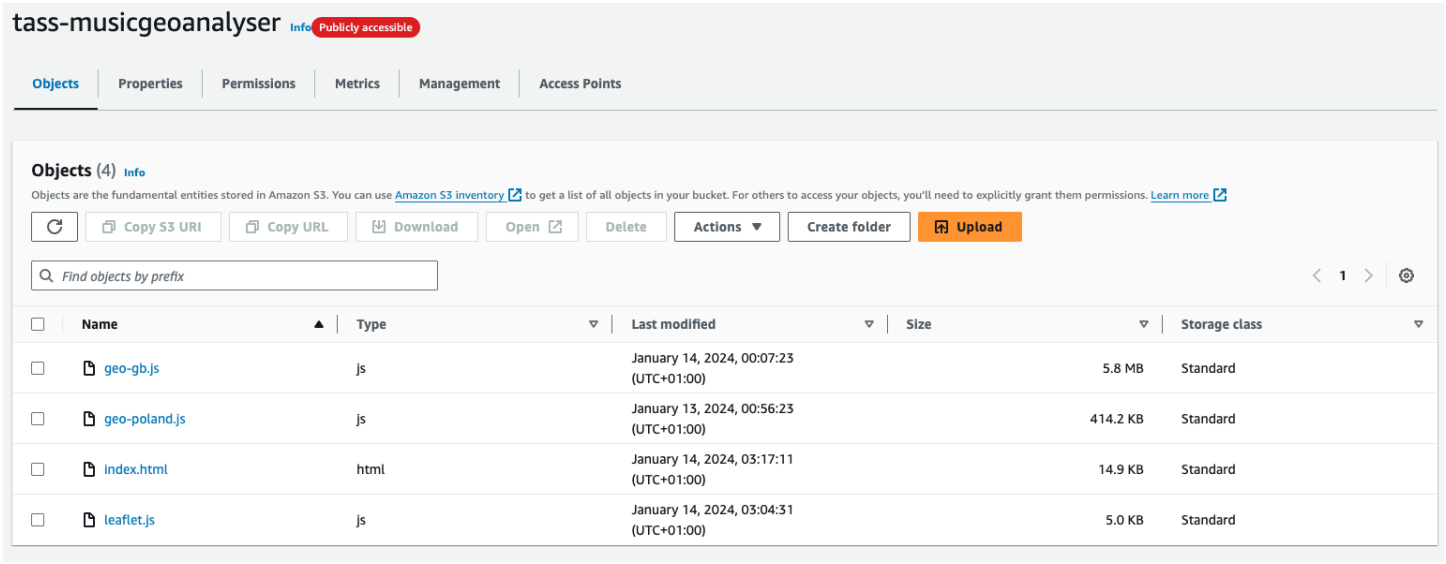
Architektura aplikacji



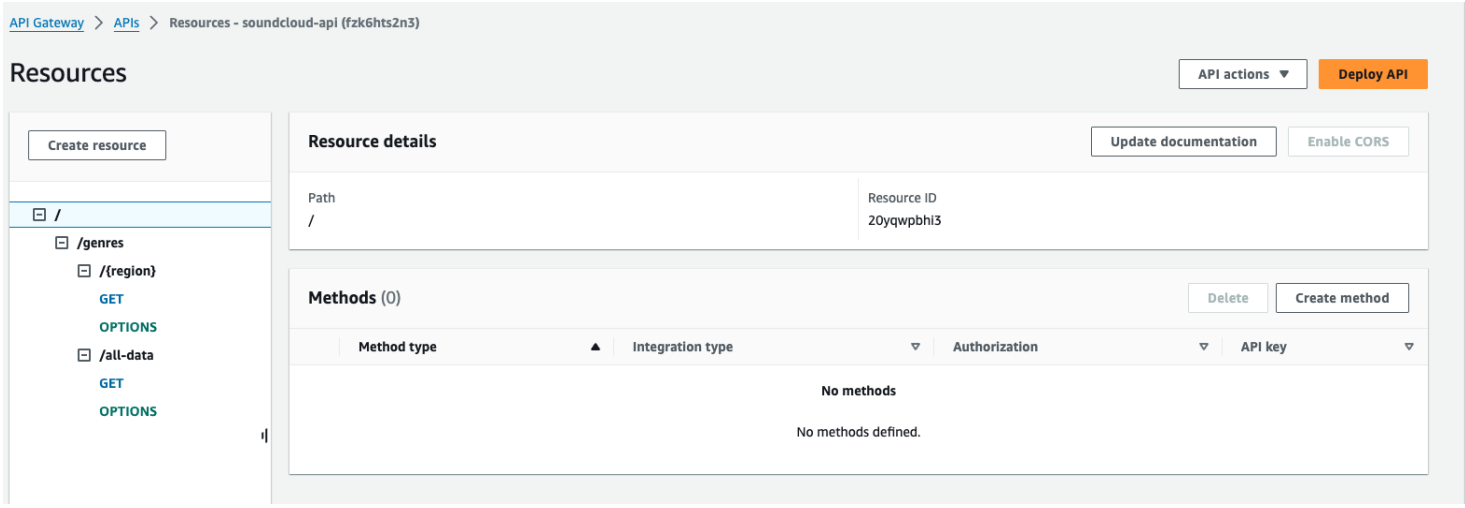
- **Dostęp Użytkownika do Strony:**
  - Użytkownik łączy się ze stroną poprzez URL: **https://tass-musicgeoanalyser.s3.eu-west-1.amazonaws.com/index.html**.
  - Plik **index.html** ładuje skrypty z **leaflet.js** i **geoData.js**. Jest to obejście problemów z CORS, które zwykle występują w AWS S3.
- **Interakcja między Usługami AWS:**
  - **Bucket S3:** Hostuje statyczne pliki (**index.html**, **leaflet.js**, **geoData.js**).
  - **API Gateway:** Zarządza przychodzącymi żądaniami i łączy się z funkcjami Lambda.
  - **Funkcje Lambda:** Obsługują logikę różnych endpointów (**/genres/{region}**, **/genres/all-data**).
  - **DynamoDB:** Przechowuje i dostarcza dane dotyczące gatunków muzycznych i liczby słuchaczy w miastach.
- **Funkcja Lambda dla Endpointu /genres/{region}:**
  - Mapuje dany region na odpowiadające mu miasta za pomocą **regionToCityMapping**.
  - Dla każdego miasta w regionie, funkcja Lambda wykonuje zapytanie do DynamoDB.
  - Agreguje dane, aby obliczyć top 10 gatunków muzycznych dla tego regionu.
- **Funkcja Lambda dla Endpointu /genres/all-data**:
  - Pobiera wszystkie dostępne dane z DynamoDB bez dodatkowego przetwarzania.

- **Skalowalność i Obsługa CORS:**
  - Architektura jest skalowalna, co umożliwia łatwe dodawanie nowych regionów lub miast.
  - W obsłudze CORS, funkcja Lambda zawiera nagłówek **'Access-Control-Allow-Origin': '*'**.

