Aplikacja do porównywania artystów zarejestrowanych w serwisie Spotify

PROJEKT ZALICZENIOWY Z PRZEDMIOTU

SKŁADOWANIE DANYCH W SYSTEMACH BIG DATA

MAKAREWICZ AGATA

WIŚNIEWSKI JACEK

10.01.2022r.

Spis treści

[1 Cel projektu 3](#_Toc92828640)

[2 Zbiory danych 3](#_Toc92828641)

[3 Architektura systemu 3](#_Toc92828642)

[3.1 Konwersja daty zakupu 3](#_Toc92828643)

[4 Opis rozwiązania 3](#_Toc92828644)

[4.1 Pozyskiwanie danych 3](#_Toc92828645)

[4.2 Przetwarzanie danych 3](#_Toc92828646)

[4.3 Składowanie danych 3](#_Toc92828647)

[4.4 Analiza danych (+ generowane widoki wsadowe?) 3](#_Toc92828648)

[4.5 Warstwa prezentacyjna 3](#_Toc92828649)

[5 Testowanie rozwiązania 3](#_Toc92828650)

[6 Podsumowanie 3](#_Toc92828651)

[6.1 Podział pracy 3](#_Toc92828652)

# Cel projektu

Jednym z największych wyzwań dealera samochodowego kupującego używany samochód na aukcji

# Zbiory danych

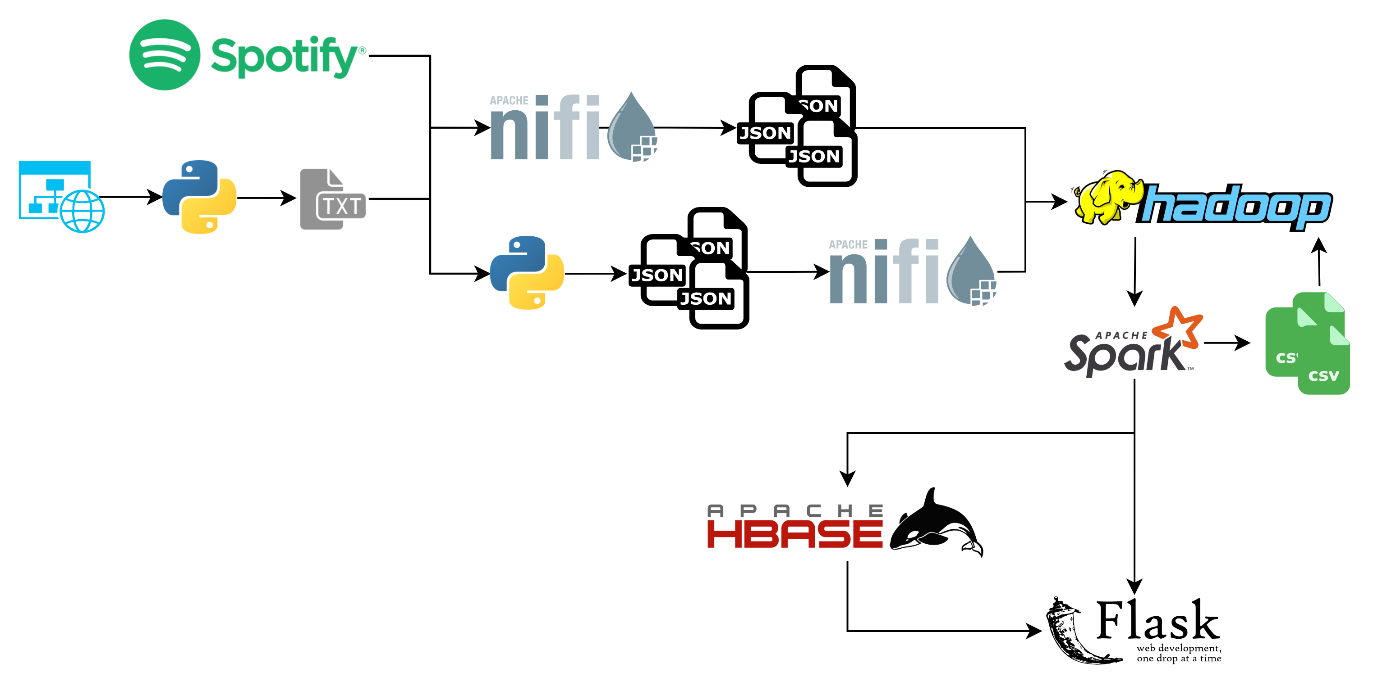
Zbiór danych zawiera informacje o samochodach używanych zakupionych na aukcjach w różnych samochodów, opisanych przy pomocy 33 atrybutów. Informacje można podzielić na kilka grup:

* atrybuty identyfikujące samochody – rok produkcji, wiek, marka, model, wersja, typ, kolor, przekładnia, koła, stan drogomierza, miejsce produkcji (państwo), rozmiar

