

# Składowanie danych w systemach Big Data

## DOKUMENTACJA PROJEKTU

Zespół “*Problem, Plan, Problems*” w składzie:

Przemysław Chojecki, Paweł Morgen, Paulina Przybyłek

# Spis treści:

<b>Spis treści:</b>	<b>2</b>
<b>1 Cel projektu</b>	<b>3</b>
<b>2 Opis źródeł danych wejściowych</b>	<b>3</b>
2.1 Free News API	3
2.2 Twitter API	4
<b>3 Opis architektury rozwiązania</b>	<b>4</b>
3.1 Diagram architektury - działanie stworzonego systemu	5
3.2 Szczegółowe informacje o danych	5
<b>4 Pozyskiwanie i przetwarzanie danych od API z wykorzystaniem Apache NiFi</b>	<b>6</b>
4.1. Schemat działania	6
4.2 Przekształcenia i transformacje	7
4.3 Free News API	8
4.4 Twitter API	9
<b>5 Składowanie danych źródłowych</b>	<b>11</b>
5.1 HDFS	11
5.2 Hive	12
5.2.1 Tabela articles	12
5.2.2 Tabela tweets	12
5.2.3 Tabela publishers	13
<b>6 Analiza danych i generowanie widoków wsadowych - Apache Spark</b>	<b>14</b>
<b>7 Składowanie widoków wsadowych - Apache HBase</b>	<b>14</b>
7.1 Krótki opis tabel w HBase	14
7.2 Przykłady danych dostępnych dla warstwy prezentacyjnej	14
<b>8 Testy funkcjonalne</b>	<b>15</b>
8.1 Pobranie i przetworzenie danych, zapis do HDFS oraz Hive	15
8.2 Wykonanie analizy sparkowej i ładowanie do HBase'a	22
8.2.1 Stan "przed"	22
8.2.2 Stan "po"	23
<b>9 Podsumowanie</b>	<b>25</b>
9.1 Konkluzje	25
9.2 Co się udało?	25
9.3 Co się nie udało?	25
9.4 Możliwe kierunki rozwoju	26
9.5 Podział prac w zespole	26

# 1 Cel projektu

Celem projektu jest zaprojektowanie i wdrożenie narzędzia do przechowywania danych o artykułach prasowych oraz podstawowa analiza o nich oraz o informacjach z nimi związanymi. Projekt skupi się na wydajności, a wdrożone rozwiązanie będzie wysoce skalowalne i będzie w stanie przetwarzać duże ilości danych.

Plan projektu zakłada składowanie i analizę danych o artykułach, tweetach na ich temat oraz informacjach o wydawnictwie, w którym pojawił się artykuł. Zebrane informacje mogą dostarczyć autorom artykułów, publicystom czy nawet wydawnictwom znaczących informacji o ich odbiorcach i preferencjach ich odbiorców, co może doprowadzić do zmiany kreowania treści czy chociażby nagłówków artykułów.

## 2 Opis źródeł danych wejściowych

W tym rozdziale opisane są źródła danych, które były wykorzystane w projekcie. Dostęp do wszystkich wykorzystanych źródeł jest nieodpłatny.

### 2.1 Free News API

To API jest udostępnione przez newscatcherapi (<https://newscatcherapi.com>) jako darmowe do wykorzystania niekomercyjnego. Dane udostępniane są z myślą o niezależnych deweloperach, hakerach-amatorach, studentach oraz data scientist'ach.

Istnieją dwa rodzaje użytkowników - podstawowy (Basic) oraz uczestnik (Contributor). Dostęp do obu jest darmowy, jednak o status uczestnik trzeba się dodatkowo starać. Niestety, autorom niniejszej pracy nie udało się zdobyć statusu uczestnika mimo wypełniania formularzy oraz wysyłania maili. Dlatego rozwiązanie powstało w oparciu o użytkownika podstawowego. Z tej perspektywy napisana będzie reszta niniejszego raportu.

API udostępnia użytkownikom dostęp do informacji o artykułach nie starszych niż siedem dni. Informacje, jakie udostępnia to m.in. tytuł, autor, pierwsze pięćset znaków treści artykułu (nazywane dalej "podsumowaniem artykułu"), informacji czy artykuł zawiera opinie autora, nazwę konta wydawnictwa w serwisie Twitter.

W celu uzyskania dostępu do wyżej wymienionych danych należało formułować zapytania REST z odpowiednim tekstem, którego użytkownik szuka w artykule (dalej nazywany kwerendą, ang. query).

Niestety, to API nie jest idealne, co nie dziwi biorąc pod uwagę jego darmowość. Jedną kwestią jest utrudniony dostęp do API ze względu na problem w skomunikowaniu się z administratorami API w celu uzyskania statusu uczestnika. Drugą kwestią jest jakość uzyskiwanych danych. W czasie pracy nad projektem odkryto, że często artykułom brakuje informacji o nazwie konta wydawnictwa w serwisie Twitter, a zdarza się, że jest ona podana błędna. Poza tym, czasem treść artykułu (podsumowanie) nie zawiera odpowiedniej treści

artykułu, co sugeruje, że program przeszukujący Internet i zbierający informacje dla tego API nie działa idealnie.

Jednakże napotkane problemy nie uniemożliwiły korzystania z tego źródła danych.

## 2.2 Twitter API

To API jest udostępniane przez właściciela popularnego serwisu typu medium społecznościowe, Twitter ([twitter.com](https://twitter.com)).

Właściciel platformy udostępnia API dla swojego serwisu, aby udostępnić deweloperom z całego świata łatwą integrację swoich produktów z serwisem Twitter.

Twitter API ma wiele wersji i poziomów dostępu - niektóre odpłatne. W tym projekcie wykorzystano darmową wersję dostępu o numerze v1.1. Z tej perspektywy napisana będzie reszta niniejszego raportu.

API serwisu Twitter wykorzystano na dwa różne sposoby. Jednym z nich było wykorzystanie dostępu do informacji o tweetach nie starszych niż siedem dni, a drugim dostępu do informacji o użytkownikach serwisu.

Pobierano informacje o tweetach na temat danego artykułu. Pobierano również informacje o użytkownikach Twittera - wydawnictwach publikujących artykuł.

W celu uzyskania dostępu do wyżej wymienionych danych należało formułować zapytania REST z odpowiednim tekstem, którego użytkownik szuka, bądź nazwą poszukiwanego użytkownika serwisu.

W czasie pracy nad projektem nie natrafiono na żadne problemy dotyczące tego źródła danych poza problemem integracyjnym szerzej opisany w sekcji [NiEi](#).

## 3 Opis architektury rozwiązania

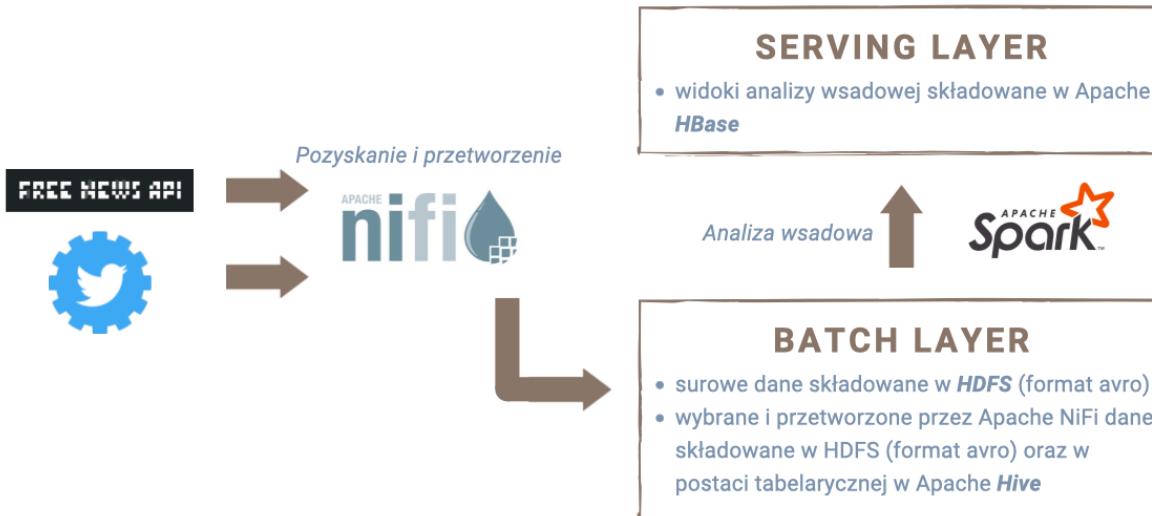
W tym rozdziale opisano wykonane rozwiązanie. Rozwiązanie było zaprojektowane z myślą o dużym wolumenie danych mimo, iż źródła danych w użytych wersjach nie pozwalały na zdobycie takowych.

Architekturę zaprojektowano z myślą o zapewnianiu najwyższej jakości rozwiązania oraz prędkości przetwarzania potencjalnie dużych wolumenów danych. Z tego powodu zastosowana architektura wzorowana była na architekturze lambda. Do pełni implementacji architektury lambda brakuje jedynie implementacji Speed Layer, o czym więcej informacji znajduje się w rozdziale [Możliwe kierunki rozwoju](#).

Dostarczono zatem warstwy Batch Layer oraz Serving Layer, a także etap pobierania i zapisywania danych oraz analizy wsadowej.

### 3.1 Diagram architektury - działanie stworzonego systemu

Na poniższym obrazku zaprezentowany jest schemat rozwiązania. Widnieje na nim schematyczna reprezentacja zaimplementowanego rozwiązania.



Na połączeniu warstw ze źródłami danych znajduje się Apache NiFi, który dokonuje autoryzacji w obu ze źródeł danych. Zapisuje on surowe dane do HDFS w formacie AVRO. Surowe dane są także przetwarzane, a wyniki tego przetworzenia są zapisywane w dwóch miejscach - jako pliki na HDFS w formacie AVRO, ale również w formie tabel Hive.