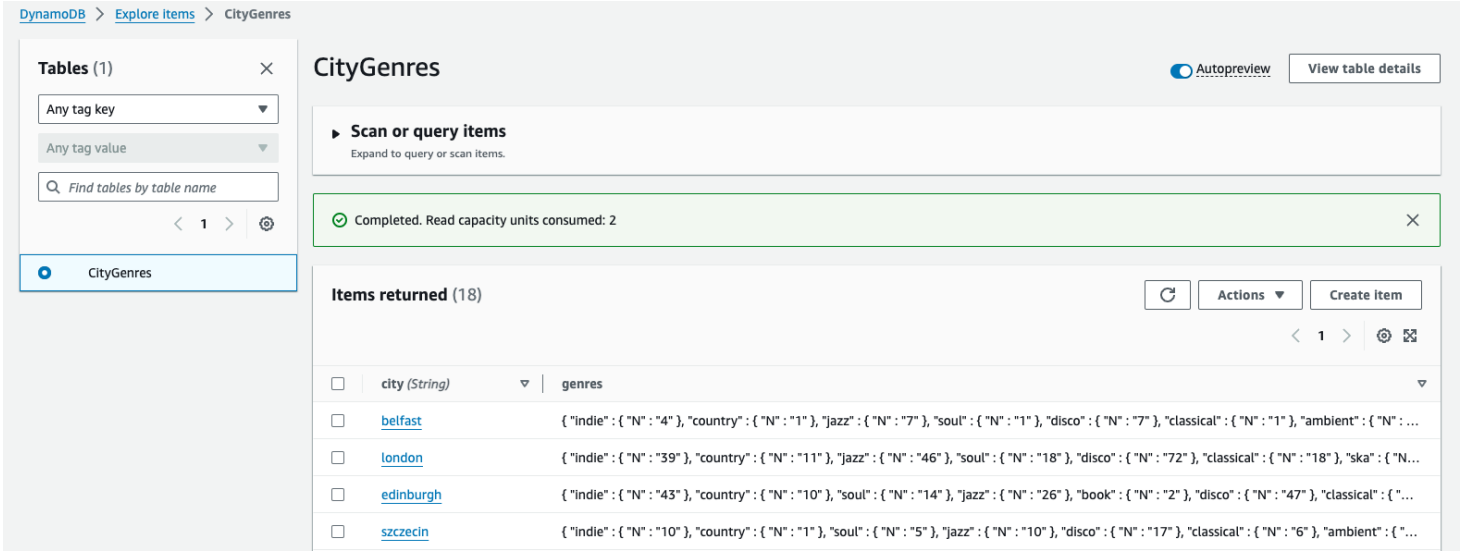
AWS S3



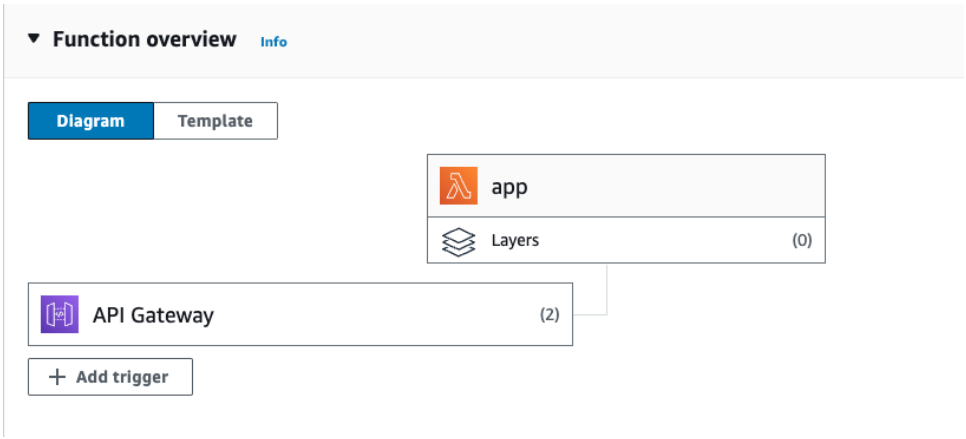
AWS API Gateway



AWS DynamoDB



AWS Lambda



Opis aplikacji

Opis pliku Index.html

- **Struktura Podstawowa:**
  - Deklaracja typu dokumentu (DOCTYPE) jako HTML.
  - Zdefiniowanie języka strony jako angielski (**lang="en"**).
  - Ustawienie meta tagów dla kodowania znaków (UTF-8) i widoczności na urządzeniach mobilnych (viewport).
- **Zależności z Zewnętrznych Źródeł:**
  - Załączenie arkusza stylów dla Leaflet, biblioteki do map, ze wskazaniem na wersję i zabezpieczenia (integrity, crossorigin).
  - Załączenie skryptów Leaflet oraz Handlebars (biblioteka do szablonów).
- **Tytuł Strony:**
  - Tytuł ustawiony na "MusicGeoAnalyzer".
- **Stylowanie CSS:**
  - Zdefiniowane style CSS dla elementów strony, w tym **body**, **header**, **main**, i różnych elementów interfejsu użytkownika takich jak przyciski, listy, i pola wyboru.
- **Skrypty JavaScript:**
  - Funkcje JavaScript do obsługi interakcji użytkownika, takie jak przełączanie widoczności elementów interfejsu (**toggleDropdowns**), porównywanie danych (**compare**) i inne funkcje pomocnicze do przetwarzania i wyświetlania danych.
- **Struktura HTML:**
  - Główne sekcje strony, w tym nagłówek (**header**), główna zawartość (**main**) z panelem do wyboru gatunków muzycznych i mapą.
  - Elementy **select**, **button**, i kontenery do prezentacji danych muzycznych (np. top 10 gatunków).
- **Dynamika i Interaktywność:**
  - Skrypty do dynamicznego ładowania i wyświetlania danych muzycznych, obsługi mapy oraz interakcji z użytkownikiem.
  - Załączenie dodatkowych skryptów zewnętrznych do obsługi danych geograficznych i mapy.
- **Łączenie z Backendem:**
  - Skrypty do pobierania danych z back-endu poprzez wywołania API (np. **fetchAllRegionData**).

Ten plik jest centrum interakcji użytkownika z aplikacją webową, łącząc frontend z backendem i zapewniając dynamiczne wyświetlanie danych oraz interakcje z mapą. Jest to kompleksowy dokument, który integruje różne technologie i narzędzia, tworząc spójny i interaktywny interfejs użytkownika.

