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

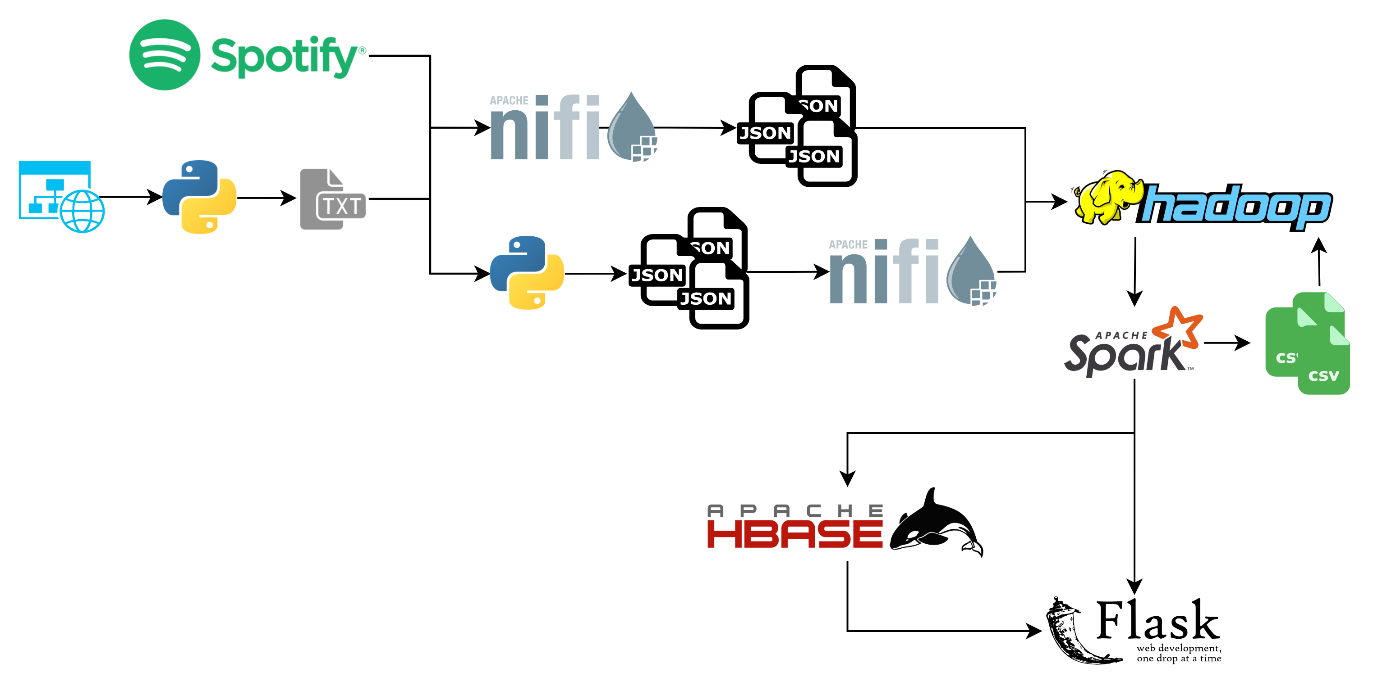
Zbiór danych zawiera informacje o samochodach używanych zakupionych na aukcjach w różnych samochodów, opisanych przy pomocy 33 atrybutów. Informacje można podzielić na kilka grup:

* 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

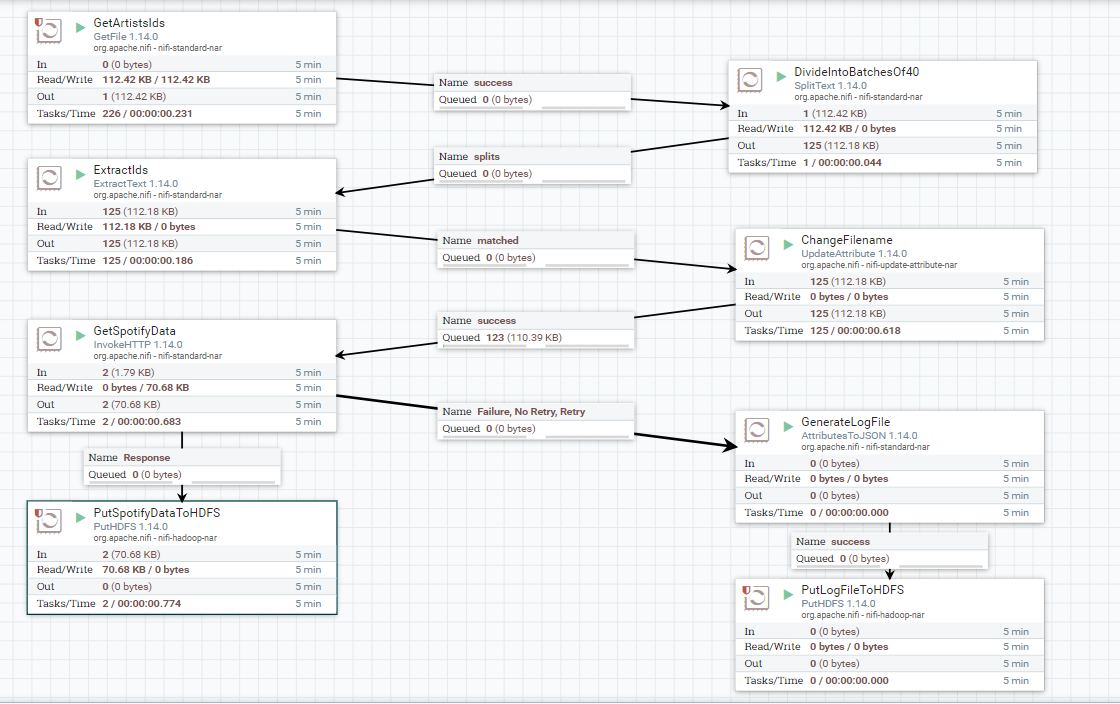


# Opis rozwiązania

Aaaaa

## Pozyskiwanie, przetwarzanie i składowanie danych

Aaaaa

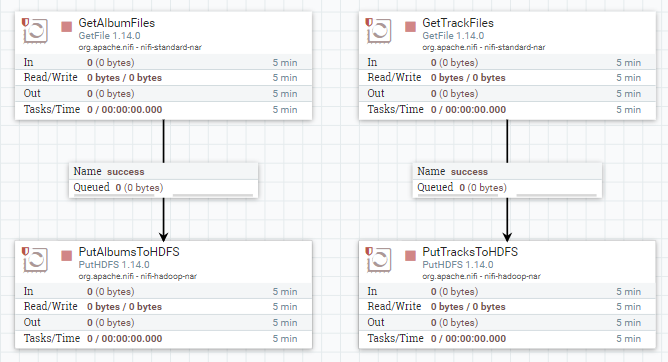


Na początku do przepływu pobierany jest plik tekstowy z identyfikatorami artystów. Plik zawiera 5000 identyfikatorów, po 40 w jednej linii. Taki podział spowodowany jest ograniczeniami Web API Spotify – w jednym zapytaniu nie można podać więcej elementów. Przy pomocy procesora *SplitText* plik dzielony 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.