Analiza na tabelach Hive wykonywana jest przez Apache Spark, a wynik tej analizy zapisywany jest w Apache HBase w warstwie Serving Layer w formie dwóch tabel.

Miedzy Batch Layer a Apache Spark oraz między Apache Spark a Serving Layer również wykorzystywany jest Apache NiFi, który dba o przesył danych między tymi środowiskami (wczytuje i zapisuje dane). Nie uwzględniono go na diagramie w celu uproszczenia prezentowanego schematu.

W poniższych rozdziałach opisano szczegóły każdego z fragmentów architektury. Szczegółowe informacje na temat wykorzystanych źródeł danych znajdują się w rozdziale [Opis źródeł danych wejściowych](#).

### 3.2 Szczegółowe informacje o danych

Wykorzystane API pozwalają na dostęp do danych raz w tygodniu. Szacujemy pobieranie informacji o 20 000 artykułów każdego tygodnia. Szacunki pokazują, że będzie to się przekładało na 12 000 aktualizacji informacji o wydawnictwach oraz 450 000 tweetów.

Zgodnie z praktyką rozwiązań informatycznych, realia pokazują, że błędy się zdarzają. Dlatego rozwiązanie przygotowane jest do możliwości edycji - wykonania prostych zmian przez administratora. Przewidywane jest, że w początkowej fazie działania wyłapane zostaną wszystkie poważne błędy i już po miesiącu system działał będzie całkowicie bez ingerencji człowieka.

Stosowanie architektury lambda pozwala na skalowalność, ale powoduje skomplikowanie rozwiązania. Przykładowo, do załadowania wyników analizy ze sparka do HBase wykorzystany jest NiFi, który jedynie wywołuje dwa procesory.

Dane pobierane z API przychodzą w formie plików JSON, a wszystkie przechowywane pliki są w formacie AVRO.

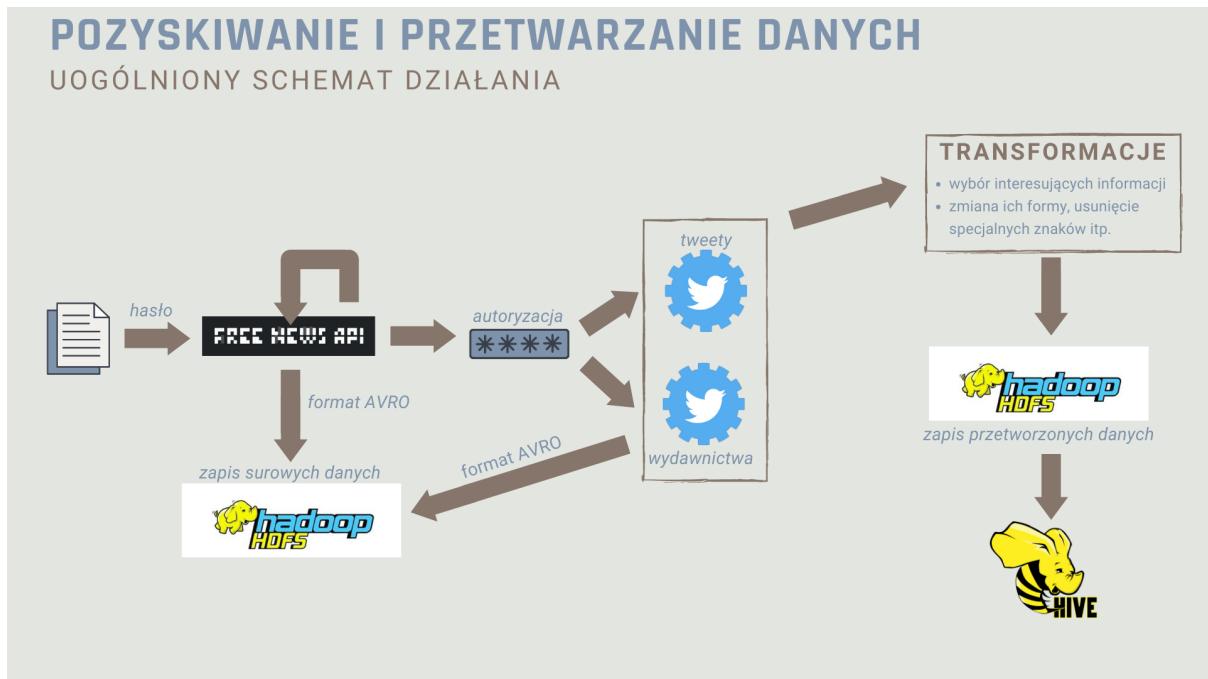
## 4 Pozyskiwanie i przetwarzanie danych od API z wykorzystaniem Apache NiFi

Apache NiFi został użyty w celu połączenia źródeł danych z warstwami dostępowymi. Został on wybrany ze względu na swoją popularność w tego typu rozwiązańach oraz ze względu na skalowalność i wysoką wydajność rozwiązań, które z niego korzystają.

### 4.1. Schemat działania

Flow w Apache NiFi został zaprojektowany w taki sposób, aby być prosty w zrozumieniu oraz w edycji. Aby ewentualne zmiany architektury w przyszłości były możliwe szybkie do wdrożenia nawet przez zespół nie biorący udział w tworzeniu tego rozwiązania.

Dla uproszczenia komunikacji stworzony został Schemat działania flow w aplikacji Canva:



Widać na nim kluczowe fragmenty przetwarzania zapytań oraz przetwarzania danych a także moment przesłania ich do warstwy Barch Layer.

NiFi został skonfigurowany tak, aby regularnie, raz w tygodniu pobierać nowe dane ze źródeł danych, gdyż w użytych konfiguracjach udostępniają one swoje zasoby na siedem dni w

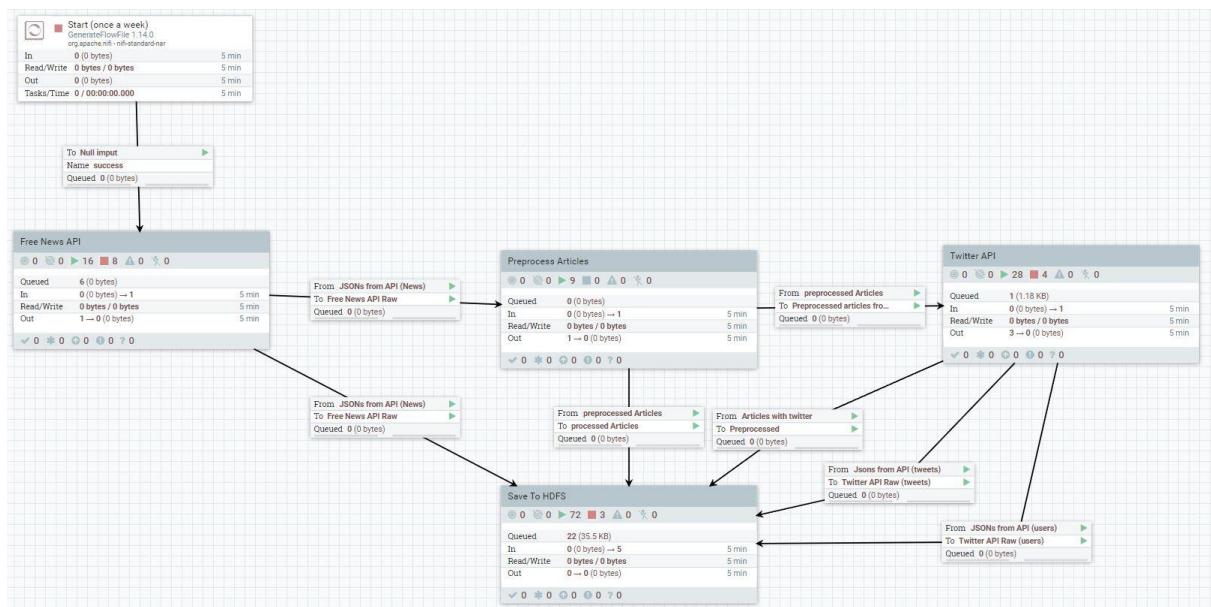
przeszłość. To rozwiązanie zapewnia dostęp do wszystkich możliwych danych, używając minimalnych potrzebnych zasobów komputerowych.

Pobrane surowe dane ze wszystkich źródeł danych są zamieniane z formatu JSON na format AVRO i zapisywane na HDFS.

Pobrane surowe dane są dodatkowo przetwarzane w NiFi, a na końcu tego procesu ustrukturyzowane do postaci trzech tabeli. Tabele te są zapisywane w dwóch niezależnych systemach - jako surowe pliki typu AVRO oraz jako tabele w Apache Hive.

Przed uzyskaniem dostępu do Twitter API, NiFi musi uzyskać autoryzację. Niestety, NiFi nie udostępnia dedykowanego procesora do tego celu. Trzeba było wykonać tą autoryzację ręcznie przy pomocy kodu w Groovy.

Poniżej przedstawione jest przykładowe przejście całego procesu ładowania, przetwarzania oraz zapisu danych.



Przykładowe wywołanie flow w NiFi.

Wszystkie pobrane jak i przetworzone dane spływają do grupy zapisującej.

Widzimy po prawej, że jeden z artykułów jest w kolejce. Oznacza to, że w czasie pobierania danych z Twittera nastąpił błąd. Okazało się, że Free News API udostępniło ten artykuł z błędna informacją o koncie na Twitterze.

Widzimy na dole, że 22 flowfile oczekują w kolejce. Jest to spowodowane tym, że oczekują one, aż zbierze się ich większa ilość, aby zostać wspólnie zapisane na HDFS.