Opis Pliku leaflet.js

- **Stylizacja Mapy:**
  - Zdefiniowanie stylów dla wyróżnienia (highlightStyle) i domyślnego wyglądu (defaultStyle) obszarów na mapie. Styl wyróżnienia ma intensywniejszy kolor, większą wagę granic i większą przezroczystość wypełnienia.
- **Przechowywanie Warstw Regionów:**
  - Utworzenie zmiennej **regionLayers** jako pustego obiektu do przechowywania referencji do wszystkich warstw regionów na mapie.
- **Inicjalizacja Mapy:**
  - Utworzenie nowej mapy z użyciem biblioteki Leaflet, ustawienie środka i poziomu zoomu.
  - Dodanie warstwy mapy bazowej (tileLayer) z OpenStreetMap.
- **Mapowanie Regionów do Miast:**
  - Definiowanie mapowania **regionToCityMapping** i **countryToRegionMapping**, które służą do przypisania regionów do miast i krajów.
- **Konwersja Nazw Regionów:**
  - Funkcja **convertRegionName** do konwersji nazw regionów, uwzględniająca specjalne znaki diakrytyczne.
- **Pobieranie Danych Gatunków Muzycznych:**
  - Funkcja **fetchGenreData**, która wykonuje wywołanie API do pobrania danych o gatunkach muzycznych dla danego regionu.
- **Interakcje z Mapą:**
  - Użycie **L.geoJSON** do ładowania danych geograficznych i tworzenia warstw na mapie.
  - Dla każdej funkcji (feature) na mapie, przypisanie warstwy do odpowiedniego regionu w **regionLayers**.
  - Ustawienie zdarzenia kliknięcia na warstwę, które pobiera dane gatunków muzycznych i wyświetla je w dymku informacyjnym (popup).
- **Stylizacja Regionów na Mapie:**
  - Funkcja **style** w **L.geoJSON**, która stylizuje regiony na mapie. Styl zależy od tego, czy region znajduje się w **regionToCityMapping**. Regiony spoza mapowania mają szary kolor i inny styl granic i wypełnienia.
- **Dodawanie GeoJSON dla Wielkiej Brytanii:**
  - Ładowanie i dodanie danych GeoJSON dla Wielkiej Brytanii, z podobnymi funkcjami jak w przypadku poprzednich warstw.

Ten plik jest niezbędny dla działania interaktywnej mapy w aplikacji. Obsługuje on dynamiczne ładowanie danych, interakcje użytkownika z mapą (takie jak kliknięcia na regiony), oraz wizualizację danych związanych z regionami i gatunkami muzycznymi. Umożliwia użytkownikom przeglądanie danych muzycznych w kontekście geograficznym, co stanowi kluczową funkcjonalność aplikacji.

Opis plików geo-gb.js i geo-poland.js

- **Przeznaczenie Plików**:
  - **geo-gb.js**: Ten plik zawiera dane geograficzne dotyczące Wielkiej Brytanii. Służy do wizualizacji regionów, takich jak Anglia, Szkocja, Walia i Irlandia Północna, w aplikacji mapowej.
  - **geo-poland.js**: Plik ten obejmuje dane geograficzne Polski, umożliwiając wizualizację polskich województw i ich granic.
- **Struktura i Format**:

    - Oba pliki zostały przekształcone z formatu JSON na skrypty JavaScript. Jest to rozwiązanie, które pozwala na łatwe ich załadowanie i użycie w aplikacji webowej, szczególnie w kontekście ograniczeń związanych z zasadami CORS (Cross-Origin Resource Sharing).
    - Pliki są dołączane do strony HTML za pomocą tagów **<script>**, co umożliwia bezpośredni dostęp do zawartych w nich danych geograficznych w kodzie JavaScript.
- **Wykorzystanie w Aplikacji**:
    - Po załadowaniu, dane z tych plików są używane przez bibliotekę Leaflet.js do renderowania interaktywnych map w aplikacji.
    - Pliki te dostarczają istotnych informacji geograficznych, takich jak granice i kształty regionów, które są kluczowe do wizualizacji danych na mapie.
- **Zastosowanie w Kontekście Aplikacji**:
    - W aplikacji, te dane geograficzne są używane do przedstawienia statystyk specyficznych dla danego regionu (popularność różnych gatunków muzycznych w Wielkiej Brytanii i Polsce)
    - Zmieniona struktura tych plików (z JSON na JavaScript) ułatwia ich integrację i wykorzystanie w dynamicznym środowisku aplikacji webowej, szczególnie gdy interakcje użytkownika z mapą (takie jak kliknięcia na regiony) wyzwalają określone akcje czy zapytania do serwera.