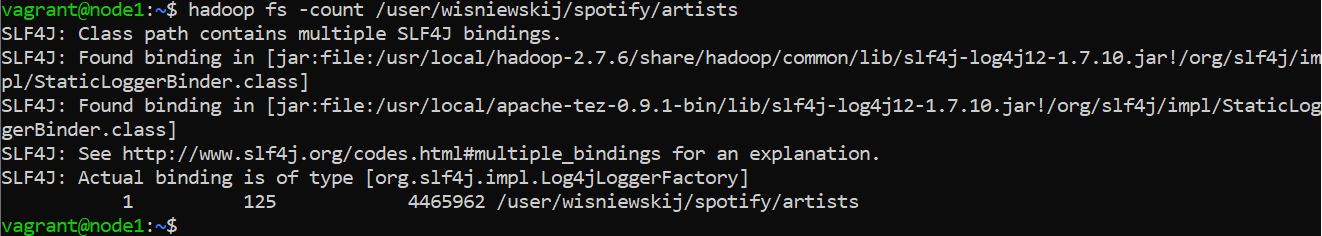
# Testowanie rozwiązania

W tym rozdziale zostały zaprezentowane testy funkcjonalne obejmujące zaimplementowane i skonfigurowane komponenty.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr** | **Cel** | **Podjęte kroki** | **Oczekiwany wynik** |
| 1 | Otrzymanie listy ID najpopularniejszych 5000 artystów według Spotify | Uruchomiony został skrypt *scraping.py*, który pobiera ze strony [*https://kworb.net/spotify/artists.html*](https://kworb.net/spotify/artists.html) ID pierwszych 5000 artystów. | Plik tekstowy zawierający 5000 ID, oddzielonych przecinkami, po 40 w jednej linii. |
| 2 | Załadowanie informacji na temat artystów do HDFS | Uruchomiony został przepływ danych (dedykowany danym o artystach) utworzony w Apache NiFi. | Folder *artists* w HDFS zawiera 125 plików w formacie JSON, o nazwach artists\_\* . |
| 3 | Otrzymanie informacji na temat albumów najpopularniejszych 10 artystów według Spotify | Uruchomiony został skrypt *spotipy\_albums.ipynb* z wygenerowanym na stronie *developer.spotify.com* nowym kluczem autoryzującym. | Plik w formacie JSON zawierający informacje na temat albumów 10 artystów. |
| 4 | Otrzymanie informacji na temat piosenek najpopularniejszych 10 artystów według Spotify | Uruchomiony został skrypt *spotipy\_tracks.ipynb* z wygenerowanym na stronie *developer.spotify.com* nowym kluczem autoryzującym. | Plik w formacie JSON zawierający informacje na temat piosenek 10 artystów. |
| 5 | Utworzenie tabeli zawierającej komplet informacji otrzymanych w poprzednich krokach | Uruchomiony został skrypt *PySparkDataPreprocessing.ipynb*, w którym dane dotyczące artystów, albumów oraz utworów są transformowane i łączone w jedną tabelę. | Na końcu skryptu powinna pojawić się tabela zawierająca dane dotyczące artystów, albumów i utworów. |
| 6 | Zapisanie tabeli z kompletem informacji do HDFS | Potrzebne kroki wykonane przy okazji poprzedniego testu (na końcu wspomnianego skryptu). | Folder *data* w HDFS zawiera pliki CSV o schemacie odpowiadającym stworzonej tabeli. |
| 7 | Utworzenie raportu dla artysty o ID 4O15NlyKLIASxsJ0PrXPfz | Uruchomiony został skrypt */usr/local/hbase/bin/hbase thrift* na wirtualnej maszynie, a następnie został uruchomiony skrypt *PySpark\_report\_test.ipynb* z podanym ID jako parametrem. | Na końcu skryptu powinny pojawić się 3 tabele utworzone przy pomocy funkcji języka Python z wykorzystaniem pakietu PySpark. |
| 8 | Ponowne utworzenie raportu dla artysty o ID 4O15NlyKLIASxsJ0PrXPfz (w celu przetestowania wczytywania raportu z Apache HBase) | Zostały powtórzone kroki z poprzedniego testu. | Skrypt powinien zwrócić te same wyniki co poprzedni test, ale w znacznie krótszym czasie. |
| 9 | Utworzenie raportu dla artysty o ID 1SKeSGQ3LMHYCEgqFGvJbE (Krzysztof Krawczyk) | Zostały powtórzone kroki z poprzednich testów, ze zmienionym wyłącznie ID artysty. | Ponieważ artysta o podanym ID nie należy do grona 5000 najpopularniejszych artystów według wykorzystywanego rankingu, powinny zostać wyświetlone wartości „-1”. |
| 10 | Uruchomienie aplikacji Flask | W terminalu, po przejściu do folderu *spotify*, wywołane zostały kolejno komendy:  *export FLASK\_APP=app*  *python3 -m flask run --host=0.0.0.0* | W przeglądarce pod adresem *localhost:5000/search* wyświetla się strona z polem formularza. |
| 11 | Stworzenie raportu w aplikacji Flask dla artysty o ID 7v49oVVUhvIQG5EK0jkcF7 (Probl3m) | Przy wykonanych krokach z poprzedniego testu, w przeglądarce pod adresem *localhost:5000/search* w polu formularza wpisano podany ID artysty. | W przeglądarce pod adresem *localhost:5000/report* wyświetla się strona z raportem dla danego artysty (3 tabele). |
| 12 | Stworzenie raportu w aplikacji Flask dla artysty o ID 72T7x96EAqN2UWvAgobYfv (Sizzla) (artysta nie jest w zbiorze artystów) | Zostały powtórzone kroki z poprzedniego testu, ze zmienionym wyłącznie ID artysty. | W przeglądarce pod adresem *localhost:5000/report* wyświetla się komunikat o tym, że danego artysty nie ma w bazie. |
| 13 | Stworzenie raportu w aplikacji Flask dla artysty o ID 12345 (nieprawidłowy ID) | Zostały powtórzone kroki z poprzedniego testu, ze zmienionym wyłącznie ID artysty. | W przeglądarce pod adresem *localhost:5000/report* wyświetla się komunikat o tym, że podany ID jest błędny. |