*Tabela 1: Atrybuty identyfikujące samochody.*

# Architektura systemu

Aaaaa

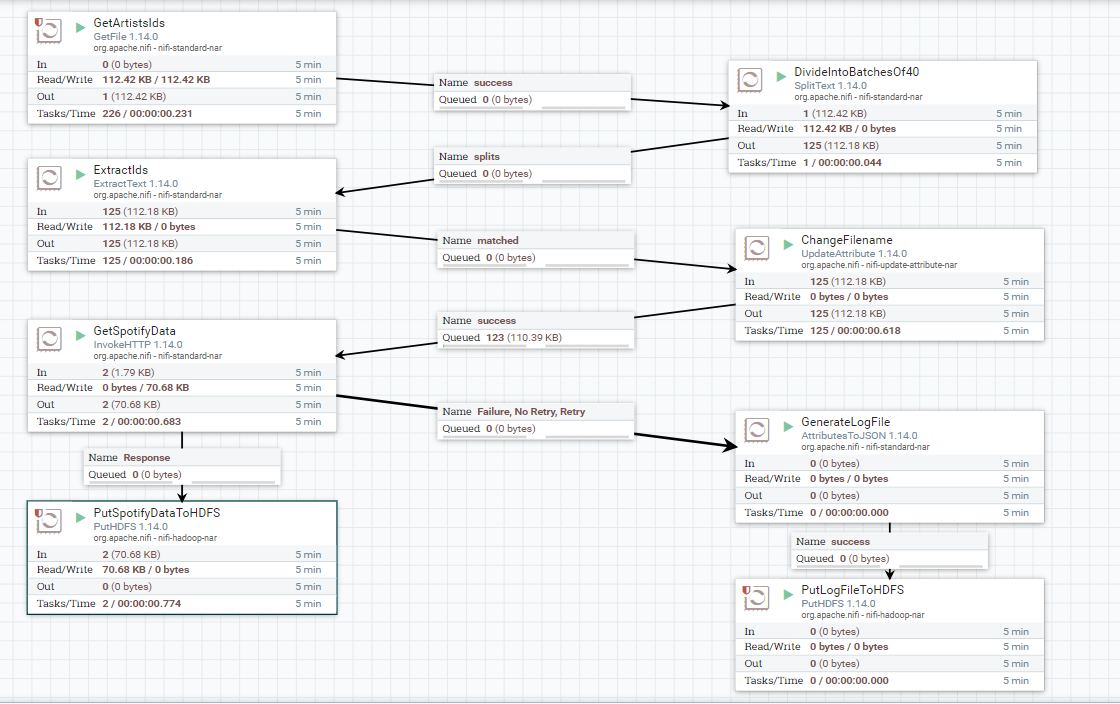


# Opis rozwiązania

Aaaaa

## Pozyskiwanie, przetwarzanie i składowanie danych

Aaaaa

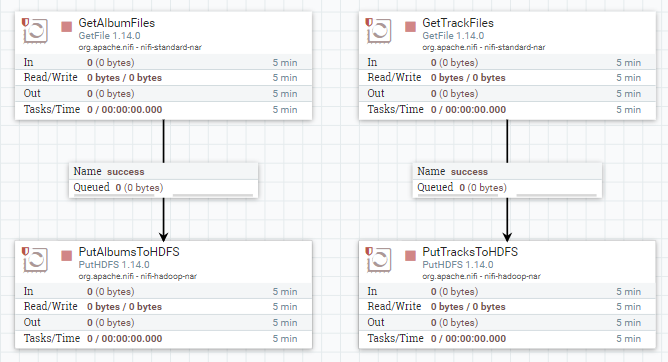


Na początku do przepływu pobierany jest plik tekstowy z identyfikatorami artystów. Plik zawiera 5000 identyfikatorów, po 40 w jednej linii. Taki podział spowodowany jest ograniczeniami Web API Spotify – w jednym zapytaniu nie można podać więcej elementów. Przy pomocy procesora *SplitText* plik dzielony jest na 125 plików przepływu (po 40 identyfikatorów w każdym). Następnie procesor *ExtractText* wydobywa zawartość każdego z plików i przypisuje ją do nich jako atrybut.

///

*InvokeHTTP*, przy pomocy którego wykonywane jest zapytanie do Web API Spotify, z zawartością plików przepływu (identyfikatorami artystów) jako parametrem. Rezultatem są pliki w formacie JSON zawierające informacje o poszczególnych artystach, takie jak . Pliki te ładowane są do dedykowanego folderu w HDFS (*/spotify/artists*) przy pomocy procesora *PutHDFS*.

Dodatkowo do przepływu dołączone zostały dwa procesory – *AttributesToJSON* oraz *PutHDFS* – odpowiedzialne za obsługę błędów. Pierwszy z nich wydobywa atrybuty pliku przepływu takie jak …. do pliku w formacie JSON, a drugi ładuje wygenerowane pliki do HDFS (folder).



Pliki w formacie JSON, z identyfikatorem artysty jako nazwą, ładowane są do odpowiednich folderów w HDFS (*/spotify/albums, /spotify/tracks*) przy pomocy przepływu danych zbudowanego przy pomocy Apache NiFi.

## Analiza danych (+ generowane widoki wsadowe?)

Aaaaa

## Warstwa prezentacyjna

Warstwa prezentacyjna rozwiązania przygotowana została w postaci aplikacji Flask. Składa się z dwóch widoków – *search* oraz *report*. Pierwszy z nich zawiera formularz z jednym polem tekstowym, w którym użytkownik może podać identyfikator artysty. Po przesłaniu formularza generowany jest drugi z widoków, który zawiera raport przygotowany przy pomocy frameworka (?) PySpark.

# Testowanie rozwiązania

Aaaaa

# Podsumowanie

Aaaaa

## Podział pracy

Aaaaa

|  |  |
| --- | --- |
| **Członek zespołu** | **Zakres pracy** |
| Agata Makarewicz | * Pozyskiwanie danych (scraping) * Przepływ danych w Apache NiFi * Przetwarzanie danych źródłowych w Apache Spark * Warstwa prezentacyjna (aplikacja Flask) |
| Jacek Wiśniewski | * Pozyskiwanie danych (spotipy) * Przepływ danych w Apache NiFi * Analiza danych i generowanie widoków wsadowych w Apache Spark * Składowanie danych w Apache HBase |