- **Dostosowanie do Specyficznych Wymagań**:
    - Wybierając podczas pobierania opcję "level-1", uzyskuje dane odpowiadające pierwszemu poziomowi podziału administracyjnego, co w przypadku Wielkiej Brytanii obejmuje główne regiony, a w Polsce – województwa. Jest to szczególnie ważne, gdyż zapewnia to zgodność danych z wymaganiami aplikacji.
    - Struktura danych w tych plikach musi być zgodna z oczekiwaniami biblioteki Leaflet.js, aby zapewnić poprawną interpretację i wizualizację geograficzną na mapie.
- **Integracja z Leaflet.js**:
    - Po załadowaniu, aplikacja wykorzystuje te dane do tworzenia warstw na mapie przy użyciu Leaflet.js. Każdy region jest reprezentowany jako osobna warstwa, co umożliwia interaktywne elementy, takie jak kliknięcia na regiony, wyświetlanie dymków z informacjami, czy zmianę stylów regionów w zależności od danych.
- **Optymalizacja i Wydajność**:
    - Przekształcenie plików JSON w skrypty JavaScript może również przyczynić się do poprawy wydajności aplikacji, ponieważ dane są ładowane razem ze stroną, eliminując potrzebę dodatkowych zapytań AJAX do serwera w celu pobrania tych danych.
    - Ważne jest, aby pamiętać o optymalizacji tych plików, szczególnie jeśli zawierają one dużą ilość danych, co może wpływać na czas ładowania strony i ogólną wydajność aplikacji.

Podsumowując, pliki **geo-gb.js** i **geo-poland.js** są kluczowymi elementami aplikacji, dostarczającymi niezbędnych danych geograficznych, które są efektywnie wykorzystywane do interaktywnej wizualizacji na mapie z wykorzystaniem Leaflet.js. Twój sposób załadowania i wykorzystania tych danych demonstruje efektywne podejście do rozwiązywania wyzwań związanych z integracją danych geograficznych w aplikacjach webowych.

Szczegółowy opis skryptu w Lambdzie:

- **Inicjalizacja AWS SDK i DynamoDB Client:**
    - Importowanie modułu AWS SDK i tworzenie nowej instancji klienta DynamoDB (**DocumentClient**).
- **Mapowanie Regionów do Miast:**
    - Zdefiniowanie **regionToCityMapping**, mapującego regiony na odpowiadające im miasta.
- **Funkcja getGenresByCity:**
    - Asynchroniczna funkcja, która wykonuje zapytanie do tabeli DynamoDB (**CityGenres**), aby uzyskać dane o gatunkach muzycznych dla określonego miasta.
    - Obsługa błędów i zwracanie danych lub komunikatu o błędzie.
- **Funkcja getTopGenres:**

    - Funkcja do wyznaczania top 10 gatunków muzycznych z danego słownika (**genreDict**), sortując je według liczby słuchaczy i zwracając pierwsze dziesięć wyników.

- **Funkcja getAllData:**
    - Asynchroniczna funkcja do skanowania całej tabeli **CityGenres** w DynamoDB, zwracająca wszystkie zapisy.
    - Obsługa błędów i zwracanie danych lub komunikatu o błędzie.
- **Handler Główny Lambdy (exports.handler):**
    - Funkcja obsługi zdarzeń Lambda, która analizuje ścieżkę zdarzenia (**event.path**).
    - Dla ścieżki **/genres/all-data**, funkcja wykonuje **getAllData** do pobrania wszystkich danych.
    - Dla innych ścieżek, funkcja przetwarza region podany w parametrze ścieżki, agreguje dane z miast powiązanych z tym regionem, i zwraca top 10 gatunków muzycznych dla tego regionu.
    - W obu przypadkach, ustawia odpowiednie nagłówki, w tym **'Access-Control-Allow-Origin': '*'** do obsługi CORS, oraz kod statusu i treść odpowiedzi w formacie JSON.
- **Obsługa Błędów:**
    - Ogólna obsługa błędów na poziomie handlera, zwracająca kod statusu HTTP 500 i komunikat o błędzie w przypadku wystąpienia wyjątku.

Ten skrypt jest kluczowym elementem architektury systemu, odpowiada za przetwarzanie i udostępnianie danych muzycznych na poziomie regionu i miasta. Dzięki asynchronicznym funkcjom i obsłudze błędów zapewnia niezawodność i efektywność operacji na danych. Ponadto, włączenie nagłówków CORS zapewnia prawidłową integrację z frontendem aplikacji.