**Rezultaty**

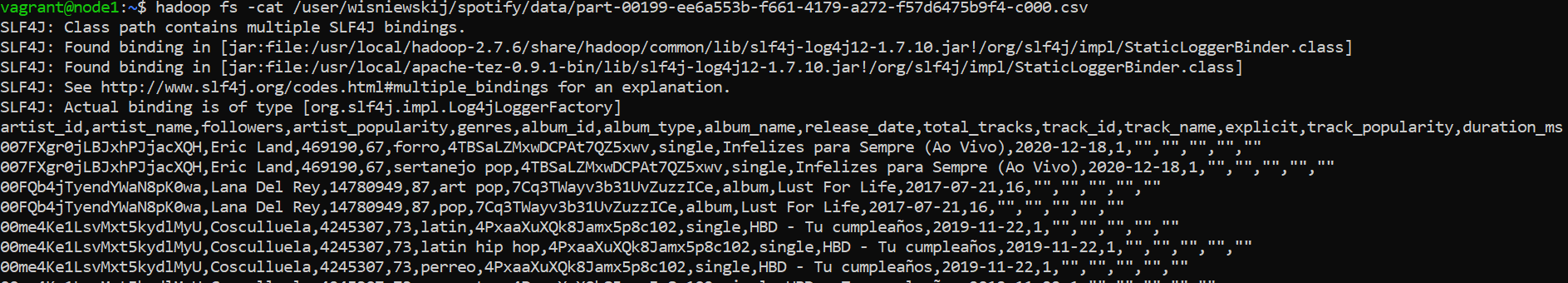
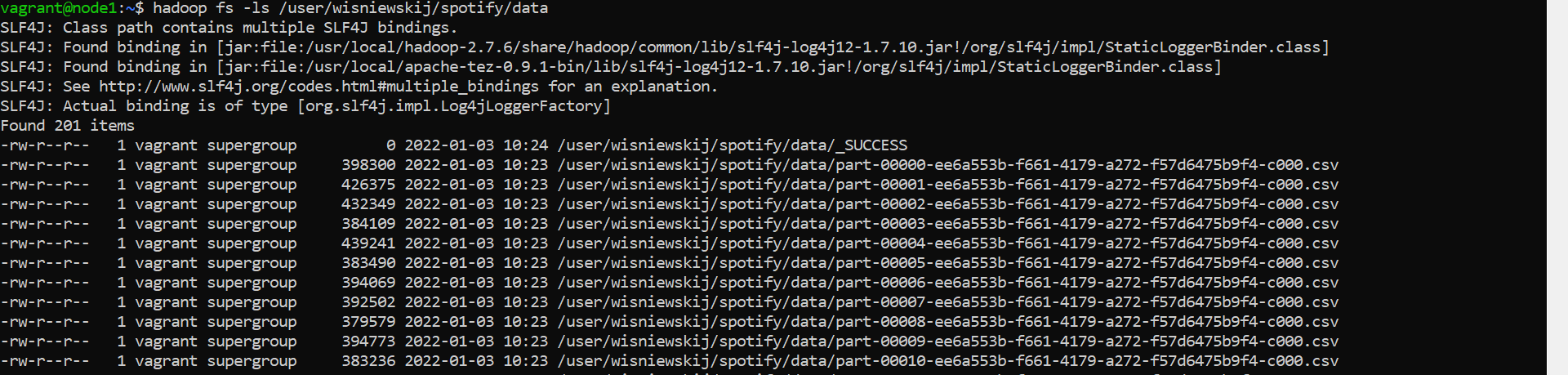
1. Obraz zawierający tekst