## 4.2 Przekształcenia i transformacje

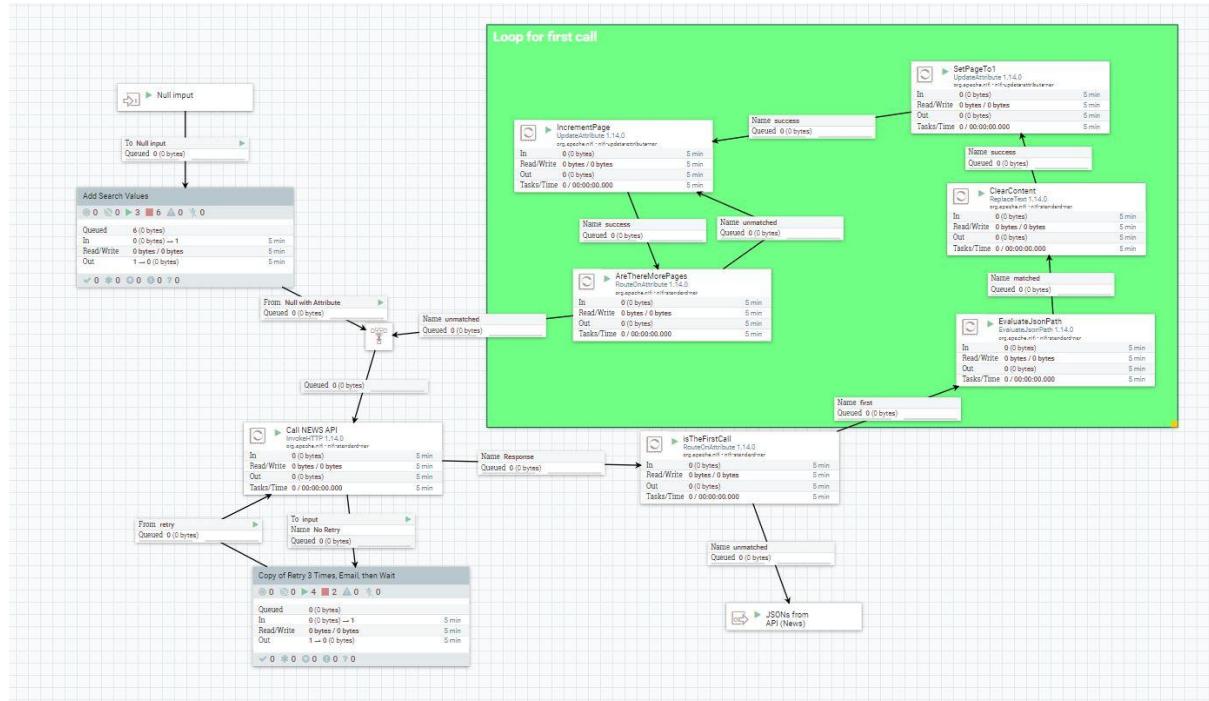
W ramach stworzonego procesu NiFi możliwe jest wiele operacji na flowfiles. Są to m.in.:

1. Zapytanie do Free News API oraz Twitter API (wymaga automatycznej autoryzacji wykonywanej za pomocą Groovy)

2. Przekształcenie plików typu JSON do typu AVRO.
3. Zapis plików na HDFS.
4. Filtrowanie odpowiedzi od API posiadających odpowiednie pola (np. twitter publisher account).
5. Ustawianie timestamp dla utworzonych tabel.

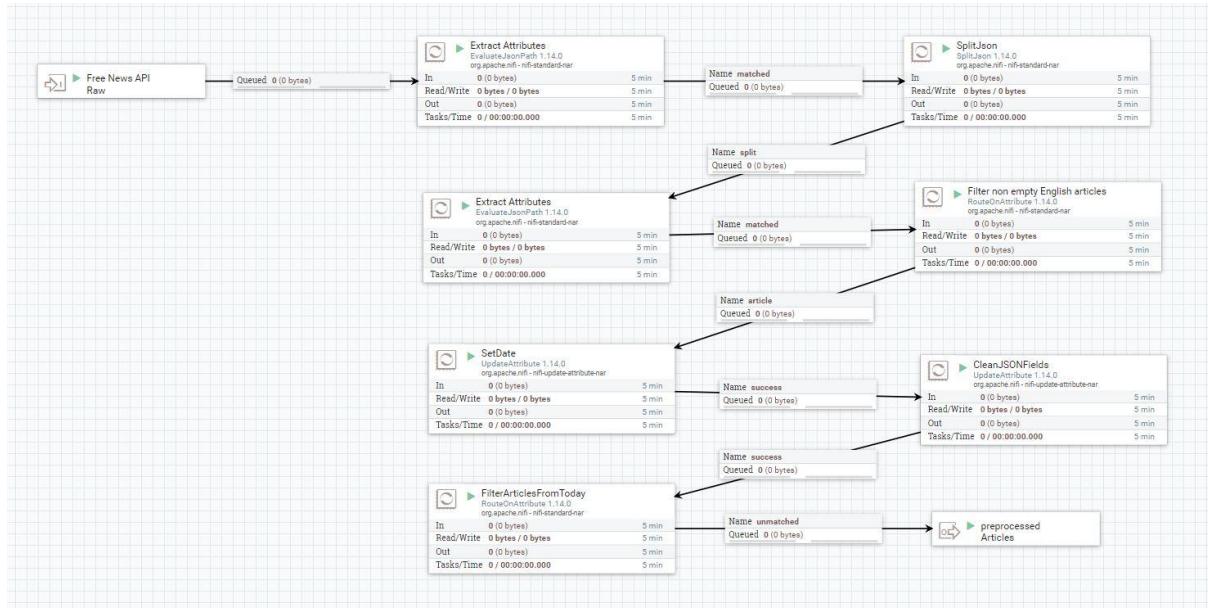
### 4.3 Free News API

W pierwszej części pobierane są artykuły z Free News API. Pierwsze zapytanie dostarcza informacji o tym ile zapytań należy wykonać, a następnie na zielono zaznaczono pętlę, która dostarcza flowfiles w liczbie równej ilości zapytań do Free News API, które są potrzebne do pobrania wszystkich danych.



Następnie pobrane surowe dane w formie JSON idą w dwa miejsca: do zapisu surowych danych oraz do przetworzenia.

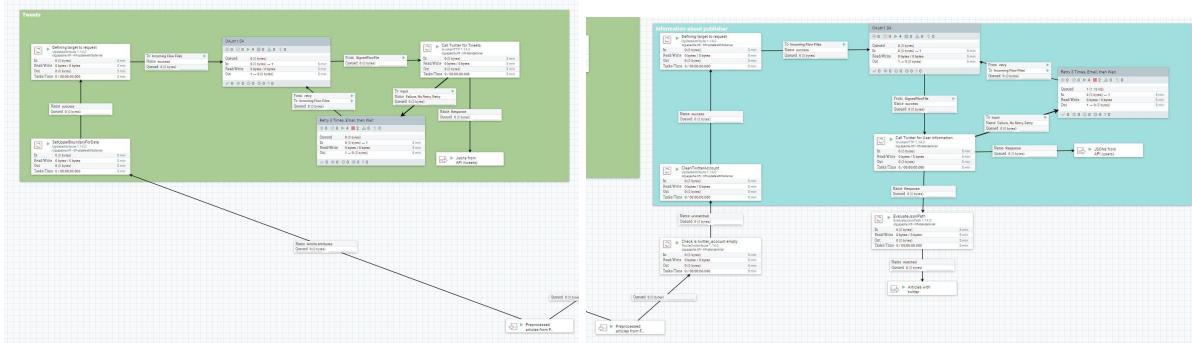
Przetworzenie danych polega na wyciągnięciu z JSONa informacji, które będą składowane.



Następnie przetworzone artykuły trafiają w dwa miejsca. Do zapisu do HDFS oraz do modułu wywołującego Twitter API.

## 4.4 Twitter API

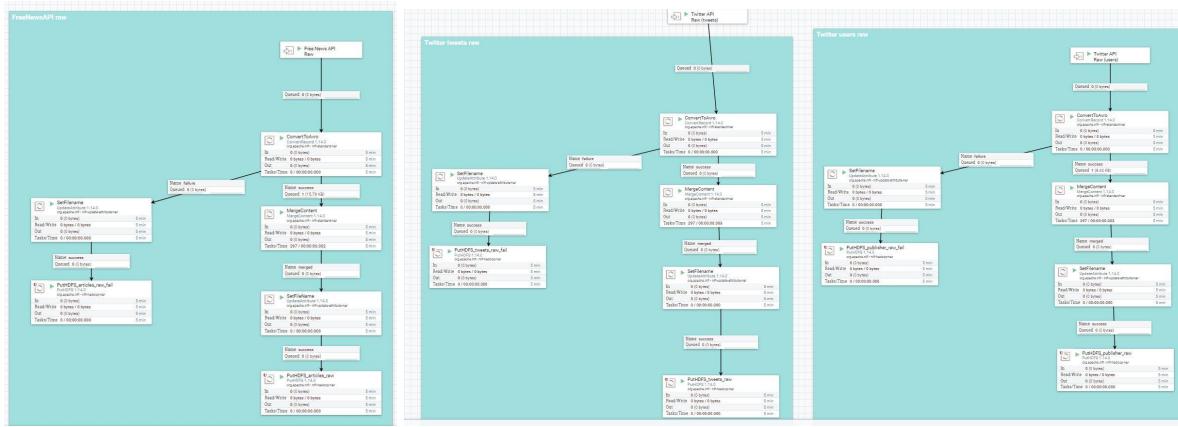
W module Twitter API flowfiles z przetworzonymi artykułami trafiają w dwa miejsca - do modułu pobierającego treści tweetów (zielone tło) oraz do modułu zbierającego informacje o wydawnictwie (turkusowe/niebieskie tło).



## 4.5 Surowe dane

Z obu tych modułów pobrane dane trafiają w dwa miejsca - do modułu zapisującego surowe dane (przekształcające je uprzednio z formatu JSON do AVRO) oraz do modułu przetwarzającego. Tutaj również przetworzenie polega na wyborze interesujących informacji do późniejszej analizy.