Ekstrakcja danych:

Proces ekstrakcji danych z platformy Soundcloud odbył się z wykorzystaniem platformy Rapidapi. Rapidapi to platforma, która umożliwia łatwe zarządzanie różnymi API i korzystanie z nich w jednym miejscu. Zapewnia jednolite środowisko zarządzania, dokumentacji i dostępu do interfejsów aplikacji.
Podczas procesu ekstrakcji danych wysyłane były zapytania get z wykorzystaniem biblioteki requests dla środowiska python.
Pobrane zostały dane użtkowników o najpopularniejszych imionach dla dwóch krajów – Wielkiej Brytanii oraz Polski, z rozróżnieniem ich regionów takich jak np. Mazowsze. Proces pobierania danych odbywa się offline i może zostać zaharmonogramowany w określonym odstępie czasowym.
Dotychczas pobrane zostały dane:

- Dla Polski:
    - około 2000 użytkowników dla regionów Mazowsze, Wielkopolska, Małopolska, Śląsk, Pomorze.
    - około 26000 polubień użytkowników dla utworów na tej platformie
- Dla Wielkiej Brytanii:
    - około 2000 użytkowników dla regionów England, Scotland, Wales, Northern Ireland
    - około 28500 polubień użytkowników dla utworów na tej platformie

Opis skryptu do ekstrakcji danych

- **Importowanie bibliotek**: Skrypt wykorzystuje **requests** do wykonywania zapytań HTTP, **json** do obsługi danych w formacie JSON, **time** i **random** do zarządzania czasem i losowością oraz **names_dataset** do pracy z zestawem danych zawierającym imiona.
- **Konfiguracja i inicjalizacja**: Ustawiane są klucz API, wartości dotyczące nazw imion i opóźnień, a także plik zawierający dane o regionach.
- **Wczytywanie danych o regionach**: Próba wczytania danych z pliku **regions.json**, które zawierają informacje o regionach, takich jak województwa i miasta.
- **Funkcje pomocnicze**:

    - **get_popular_names**: Pobiera popularne imiona z określonego kraju.
    - **get_users**: Pobiera użytkowników SoundCloud na podstawie imienia i miasta.
    - **get_users_for_country**: Pobiera użytkowników SoundCloud dla określonego kraju, używając popularnych imion.
    - **clean_users**: Usuwa zbędne informacje z danych użytkowników.
    - **clean_user_likes**: Usuwa zbędne informacje z danych o polubieniach użytkowników.
    - **get_user_likes**: Pobiera polubienia określonego użytkownika.
    - **get_users_likes**: Pobiera polubienia użytkowników dla określonego zestawu użytkowników.
    - **save_to_file**: Zapisuje dane w formacie JSON do pliku.
- **Wykonywanie skryptu dla konkretnych krajów**: Skrypt wykonuje funkcje dla Polski (**'PL'**) i Wielkiej Brytanii (**'GB'**), zbierając informacje o użytkownikach i ich polubieniach, a następnie zapisuje te dane do plików JSON.

Skrypt skupia się na zbieraniu danych o użytkownikach SoundCloud na podstawie popularnych imion w różnych regionach, a także na ich polubieniach, co może być przydatne do analizy trendów, popularności utworów czy zachowań użytkowników w różnych lokalizacjach.

Przetwarzanie danych:

Proces przetwarzania danych obejmował ich normalizację oraz wyczyszczeni:

- ujednolicenie zapisu nazw miast
- naprawa unicode
- odfiltrowanie użytkowników, bez żadnych polubień liczba takich użytkowników to 605 dla Polski i 547 dla Wielkiej Brytanii
- ekstrakcję użytkowników
- kalkulacja wartości poszczególnych gatunków słuchanych przez użytkownika
- ujednolicenie nazw gatunków, wyekstrahowanie najważniejszych cech z nazwy i zmapowanie gatunków muzycznych.

Dokumentacja Skryptów do Przetwarzania Danych Muzycznych

Przegląd

Niniejsza dokumentacja opisuje zestaw skryptów wykorzystywanych do przetwarzania i analizy danych muzycznych pozyskanych z SoundCloud za pośrednictwem RapidAPI. Skrypty te umożliwiają czyszczenie, mapowanie, klasyfikowanie i agregowanie danych muzycznych w celu ich dalszej analizy i wykorzystania.