   Opis wygenerowany automatycznie
2. Obraz zawierający tekst

   Opis wygenerowany automatycznie
3. Obraz zawierający tekst

   Opis wygenerowany automatycznie
4. Obraz zawierający tekst

   Opis wygenerowany automatycznie
5. Obraz zawierający stół

   Opis wygenerowany automatycznie
6. 
7. Obraz zawierający stół

   Opis wygenerowany automatycznie Obraz zawierający stół

   Opis wygenerowany automatycznie

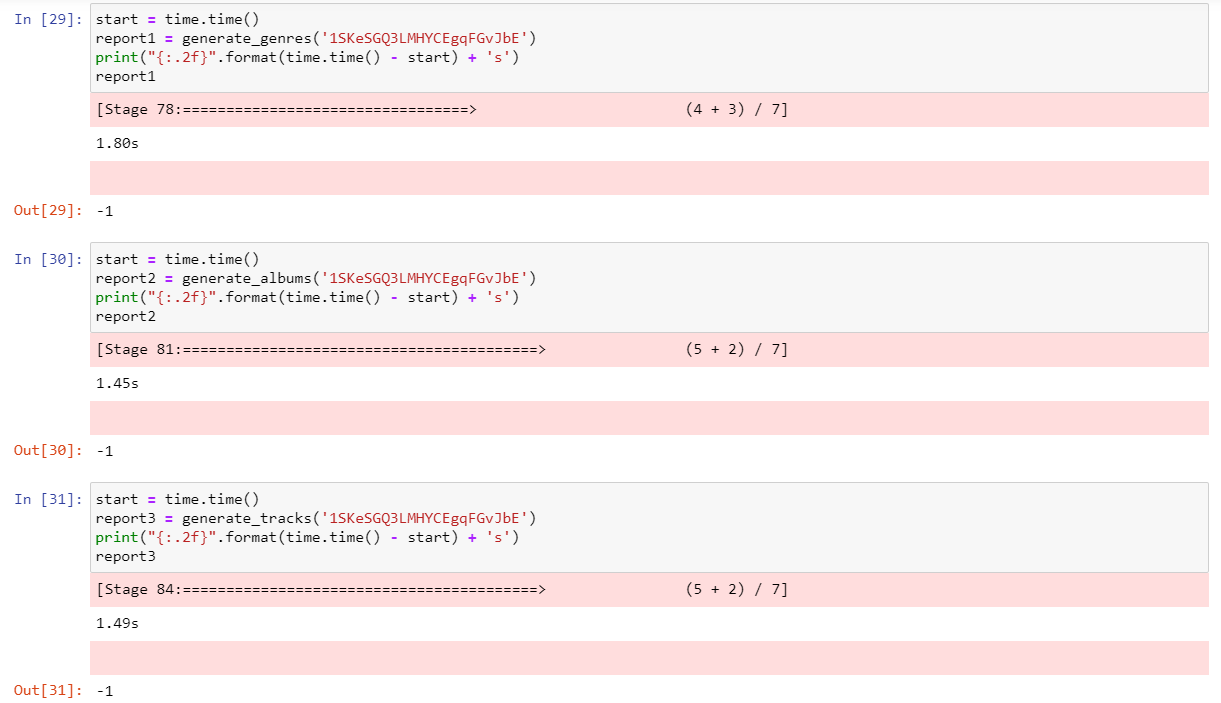
Obraz zawierający stół

Opis wygenerowany automatycznie

1. Obraz zawierający stół

   Opis wygenerowany automatycznie Obraz zawierający stół

   Opis wygenerowany automatycznie  
   Obraz zawierający stół

   Opis wygenerowany automatycznie
2. 
3. Obraz zawierający tekst

   Opis wygenerowany automatycznie
4. Obraz zawierający stół

   Opis wygenerowany automatycznie
5. & 13.  
   Obraz zawierający tekst

   Opis wygenerowany automatycznie Obraz zawierający tekst

   Opis wygenerowany automatycznie

# Podsumowanie

Aaaaa

## Podział pracy

|  |  |
| --- | --- |
| **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 |