Wszystkie trzy moduły zapisujące mają turkusowe/niebieskie tło:

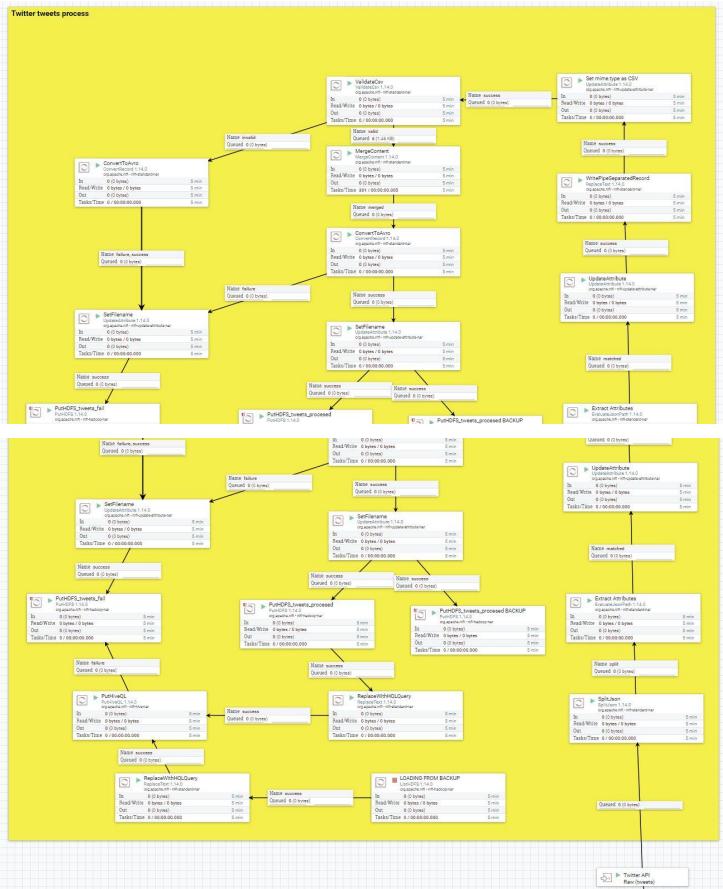


Moduły strukturyzujące dane do formy tabelarycznej, a potem zapisujące je do HDFS oraz Hive, są na żółtym tle (rysunki poniżej). Zapis do Hive polega na zapisie danych do HDFS i podniesieniu ich przez NiFi w celu przekształcenia do odpowiedniej postaci możliwej do zapisu w uprzednio stworzonych tabelach Hive.

## 4.6 Przetworzone dane - HDFS oraz Hive

Przetworzone dane są zapisywane w dwóch miejscach: na HDFS w postaci AVRO oraz w Hive.





Każdy z tych modułów ma procesor "LOADING FROM BACK UP" umożliwiający załadowanie danych z tak zwanego back-up'u bez potrzeby przetwarzania surowych danych całym NiFi-flow.

## 5 Składowanie danych źródłowych

Warstwa batchowa wykorzystuje HDFS oraz Hive do składowania danych w systemie.

### 5.1 HDFS

Do składowania danych użyto systemu HDFS przede wszystkim ze względu na jego skalowalność. Pozwala on bowiem na wygodne składowanie dużej ilości danych. Na dodatek system ten jest popularny i pozostałe używane w tym projekcie systemy (NiFi, Spark) natywnie się z nim komunikują.

Użyty system HDFS jest łatwo skalowalny wszerz oraz zapewnia replikację danych.

Przypominając, składowane w nim dane to zarówno surowe informacje z API jak i przetworzone dane przez NiFi.

## 5.2 Hive

Dane w warstwie batchowej w Hive są składowane w trzech tabelach Hive'owych: *articles*, *tweets* oraz *publishers*.

### 5.2.1 Tabela *articles*

Tabela *articles* przechowuje informacje o zebranych artykułach, jak tytuł, autorzy czy temat. Jej struktura jest zaprezentowana poniżej:

Kolumna	Typ	Opis
id	string	Identyfikator artykułu od Free News API
published_date	string	Data opublikowania artykułu
title	string	Tytuł artykułu
author	string	Nazwisko autora artykułu
topic	string	Temat artykułu (przypisany przez Free News API)
country	string	Kraj którego dotyczy artykuł
language	string	Język, w którym artykuł został opublikowany
is_opinion	boolean	Czy artykuł został oznaczony jako opinia przez wydawcę
query	string	Zapytanie jakiego użyto do pobrania artykułu z Free News API
summary	string	Pierwsze 500 znaków artykułu
my_timestamp	bigint	Timestamp

Na poniższym obrazku przedstawiono przykładowe wiersze z tabeli *articles* otrzymane w wyniku zapytania w beelinie. Dla lepszej czytelności kolumna *summary* została pominięta.

0: jdbc:hive2://localhost:10000/ > SELECT id,published_date,title,author,topic,country,language,is_opinion,query,my_timestamp FROM articles LIMIT 5;									
id	published_date	title	author	topic	country	language	is_opinion	query	my_timestamp
cF8049c89481f1a9cb26fb867a719aef	2022-01-15	Fire breaks out at NJ chemical plant 'the worst that I've ever seen'	NULL	news	US	en	false	the	1642861434487
76c66e14b9596bc5f496980259aa4cb	2022-01-15	Sharecare Inc.   When Finance Editors	news	en	false	the	1642861434491		
c068132b232b3de6cc6131af0e0b464685	2022-01-15	Missouri Alert System Warns Citizens To Be On The Lookout For The Joker	167.9 YD   Scott Stevens	news	US	en	false	the	1642861434492
p9e7ab73bb55466e08bf74354df3805b	2022-01-21	yoboo Launched New Nursing Pad and Entered Southeast Asian Market	NULL	news	US	en	false	the	1642861434485
1ad3ca699e79a700dbad985ee89805	2022-01-17	Scottish government in line for near-E70m payday after windfarm auction	Jillian Ambrose	sport	GB	en	false	the	1642861434492

### 5.2.2 Tabela *tweets*

Tabela *tweets* przechowuje dane o wpisach na Twitterze na temat artykułów z tabeli *articles*. Jej struktura jest zaprezentowana poniżej:

Kolumna	Typ	Opis

id	string	Identyfikator; połączenie article_id oraz tweet_id
article_id	string	Identyfikator artykułu wspomnianego w tweecie
tweet_id	bigint	Identyfikator dostarczony przez Twittera
tweet_text	string	Przeczyszczony tekst tweeta
my_timestamp	bigint	Timestamp

Na poniższym obrazku przedstawiono przykładowe wiersze z tabeli *tweets* otrzymane w wyniku zapytania w beelinie.

```
0: jdbc:hive2://localhost:10000/> SELECT * FROM tweets LIMIT 5;
+-----+-----+-----+-----+-----+
| tweets.id | tweets.article_id | tweets.tweet_id | tweets.tweet_text | tweets.my_timestamp |
+-----+-----+-----+-----+-----+
| 0x40879a923b0d0a6930644416695d_14824534532265984 | 0x40879a923b0d0a6930644416695d | 14824534532265984 | RT smh tomic raising glasses and eyebrows before COVID diagnosis lucky_manly https_t.co_5t8irC1lgn | 1642862599569
| 14b9311b7f8935b7c0854c09c3e53fa_1483842699941847648 | 14b9311b7f8935b7c0854c09c3e53fa | 1483842699941847648 | Never run empty with these winning pre-workout supplements https_t.co_wL2hQyMpn | 1642862611592
| 24bed4d796a388569f1017875f53838 | 1483328710040207473 | 1483328710040207473 | When is bin day in london. How to check your borough's collection day https_t.co_VekjBj0mR | 1642862575588
| 24bed4d796a388569f1017875f53838 | 1483319937523492145 | 1483319937523492145 | London When is bin day. How to check your borough's collection day https_t.co_vL2VinetR | 1642862575589
| 1cd9b9cc598bb940ef33df9838929d_148274609325479788 | 1cd9b9cc598bb940ef33df9838929d | 148274609325479788 | Welcome to January's Observer Food Monthly my gran would laugh to see milt extract being hailed as any modern che... http_t.co_rv768mESF | 1482863597586
+-----+-----+-----+-----+-----+
5 rows selected (0.447 seconds)
```

### 5.2.3 Tabela *publishers*

Tabela *publishers* zawiera dane o wydawcach artykułów z tabeli *articles*. Część przechowywanych danych zmienia się w czasie (ilość followersów, ilość tweetów autorstwa wydawcy). Jej struktura jest zaprezentowana poniżej:

Kolumna	Typ	Opis
id	string	Identyfikator
article_id	string	Identyfikator artykułu opublikowanego przez tego wydawcę
twitter_id	bigint	Identyfikator konta wydawcy na Twitterze
twitter_account	string	Nazwa konta wydawcy na Twitterze
publisher_name	string	Nazwa wydawcy
location	string	Lokalizacja wydawcy
followers_count	int	Ilość followersów wydawcy
list_count	int	Ilość list twitterowych, jakie wydawca utworzył na swoim koncie
number_of_tweets	int	Ilość tweetów z konta wydawcy

Poniżej przykładowe wiersze z tabeli *publishers* otrzymane w wyniku zapytania w beelinie.

```
0: jdbc:hive2://localhost:10000/> SELECT * FROM publishers LIMIT 4;
+-----+-----+-----+-----+-----+-----+-----+-----+
| publishers.id | publishers.article_id | publishers.twitter_id | publishers.twitter_account | publishers.publisher_name | publishers.location | publishers.followers_count | publishers.list_count |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1ba05ecbdcc6dd62a1294d9853f648988_7113672 | 1ba05ecbdcc6dd62a1294d9853f648988 | 7113672 | businesswire | Business Wire | San Francisco, CA | 71904 | 2430
| 30538 | 30538 | 30538 | 30538 | 30538 | 30538 | 30538 | 30538
| d31c64262e99c5642c348948db9df2_23859499 | d31c64262e99c5642c348948db9df2 | 23859499 | SeekingAlpha | Seeking Alpha | New York, NY | 195554 | 4866
| 44ea89026a20d8c92a51213b128f2_18639734 | 44ea89026a20d8c92a51213b128f2 | 18639734 | Nasdaq | Nasdaq | NULL | 709686 | 2768
| 43ae9fd7f75af5bcce55212786fd1_19388029 | 43ae9fd7f75af5bcce55212786fd1 | 19388029 | Yahoo | Yahoo | Sunnyvale CA | 1418298 | 8387
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows selected (0.321 seconds)
```

# 6 Analiza danych i generowanie widoków wsadowych - Apache Spark

Dane przetworzone przez NiFi zostają poddane przykładowej analizie za pomocą narzędzia Apache Spark. W toku pracy za pomocą biblioteki *spark-nlp* zostaje wyznaczony sentyment tytułu dla każdego artykułu. Rezultatem analizy są dwie tabele agregujące dane odpowiednio po wydawcach i tematach. Dla obu tabel zostają wyznaczone miarki: sumy artykułów, procentu artykułów z pozytywnym i negatywnym sentymentem oraz sumy tweetów na temat artykułów od danego wydawnictwa/na dany temat. Ponadto dla wydawnictw jest zawarta informacja o procencie artykułów będących opinią oraz o liczbie followersów.