Skrypty

Skrypt Python: Czyszczenie i Agregacja Danych (data_cleaning.py)

Ten skrypt w Pythonie służy do przetwarzania danych użytkowników i ich zapisanych polubień. Wykonuje następujące funkcje:

- Wczytuje dane z plików JSON i konwertuje tekst z Unicode na ASCII.
- Łączy dane użytkowników z ich zapisanymi polubieniami.
- Filtruje użytkowników, wybierając tych z istotnymi danymi.
- Przetwarza polubienia każdego użytkownika, zliczając występowanie gatunków muzycznych.
- Transformuje dane, usuwając zbędne pola i normalizując nazwy miast.
- Agreguje dane według miast i gatunków muzycznych, sortując je według liczby wystąpień.

Skrypt Python: Mapowanie Gatunków Muzycznych (map_genres.py)

Ten skrypt rozszerza funkcjonalność poprzedniego, skupiając się na mapowaniu gatunków muzycznych:

- Definiuje listy gatunków muzycznych i specyficznych podgatunków jazzu.
- Czyści i normalizuje nazwy gatunków muzycznych.
- Mapuje każdy gatunek muzyczny do odpowiedniej kategorii z zdefiniowanych list.
- Przetwarza i mapuje dane muzyczne, kategoryzując je według standardowych gatunków muzycznych.

- Łączy blisko powiązane gatunki, takie jak 'rap' i 'hip-hop', w jedną kategorię.
- Zapisuje zmapowane i zaktualizowane dane do nowych plików JSON.

Skrypt Bash: Wdrażanie i Czyszczenie Danych (deploy-data.sh)

Dodatkowo, dostępny jest skrypt bash, który ułatwia proces wdrażania i czyszczenia danych:

- Sprawdza, czy podano dwa argumenty: ścieżki do plików z polubieniami i danymi użytkowników.
- Kopiuje źródłowe pliki danych do określonych lokalizacji docelowych.
- Uruchamia skrypty Pythonowe do czyszczenia i mapowania danych.
- Przenosi przetworzone dane do określonej lokalizacji.
- Usuwa tymczasowe pliki danych wygenerowane w trakcie procesu przetwarzania.

Użycie

- **Przygotowanie danych**: Przed uruchomieniem skryptu **deploy-data.sh**, należy przygotować pliki z danymi o zapisanych polubieniach i użytkownikach.
- **Wdrażanie danych**: Uruchom skrypt **deploy-data.sh** z odpowiednimi argumentami, aby skopiować dane źródłowe i uruchomić proces przetwarzania.
- **Przetwarzanie danych**: Skrypty Pythonowe będą automatycznie uruchomione przez skrypt bash, przeprowadzając proces czyszczenia i mapowania danych.
- **Analiza danych**: Przetworzone dane są zapisywane w określonym miejscu i są gotowe do dalszej analizy lub wykorzystania.

Wnioski

Zestaw tych skryptów stanowi kompleksowe narzędzie do przetwarzania danych muzycznych, umożliwiając dogłębną analizę i zrozumienie trendów muzycznych w różnych miastach. Skrypty te są szczególnie przydatne w kontekście badań nad trendami muzycznymi, personalizacji rekomendacji muzycznych, czy tworzenia interaktywnych wizualizacji danych muzycznych.

Opis Skryptu do Wdrażania Danych do DynamoDB (deploy_to_dynamodb.py)

Przeznaczenie

Ten skrypt w Pythonie służy do wdrażania przetworzonych danych muzycznych do bazy danych DynamoDB w AWS. Jest on zaprojektowany, aby załadować dane dotyczące gatunków muzycznych związanych z różnymi miastami.

Funkcjonalność

- **Inicjalizacja Klienta DynamoDB**: Skrypt rozpoczyna od utworzenia sesji z profilem **sandbox**, co pozwala na połączenie z usługą AWS DynamoDB.
- **Określenie Tabeli DynamoDB**: Następnie, skrypt określa tabelę DynamoDB o nazwie **CityGenres**, do której będą załadowane dane.
- **Wczytywanie Danych JSON**: Dane są wczytywane z pliku JSON (**GB_DATA_RAP.json**), który zawiera informacje o gatunkach muzycznych dla poszczególnych miast.
- **Funkcja do Wgrywania Danych**: **upload_to_dynamodb** to funkcja, która zajmuje się wgrywaniem danych dla każdego miasta do DynamoDB. W przypadku błędów, funkcja zwraca informacje o nich.
- **Proces Wgrywania**: Skrypt iteruje przez każde miasto w wczytanym pliku JSON, używając funkcji **upload_to_dynamodb** do wgrywania danych do DynamoDB.

Użycie

- **Przygotowanie Środowiska**: Upewnij się, że środowisko Pythona ma zainstalowane **boto3** i sk

onfigurowane połączenie z AWS. 2. **Konfiguracja AWS**: Profil **sandbox** powinien być skonfigurowany w AWS CLI z odpowiednimi uprawnieniami dostępu do DynamoDB.