# 7 Składowanie widoków wsadowych - Apache HBase

Wyniki analizy w Sparku są zapisywane do plików tymczasowych, które następnie są podnoszone przez NiFi i ładowane do HBasowych tabel: *publishers* oraz *topics*.

## 7.1 Krótki opis tabel w HBase

Obie tabele w HBase mają po trzy rodziny kolumn (ang. column families): *name*, *article\_stats* oraz *twitter\_stats*. W rodzinie *name* znajdują się odpowiednio nazwa tematu lub wydawcy. Część miarek (wygenerowane statystyki) związana z twitterem (ilość tweetów oraz followersów w przypadku wydawców) związana jest z *twitter\_stats*, a pozostałe miarki - z *article\_stats*.

## 7.2 Przykłady danych dostępnych dla warstwy prezentacyjnej

Przykładowe wiersze z tabelek HBasowych zaprezentowano na screenshotach poniżej. Wczytano je za pomocą Pythonowej biblioteki *happybase* do obiektów klasy *pd.DataFrame* (struktura tych tabel na to pozwalała).

```
In [66]: 1 print(topics.head())
           article_stats:articles_with_negative_sentiment_fraction      timestamp \
0                           0.0          1642947921065
1                           0.341         1642947921598
2                           0.357         1642947921631
3                           0.2          1642947921270
4                           0.274         1642947921199

           article_stats:articles_with_positive_sentiment_fraction article_stats:total_published_articles      name:topic \
0                           1.0                      1                     beauty
1                           0.339                     528                  business
2                           0.393                     168                economics
3                           0.425                      40                  energy
4                           0.46                      113    entertainment

           twitter_stats:tweets_mentioning_articles_sum
0                           0
1                         6279
2                          619
3                          296
4                          447
```

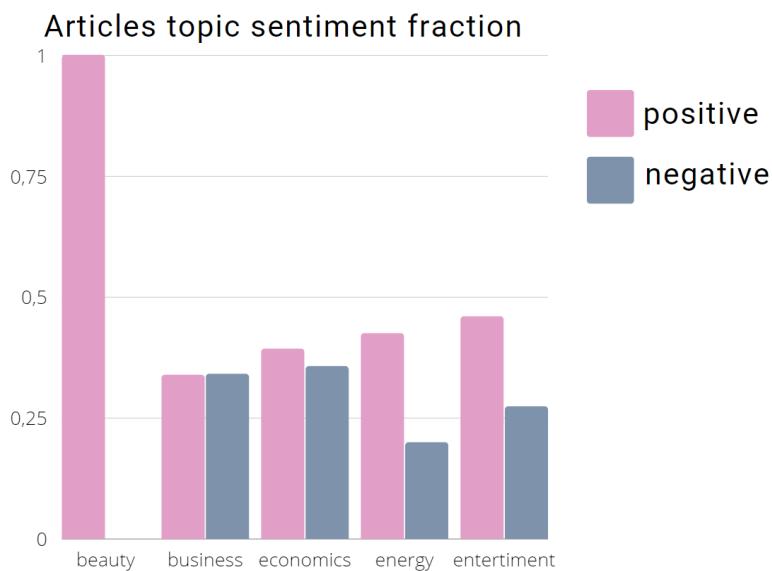
```
In [63]: 1 print(publishers.head())
           article_stats:articles_with_negative_sentiment_fraction      timestamp \
0                           0.0          1642947801727
1                           0.0          1642947801727
2                           0.8          1642947801439
3                           1.0          1642947801727
4                           0.0          1642947801010

           article_stats:articles_with_positive_sentiment_fraction article_stats:published_opinions_fraction \
0                           0.0                      0.0
1                           0.0                      0.0
2                           0.2                      0.0
3                           0.0                      0.0
4                           0.5                      0.0

           article_stats:total_published_articles article_stats:total_published_opinions name:publisher_name \
0                           1                         0   Harpreet Singh
1                           1                         0   Cormac McQuinn
2                           5                         0   Slashdot
3                           1                         0   Jon Fingas
4                           2                         0   THW

           twitter_stats:publisher_followers twitter_stats:tweets_mentioning_articles_sum
0                   9035                          10
1                   4670                          52
2                  301818                         488
3                   9236                         200
4                 1671036                         98
```

Tak przygotowane widoki można przekazać do warstwy prezentacyjnej i w ładny sposób zvisualizować, aby odbiorcy lepiej i łatwiej odebrali dane informacje. Poniżej pokazano przykładowy wykres wygenerowany z informacji z kilku kolumn z tabelki HBase'owej *topics*.



## 8 Testy funkcjonalne

W tym rozdziale opisane są przeprowadzone testy rozwiązania pozwalające stwierdzić poprawność jego działania.

### 8.1 Pobranie i przetworzenie danych, zapis do HDFS oraz Hive

Pokazano przykład przejścia danych od początku do końca z zaznaczeniem ważnych punktów przebiegu.

**Scenariusz:** Na początku w Hive brak jest jakichkolwiek danych w tabelach:

```

hive> SELECT * FROM articles;
OK
No rows selected (0.189 seconds)
hive> SELECT * FROM tweets;
OK
No rows selected (0.254 seconds)
hive> SELECT * FROM publishers;
OK
No rows selected (0.311 seconds)
hive>

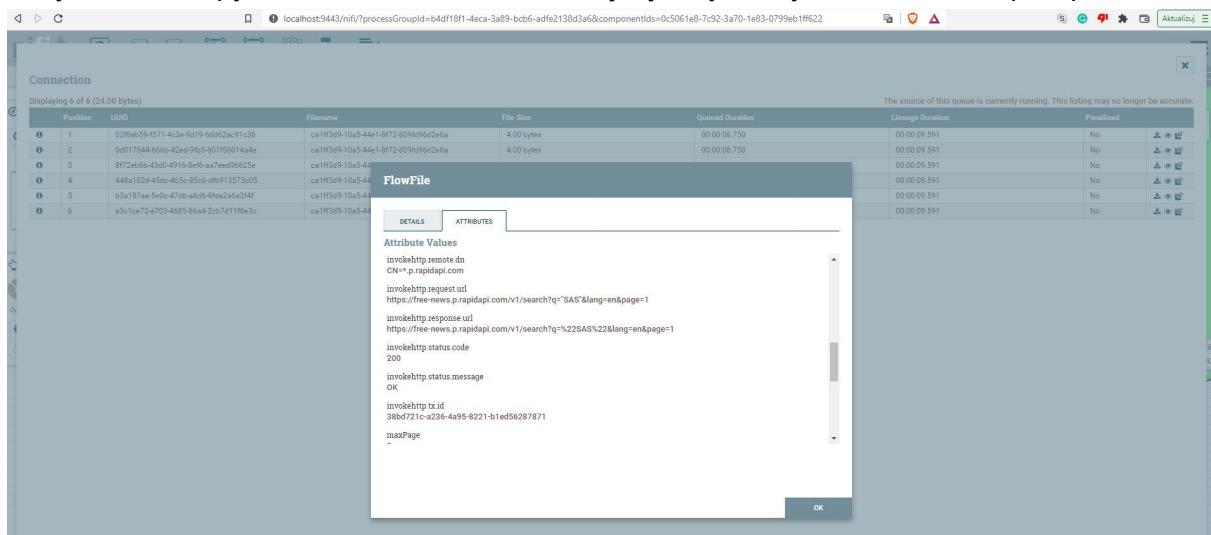
```

**Cel:** Sprawdzenie poprawności wczytania danych z API, przetworzenie ich oraz zapis do **HDFS** oraz **Hive**.

**Wykonanie:** Jednorazowo co 7 dni automatycznie wywoływany jest proces NiFi. Test wykonywano 23.01.2022 o godzinie 00:47 czasu polskiego (22.01.2022 o godzinie 23:47 czasu UTC na maszynie wirtualnej).

**Oczekiwany wynik:** Po automatycznej, cotygodniowej inicjalizacji procesu, bez ingerencji administratora, dane powinny samoczynnie zostać poprawnie załadowane do Hive.

Przykładowe zapytanie do Free News API znajduje się w atrybutie “invokehttp.request.url”:



Przykładowa odpowiedź API jest w formacie JSON:



```

hive> SELECT id FROM articles;
OK
+-----+
| id   |
+-----+
| 0    |
| 1    |
| 2    |
| 3    |
| 4    |
| 5    |
| 6    |
| 7    |
| 8    |
| 9    |
| 10   |
| 11   |
+-----+
11 rows selected (0.156 seconds)
hive>

```

Równolegle przebiega pobieranie danych z Twitter API. Przykładowa lista odpowiedzi API na wiele zapytań:

Response							
		Displaying 11 of 11 (103.74 KB)		The source of this queue is currently running. This listing may no longer be accurate.			
	Position	UUID	Filename	File Size	Queued Duration	Lineage Duration	Serialized
1	1	b111d54a-b428-4b6a-a654-a28991e51fe	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	451 00 bytes	00:00:53.847	00:04:54.954	No
2	2	a6165ae4-7d94-414a-559e-7a6e534e7369	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	435 00 bytes	00:00:41.821	00:04:54.954	No
3	3	3780016-e7f4-4b8a-7ab-e6903898e77cc	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	9.09 KB	00:00:23.843	00:04:54.954	No
4	4	1b053039-9914-41ad-975c-670fbfae4e3	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	8.48 KB	00:00:08.631	00:04:54.954	No
5	5	c0ee0d0f-0ef1-4510-99e-e6516d6b6a73	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	3.45 kB	00:01:06.596	00:04:54.954	No
6	6	79613122-c0d3-469c-8155-5a03e9797fa1	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	14.15 KB	00:01:01.235	00:04:54.954	No
7	7	fbd10f41-1616-d444-ef21-a0a3f726640	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	11.14 KB	00:00:29.835	00:04:54.954	No
8	8	b36f44d5-1644-417-ecc0-22d79943	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	31.18 KB	00:00:11.819	00:04:54.954	No
9	9	c0ea979c-1d21-a263-a50a-140ca0a1953d	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	16.52 KB	00:00:47.719	00:04:54.954	No
10	10	e1724246-5155-4214-a034-412ba12c72e	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	387 00 bytes	00:00:35.853	00:04:54.954	No
11	11	61791473-6d46-29b9571319ab7e9e9	ca1ff3d9-10a5-44e1-8772-80949692d2e6a	8.48 KB	00:00:17.858	00:04:54.954	No

Przykładowa odpowiedź Twitter API dla pojedynczego artykułu jest w postaci JSON:

localhost:3443/nifi-content-viewer/?mode=Formatted&ref=http%3A%2F%2Flocalhost%3A9443%2Fnifi-api%2Ffile-queues%2F94ec132c-9175-32...

View as:		formatted																																																												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
"statuses": [ {	"created_at": "Sun Jan 17 06:14:48 +0000 2012",	"id": 1483110588177319940,	"id_str": "1483110588177319940",	"text": "Le Parfum de Berlin Packaging: Welcome, Le Parfum to the Berlin Packaging family, our newest acquisition in France. This acquisition further reinforces our commitment to the European market.",	"truncated": false,	"entities": {	"symbols": [ ],	"hashtags": [ ],	"user_mentions": [ { "screen_name": "BerlinPackaging", "name": "Berlin Packaging", "id": 30936695, "id_str": "30936695", "indices": [ 3, 33 ] } ],	"urls": [ ] },	"metadata": {	"lang": "en",	"result_type": "recent"	"source": "a href=\"https://mobile.twitter.com/\" rel=\"nofollow\">Twitter Web App</a>",	"in_reply_to_status_id": null,	"in_reply_to_status_id_str": null,	"in_reply_to_user_id": null,	"in_reply_to_user_id_str": null,	"in_reply_to_screen_name": null,	"user": { "id": 1242545853, "id_str": "1242545853", "name": "MidwestTBC", "screen_name": "MidwestTBC", "location": "Indianapolis, IN", "description": "Bringing clarity to the food and beverage packaging industry via research and strategic consulting", "url": null, "entities": { "description": { "urls": [ { "url": "http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png", "expanded_url": "http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png", "display_url": "http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png" } ] } }, "protected": false, "followers_count": 329, "friends_count": 329, "listed_in_folders": false, "is_translator": false, "is_translation_enabled": false, "is_translation_reviewed": false, "profile_background_image_url": "http://abs.twimg.com/images/themes/theme1/bg.png", "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme1/bg.png", "profile_background_tile": false, "profile_image_url": "http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png", "profile_image_url_https": "https://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png", "profile_banner_url": null }	Filename: ca1ff2d9-10a5-44e1-8f72-809496d2e6a Content Type: application/json; charset=utf-8																																									





```
vagrant@node1:~$ hdfs dfs -ls /user/chojeckip/project/publishers_table
Found 1 items
-rw-r--r-- 1 root supergroup         3614 2022-01-24 00:06 /user/chojeckip/project/publishers_table/publishers_table_2022-01-24_00-06-06-176.avro
Vagrant@node1:~$
```

## Na koniec zapisywane w tabeli Hive:

```
hive> SELECT * FROM publishers;
OK
f05ec3b4d7a1d7c6cf332eee846d095c_4898501 f05ec3b4d7a1d7c6cf332eee846d095c_4898501 PRNewswire PR Newswire Global 120534 5932 28135
f95b7f268141193241ce340eecae3d66_15250661 f95b7f268141193241ce340eecae3d66_15250661 newscomauHQ news.com.au Australia 585087 3763 277940
bbfeef0e243c1e81e10eb346a3a9d54b_134758540 bbfeef0e243c1e81e10eb346a3a9d54b_134758540 timesofindia The Times Of India New Delhi 142048666 0 730533
8c8903a5deeef700e6f106db8811765b_19380823 8c8903a5deeef700e6f106db8811765b_19380823 Yahoo Yahoo Sunnyvale CA 1418266 8387 110708
6f7d660924781780c7cb67befbf888d_742143 6f7d660924781780c7cb67befbf888d_742143 BBCWorld BBC News (World) London, UK 34223566 127685 340816
1a664186ec0e91ca5392e62bbc66427d12_1168472395 1a664186ec0e91ca5392e62bbc66427d12_1168472395 EveningStandard Evening Standard London, United Kingdom 142055 1406 209907
624fc1f02515d5ce0a106c695d582a26_251808979 624fc1f02515d5ce0a106c695d582a26_251808979 MPIOpenAccess MPI Basel, Switzerland 21395 270 15254
dd0ede808602dc4743bd3fb7eb7e4_106532116 dd0ede808602dc4743bd3fb7eb7e4_106532116 commacmagine Cormac McQuina Dublin, Ireland 4669 89 3522
c4de57b5f7cd9a6233d617f0395b85d_81849273 c4de57b5f7cd9a6233d617f0395b85d_81849273 globenewswire GlobeNewswire Worldwide 1174 34 116
f7ddfc6d936a09419c768186a29c0f2_4898501 f7ddfc6d936a09419c768186a29c0f2_4898501 PRNewswire PR Newswire Global 120534 5932 28135
0cc4815bbad56d55ec7aa7c1de8d9aa_15250661 0cc4815bbad56d55ec7aa7c1de8d9aa_15250661 newscomauHQ news.com.au Australia 585087 3763 277941
3063377e201603b203a6f2a0006f5352_2768501 3063377e201603b203a6f2a0006f5352_2768501 abcnews ABC News Australia 19109661 13168 343867
0f44a51a5448108c5df30c6f565b6408_16675569 smh The Sydney Morning Herald Sydney, Australia 862629 6784 227023
f1f90cdd7aa7c9643869f9be26df0fab_291915848 f1f90cdd7aa7c9643869f9be26df0fab_291915848 Sam Elliott Sam Elliott London 7981 88 18911
cc5df411b16d44fe621ea0d3f726640_16887175 DailyMirror The Mirror UK 1302674 6664 883356
2fe761bf4e2b8a2d46128559d21ee97_138138925 2fe761bf4e2b8a2d46128559d21ee97_138138925 MirrorPolitics Mirror Politics UK 94682 1287 95118
50167b5f7cd9a6233d617f0395b85d_81849273 c4de57b5f7cd9a6233d617f0395b85d_81849273 globenewswire GlobeNewswire Worldwide 1174 34 116
f7ddfc6d936a09419c768186a29c0f2_4898501 f7ddfc6d936a09419c768186a29c0f2_4898501 PRNewswire PR Newswire Global 120534 5932 28135
23 rows selected (0.251 seconds)
hive>
```