- **Wczytanie Danych**: Przygotuj plik JSON z danymi, które mają być załadowane do DynamoDB.
- **Uruchomienie Skryptu**: Uruchom skrypt **deploy_dynamodb.py**. Skrypt automatycznie wczyta dane z pliku JSON i wgra je do określonej tabeli DynamoDB.

Wnioski

Ten skrypt jest kluczowym narzędziem w procesie wdrażania danych muzycznych do bazy danych DynamoDB, co umożliwia łatwe przechowywanie, zarządzanie i dostęp do dużych zbiorów danych. Jego zastosowanie jest szczególnie przydatne w środowiskach korzystających z usług chmurowych AWS, pozwalając na integrację z innymi usługami AWS i budowanie skalowalnych aplikacji wykorzystujących dane muzyczne.

Podsumowanie

Kryterium wyboru krajów

Podczas pobierania danych dysponowaliśmy ograniczoną liczbą możliwych requestów do wykonania, do analizy wybraliśmy Polskę – ze względu na to, iż jest to nasz region zamieszkania oraz Wielką Brytanię ze względu na różnorodność społeczeństwa i gatunków muzycznych. Te dwa kraje dostarczają cennych informacji na temat preferencji muzycznych i popularności różnych gatunków.
Zarówno Polska, jak i Wielka Brytania mają silne społeczności muzyczne, z licznymi artystami, producentami i fanami. Analizowanie trendów w tych krajach pozwala dostarczyć informacji o popularności lokalnych talentów oraz globalnych trendów muzycznych. Dodatkowo, Wielka Brytania i Polska są znacznymi rynkami dla platformy SoundCloud. Analiza tych rynków dostarcza wgląd w dynamikę społeczności korzystających z tej platformy.

Napotkane problemy

Pierwotnie planowaliśmy korzystać z publicznego API SoundCloud, zgodnie z dokumentacją udostępnioną przez platformę. Niestety, napotkaliśmy trudności z uzyskaniem dostępu, pomimo prób rejestracji aplikacji i zastosowania metod opisanych w dokumentacji. Liczyliśmy na uzyskanie dostępu od zespołu wsparcia SoundCloud, co ostatecznie nie okazało się możliwe. Zdecydowaliśmy się na wykupienie dostępu do platformy RapidApi, która pośrednio umożliwiła komunikację z API soundclouda, jednak ilość wykupionych requestów nie jest zadowalająca i nie pozwala na ładowanie danych do aplikacji w pojedynczej sesji.

Wnioski

Analizując dane dotyczące gatunków muzycznych w serwisie SoundCloud dla ogółu użytkowników oraz dla konkretnych regionów, Polski i Wielkiej Brytanii, możemy zaobserwować w pewnym stopniu zróżnicowanie geograficzne i regionalne preferencji muzycznych.

Pomimo różnic między Polską a Wielką Brytanią, można zauważyć, że pięć najpopularniejszych gatunków muzycznych (house, electronic, dance, rap/hip-hop, techno) dominują zarówno globalnie, jak i w obu analizowanych krajach. Sugeruje to pewną uniwersalność tych gatunków wśród użytkowników SoundCloud.

W analizie dla Polski i Wielkiej Brytanii pojawiają się różnice w hierarchii popularności niektórych gatunków. Na przykład w Polsce gatunki takie jak techno i pop zdają się być bardziej popularne niż w Wielkiej Brytanii, gdzie z kolei dominuje dance. To może wskazywać na istnienie pewnych lokalnych trendów i upodobań.

Gatunki muzyczne, takie jak rap/hip-hop, mogą odzwierciedlać wpływy lokalnej kultury i języka.

Alternatywne gatunki muzyczne zdają się być popularne zarówno w Polsce, jak i Wielkiej Brytanii. Choć hierarchia nieco się różni, oba kraje wykazują zainteresowanie alternatywnymi nurtami muzycznymi, co może wskazywać na pewne cechy wspólne wśród entuzjastów tego rodzaju muzyki.

Z uzyskanych wyników można zauważyć, że grupy słuchaczy na platformie SoundCloud nie są jedynie skupione geograficznie, ale również obejmują regiony o pewnych cechach wspólnych, takich jak lokalne preferencje muzyczne i wpływy kulturowe. Istnieje jednak również pewna uniwersalność wśród najpopularniejszych gatunków muzycznych, które cieszą się jednakowo wysoką popularnością w różnych krajach.