Jak więc widać cały proces ładowania danych do Hive przebiega bez zarzutów. Załadowano więc całe dane do analizy. Liczba wierszy w tabelach Hive jest podana na następujących zdjęciach:

```
hive> SELECT COUNT(*) FROM articles;
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = vagrant_20220124080515_1208cab1-377d-42bd-bfaa0cc4543e3
Total MapReduce Jobs Launched: 1
  Job 0 Launched: Job 1 out of 1
Number of Reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducer.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Starting Job = job_1633823195514_0016, Tracking URL = http://node1:8088/proxy/application_1633823195514_0016/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1633823195514_0016
Hadoop job Information for Stage-1: number of mappers: 0; number of reducers: 0
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 00:59:43 Stage-1 map = 100%, reduce = 100%
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 00:59:43 Stage-1 map = 100%, reduce = 100%
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: INFO mapreduce.Job: MapReduce Jobs Launched:
22/01/24 00:59:43 [HiveServer2-Background-Pool: Thread-161]: INFO mapreduce.Job: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
0
1 row selected (28.413 seconds)
hive>
```

```
hive> SELECT COUNT(*) FROM tweets;
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = vagrant_20220124010209_af2ad46-74ac-4b9b-ac4-09b7c832949
Total MapReduce Jobs Launched: 1
  Job 0 Launched: Job 1 out of 1
Number of Reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducer.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Starting Job = job_1633823195514_0017, Tracking URL = http://node1:8088/proxy/application_1633823195514_0017/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1633823195514_0017
Hadoop job Information for Stage-1: number of mappers: 0; number of reducers: 0
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 01:02:07 Stage-1 map = 100%, reduce = 0%
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 01:02:07 Stage-1 map = 100%, reduce = 0%
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 01:02:07 Stage-1 map = 100%, reduce = 0%
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: INFO mapreduce.Job: MapReduce Jobs Launched:
22/01/24 01:02:07 [HiveServer2-Background-Pool: Thread-194]: INFO mapreduce.Job: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
0
1 row selected (63.422 seconds)
hive>
```

```

hive> SELECT COUNT(*) FROM publishers;
22/01/24 01:04:41 [HiveServer2-Background-Pool: Thread-225]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = vagrant_20220124010440_d47424e4-b6a7-4958-8d75-727a52fbaec
Total jobs = 1
Launching Job: 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set hive.exec.job.reducers.count=<number>
22/01/24 01:04:41 [HiveServer2-Background-Pool: Thread-225]: WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
22/01/24 01:04:41 [HiveServer2-Background-Pool: Thread-225]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Starting Job = job_1633823195514_0018, Tracking URL = http://node1:8088/proxy/application_1633823195514_0018/
Kill Command = /usr/local/hadoop/bin/hadoop job -kill job_1633823195514_0018
Hadoop job information for Stage-1: number ofappers: 0; number of reducers: 0
22/01/24 01:04:41 [HiveServer2-Background-Pool: Thread-225]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 01:04:41:55,026 Stage-1 map + 0%, reduce + 0%
22/01/24 01:04:41 [HiveServer2-Background-Pool: Thread-225]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 01:04:41:59,858 Stage-1 map + 100%, reduce + 0%
22/01/24 01:04:41 [HiveServer2-Background-Pool: Thread-225]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
2022-01-24 01:04:41:60,978 Stage-1 map + 100%, reduce + 100%
22/01/24 01:05:21 [HiveServer2-Background-Pool: Thread-225]: WARN mapreduce.Counters: Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.TaskCounter instead
Ended Job = job_1633823195514_0018
MapReduce Jobs completed: 1
22/01/24 01:05:21 [HiveServer2-Background-Pool: Thread-225]: WARN mapreduce.Counters: Group FileSystemCounters is deprecated. Use org.apache.hadoop.mapreduce.FileSystemCounter instead
Stage-Stage-1: HDFS Read: 0 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
0 rows selected (40.279 seconds)
hive>

```

## 8.2 Wykonanie analizy sparkowej i ładowanie do HBase'a

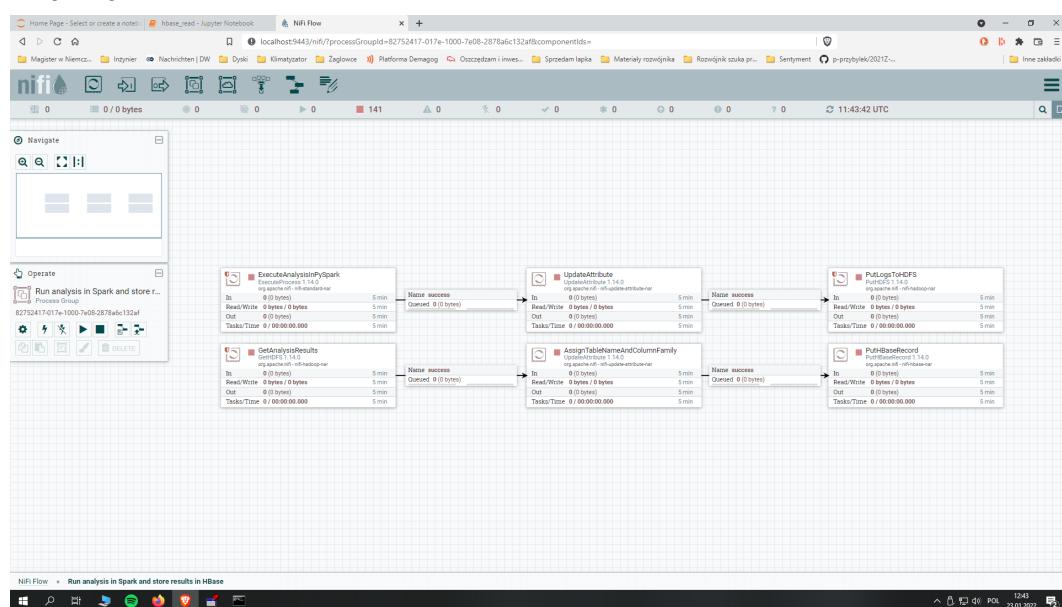
**Cel:** Sprawdzenie czy analiza sparkowa wykonuje się bez błędu po zainicjowaniu jej w NiFi'u i czy po jej wykonaniu następuje aktualizacja widoków w HBase.

**Wykonanie:** Zainicjowanie części analitycznej projektu w NiFi. Test wykonywano 23.01.2022 o godzinie 12:43 czasu polskiego (11:43 czasu UTC na maszynie wirtualnej).

**Oczekiwany wynik:** Po wykonaniu pojawi się nowy plik w folderze *logs* oraz timestampy wierszy w HBase się zaktualizują.

### 8.2.1 Stan "przed"

NiFi oczekuje na wydanie polecenia. HBase podejrzany z konsoli oraz z Pythonowego notebooka. Zwróćmy uwagę na timestampy w tabelach HBase'owych - są one z dnia 22.01.2022.



```

In [1]: 1 import happybase
2 import pandas as pd

In [16]: 1 connection = happybase.Connection(host='localhost', autoconnect=False, timeout=10)
2 table = connection.table('publishers')
3 scanner = table.scanner(include_timestamp=True)
4 scanner.set_batch_size(1000)
5 rows = []
6 for row_key, row_data in scanner:
7     processed_row_data = {}
8     for key, value in row_data.items():
9         processed_row_data[key.decode()] = value
10    if timestamp not in processed_row_data:
11        processed_row_data['timestamp'] = value[1]
12    rows.append(pd.DataFrame.from_dict(processed_row_data))
13 connection.close()

In [18]: 1 pd.concat(rows).reset_index(drop=True)

Out[18]:
   article_stats.articles_with_negative_sentiment_fraction timestamp article_stats.articles_with_p
0                   0.0 164288059980
1                   0.0 164288059980
2                   0.0 164288059980
3                   1.0 164288059980
4                   0.0 164288059980
5                   0.0 164288059980
6                   1.0 164288059980
7                   0.5 164288059940
8                   0.0 164288059940
9                   1.0 164288059980
10                  1.0 164288059980
11                  1.0 164288059980
12                  0.0 164288059980
13                  0.0 164288059980

```

```

In [19]: 1 connection = happybase.Connection(host='localhost', autoconnect=False, timeout=10)
2 table = connection.table('topics')
3 scanner = table.scanner(include_timestamp=True)
4 scanner.set_batch_size(1000)
5 rows = []
6 for row_key, row_data in scanner:
7     processed_row_data = {}
8     for key, value in row_data.items():
9         processed_row_data[key.decode()] = value[0].decode()
10    if timestamp not in processed_row_data:
11        processed_row_data['timestamp'] = value[1]
12    rows.append(pd.DataFrame.from_dict(processed_row_data))
13 connection.close()

In [20]: 1 pd.concat(rows).reset_index(drop=True)

Out[20]:
   article_stats.articles_with_negative_sentiment_fraction timestamp article_stats.articles_with_p
0                   0.0 164288059917
1                   0.343 164288059245
2                   0.359 164288059599
3                   0.2 164288059035
4                   0.274 164288059437
5                   0.303 164288059990
6                   0.5 164288059821
7                   0.398 164288059780
8                   0.0 164288059017
9                   0.368 164288059343
10                  0.427 164288059757
11                  0.255 164288059054
12                  0.317 164288059791
13                  0.41 164288059946
14                  0.167 164288059844
15                  0.423 164288059980

```

```

topic@node01 ~
Took 1.0005 seconds
base(main):>002>0> scan 'topics'
COLUMNCELL
beauty
column=article.stats:articles_with_negative_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=0.0
column=article.stats:articles_with_positive_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=1.0
beauty
column=article.stats:total_published_articles, timestamp=2022-01-22T16:14:18.889, value=1.0
beauty
column=name:topic, timestamp=2022-01-22T16:14:18.889, value=beauty
column=twitter.stats:tweets_mentioning_articles_sum, timestamp=2022-01-22T16:14:18.889, val
ue=1
business
column=article.stats:articles_with_negative_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=0.0
column=article.stats:articles_with_positive_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=1.0
business
column=article.stats:total_published_articles, timestamp=2022-01-22T16:14:18.889, value=1.0
business
column=name:topic, timestamp=2022-01-22T16:14:18.889, value=business
column=twitter.stats:tweets_mentioning_articles_sum, timestamp=2022-01-22T16:14:18.889, val
ue=525
economics
column=article.stats:articles_with_negative_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=0.0
column=article.stats:articles_with_positive_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=1.0
economics
column=article.stats:total_published_articles, timestamp=2022-01-22T16:14:18.889, value=1.0
economics
column=name:topic, timestamp=2022-01-22T16:14:19.746, value=economics
column=twitter.stats:tweets_mentioning_articles_sum, timestamp=2022-01-22T16:14:19.746, val
ue=23
energy
column=article.stats:articles_with_negative_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=0.0
column=article.stats:articles_with_positive_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=1.0
energy
column=article.stats:total_published_articles, timestamp=2022-01-22T16:14:18.889, value=1.0
energy
column=name:topic, timestamp=2022-01-22T16:14:19.37, value=energy
column=twitter.stats:tweets_mentioning_articles_sum, timestamp=2022-01-22T16:14:19.37, val
ue=40
finance
column=article.stats:articles_with_negative_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=0.0
column=article.stats:articles_with_positive_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=1.0
finance
column=article.stats:total_published_articles, timestamp=2022-01-22T16:14:18.889, value=1.0
finance
column=name:topic, timestamp=2022-01-22T16:14:18.905, value=finance
column=twitter.stats:tweets_mentioning_articles_sum, timestamp=2022-01-22T16:14:18.905, val
ue=37
food
column=article.stats:articles_with_negative_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=0.0
column=article.stats:articles_with_positive_sentiment_fraction, timestamp=2022-01-22T16:14:18.889, value=1.0
Food
Food
column=article.stats:total_published_articles, timestamp=2022-01-22T16:14:18.821, val
ue=2

```

## 8.2.2 Stan "po"

Pojedynczy Flowfile przeszedł przez część *analityczną*. Jego zawartość to napisy, które zostały skierowane na konsolę podczas wywołania Sparka. Zostały one zapisane do pliku .log, którego ostatnie 5 linijek podejrzano w konsoli.

W tabelkach HBase'owych zmienił się timestamp (wskazuje na 23.01.2022). Pozostałe dane się zmieniły, ponieważ analiza była odpalana na tych samych danych wejściowych.

Nifi Flow - Run analysis in Nifi and share results in Jupyter

```

(base) vagrant@node:~/PROJEKT$ hadoop fs -ls /user/chojeckip/project/logs
Found 2 items
-rw-r--r-- 1 root supergroup 17171630 2022-01-23 14:24 /user/chojeckip/project/logs/56010c49-f54c-4d6d-aa2c-1ec8d15421f5.log
-rw-r--r-- 1 root supergroup 8866193 2022-01-23 11:47 /user/chojeckip/project/logs/sc100049-05fc-4426-9c46-9c22091a143e.log
(base) vagrant@node:~/PROJEKT$ hadoop fs -cat /user/chojeckip/project/logs/sc* | tail
22/01/23 11:47:33 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
22/01/23 11:47:33 INFO BlockManager: BlockManager stopped
22/01/23 11:47:33 INFO BlockManagerMaster: BlockManagerMaster stopped
22/01/23 11:47:33 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
22/01/23 11:47:33 INFO SparkContext: Successfully stopped SparkContext
22/01/23 11:47:33 INFO ShutdownHookManager: Shutdown hook called
22/01/23 11:47:33 INFO ShutdownHookManager: Deleting directory /tmp/spark-73d52da1-e0f2-4e4f-8d12-6ccaa5e2d2de
22/01/23 11:47:33 INFO ShutdownHookManager: Deleting directory /tmp/spark-74cd4ab6-9e07-48d3-9110-323c56f979fa
22/01/23 11:47:33 INFO ShutdownHookManager: Deleting directory /tmp/spark-74cd4ab6-9e07-48d3-9110-323c56f979fa/pyspark-54fe4a4e-f46c-43a1-a7bd-bbb28971ee64f
(base) vagrant@node:~/PROJEKT$ 

```

jupyter hbase\_read [unresolved]

```

In [11]: connection = happybase.Connection(host='localhost', autoconnect=False, timeout=10)
          connection.open()
          scanner = table.scanner(include_timestamp = True)
          for row_key, row_data in scanner:
              processed_row_data = dict()
              for key, value in row_data.items():
                  processed_row_data[key.decode()] = value[1].decode()
              processed_row_data['timestamp'] = value[0]
              rows.append(pd.DataFrame([processed_row_data]))
          connection.close()

In [22]: pd.concat(rows).reset_index(drop=True)

Out[22]:
      article_statsarticles_with_negative_sentiment_fraction timestamp article_statsarticles_with_
0           0.0 16429347479
1           0.0 16429347479
2           0.0 16429347479
3           0.0 16429347479
4           0.0 16429347479
5           0.0 16429347479
6           0.0 16429347479
7           0.0 164293474691
8           0.0 164293474691
9           0.0 164293474691
10          0.0 164293474691
11          0.0 164293474691
12          0.0 164293474691
13          0.0 164293474691
14          0.0 164293474691

```

[4]:

```

jupyter hbase_read [unresolved]

In [23]: connection = happybase.Connection(host='localhost', autoconnect=False, timeout=10)
          connection.open()
          scanner = table.scanner(include_timestamp = True)
          for row_key, row_data in scanner:
              for key, value in row_data.items():
                  if key == 'timestamp':
                      continue
                  for key2, value2 in value.items():
                      if key2 == 'value':
                          value[0] = value2
          rows.append(pd.DataFrame([processed_row_data]))
          connection.close()

In [24]: pd.concat(rows).reset_index(drop=True)

Out[24]:
      article_statsarticles_with_negative_sentiment_fraction timestamp article_statsarticles_with_
0           0.0 164293489510
1           0.34 164293489555
2           0.36 164293489410
3           0.36 164293489511
4           0.274 164293489595
5           0.30 164293489439
6           0.5 164293489497
7           0.386 164293489578
8           0.36 164293489573
9           0.36 164293489573
10          0.36 164293489575
11          0.258 164293489198
12          0.217 164293489421
13          0.41 1642934894824
14          0.187 164293489444
15          0.423 164293489491

```

# 9 Podsumowanie

W tym rozdziale poddano pod refleksję działanie rozwiązania.

## 9.1 Konkluzje

Osiągnięto cel projektu, czyli zaprogramowano w pełni działający i skalowalny system będący gotowy przetwarzać duże ilości danych. Wszystkie zaplanowane warstwy zostały zaprojektowane oraz zaimplementowane poprawnie. Testy pokazały działanie systemu oraz poprawność odpowiedzi zarówno ze strony HDFS, Hive jak i HBase.

Dużym wsparciem dla stworzonego systemu okazał się Apache NiFi, dzięki któremu dane są pobierane i dostarczane do odpowiednich miejsc.

Niestety nie osiągnięto pełni mocy systemu, gdyż był on ograniczony możliwościami użytych API. Stąd przeprowadzone analizy wsadowe nie są tak zaawansowane i interesujące jakbyśmy chcieli, a jedynie pokazują wartość zaprogramowanego systemu oraz jego prawidłowe działanie.

## 9.2 Co się udało?

W trakcie projektu niektóre planowane działania okazały się bardziej trudne i stanowiły pewne wyzwanie dla autorów. Ostatecznie udało się je ukończyć osiągając sukces. Mianowicie:

1. Uzyskanie dostępu do całej odpowiedzi od Free News API, polegającej na sczytywaniu informacji o artykułach w pętli wywołującej odpowiednią stronę z podanej odpowiedzi.
2. Ręczna autoryzacja do Twitter API za pomocą kodu w Groovy.
3. Odnalezienie problemów związanych z nietypowymi znakami w odpowiedziach od Free News API oraz Twitter API i naprawienie bądź obejście tych problemów.
4. Stworzenie systemu, który byłby całkowicie ze sobą połączony i działał bez potrzebny ingerencji z zewnątrz, dzięki wykorzystaniu Apache NiFi nie tylko do pozyskiwania danych, ale również do zapisu i odczytu danych na łączach między istniejącymi warstwami a analizą wsadową.

## 9.3 Co się nie udało?

W projekcie niektóre przeciwności nie zostały pokonane. Znaleziono jednak obejście tych problemów. Były to:

1. Zainstalowanie TensorFlow w Python, na maszynach wirtualnych - zamiast tego wykorzystano bibliotekę do analizy sentymentu, która nie opiera się na tym pakiecie.
2. Analiza sentymentu na początkowej części treści artykułów - zamiast tego przeprowadzono analizę sentymentu na tytułach artykułów.

## 9.4 Możliwe kierunki rozwoju

W toku pracy wąskim gardłem okazało się Twitter API. W wersji darmowej bardzo szybko przekroczeno dopuszczalny limit i trzeba było znacznie ograniczyć ilość zapytań na minutę.

Przy zastosowaniach praktycznych korzystanie z płatnych wersji API twitterowego oraz z innego źródła API z artykułami o większej przepustowości jest niezbędne. Możliwym jest również wzbogacenie projektu o dodatkowe źródła danych, jak na przykład API New York Timesa czy Guardiana.

Podczas pracy nie uznano, że jest potrzeba wykorzystania partycjonowania tabel Hive. Jednak w zależności od specyfikacji używania systemu do analizy artykułów można dodać takie rozwiązanie w przyszłości w celu uzyskania szybszego dostępu do konkretnych informacji, np. do artykułów jedynie z danego tygodnia czy dnia.

Analiza w Sparku jest dość podstawowa. Takie były założenia projektu - skupiono się na składowaniu i zautomatyzowaniu przetwarzania danych wejściowych. Przeanalizowanie tekstu tweetów metodami NLP na pewno podniosłoby wartość biznesową całości rozwiązania.

Cenne również byłoby wprowadzenie automatycznej wizualizacji danych oraz generowania raportów, np. za pomocą narzędzia PowerBI. Dane - nawet przeprocesowane - w obecnej formie są trudno dostępne i kompletnie niezrozumiałe dla użytkownika bez zaawansowanych kompetencji technicznych.

Do pełni rozwiązania lambda brakuje w tym rozwiązaniu warstwy Speed Layer. Dodanie jej na pewno zwiększyłoby wartość biznesową projektu. Budując wyjście dla warstwy Speed Layer można by wykorzystać popularny i powszechnie stosowany system Apache Kafka, dla którego nie było miejsca w zastosowanej w tym projekcie rozwiązań.

## 9.5 Podział prac w zespole

Indywidualny wkład każdego z członków zespołu:

1. Przemysław Chojecki - pozyskanie, przetwarzanie i zapis danych z Free News API, testy poprawności działania NiFi oraz warstwy batchowej, odpowiednie części dokumentacji;
2. Paweł Morgen - pozyskanie danych z warstwy składowania do Sparka, analiza w Sparku, ustalenie formy tabel HBase, zapis widoków wsadowych do HBase, testy Sparka oraz HBase, odpowiednie części dokumentacji;
3. Paulina Przybyłek - pozyskanie, przetwarzanie i zapis danych z Twitter API, ustalenie formy tabel Hive, prezentacja projektu.

Dodatkowo autorzy wspólnie planowali ostateczną formę całego systemu oraz pomagali sobie podczas pojawiania się niektórych błędów czy problemów.