



***Wydział Elektrotechniki, Automatyki, Informatyki i***

***Inżynierii Biomedycznej.***

***Karol Gałka, Sebastian Bana*ś**

***Studio projektowe 1***

***WEAIiB, Inzynieria oprogramowania i systyemow***

Aplikacja webowa do ustalania odpowiednich miejsc

# Cel aplikacji

Aplikacja ma na celu udostępnienie użytkownikowi narzędzia rekomendujacego miejsca w Krakowie, które na podstawie informacji wytworzonych z postow Twittera uzytkownika dobiera miejsca które go zainteresuja.

## Analiza danych

Decyzja podejmowana jest poprzez przeprowadzenie analizy na zebranych danych. Dane te domyślnie dotycza zwyk;le codziennego życia i zainteresowań użytkownika. Na ich podstawie przeprowadzona zostaje analiza kategoryzująca poszczególne informacje do określonych uprzednio kategorii. Aby zapewnić innowacyjność rozwiązania informacje te powinny pochodzić z niekonwencjonalnych źródeł takich jak np. portale społecznościowe.

## Zwracany wynik

Wynikiem działania aplikacji powinny być miejsca znajdujące się na terenie miasta Kraków które według wyników aplikacji mogą wpasowywać się w profil osobowy użytkownika utworzony na podstawie analizy danych. Miejsca te powinny uprzednio zostać wprowadzone do bazy danych aplikacji w celu ich odpowiedniej konfiguracji pod model myślowy programu.

## Kategoryzacja danych

Zarówno profil użytkownika oraz informacje o miejscach powinny posiadać swój własny unikalny model. Modele te mają na celu ułatwienie kategoryzacji wyników analizy oraz połączenia ich z najbardziej trafnymi propozycjami z danych aplikacji. Powiązania powinny być na tyle silne, aby każdy wynik analizy użytkownika mógł zostać poprawnie przypisany do któregoś miejsca. Jednocześnie elastyczność i uniwersalność kategorii powinna być na tyle duża, aby każdy profil użytkownika mógł zostać poprawnie utworzony nawet z minimalnej ilości danych.

## Profile osobowe użytkownika

Profil powinien zostać utworzony na podstawie danych dostarczanych przez użytkownika. Decyzja o sposobie pobierania danych nie powinna wpływać na tworzenie modelów. Model powinien kategoryzować użytkownika względem pewnych cech osobowości, które można odpowiednio przyporządkować dowolnej osobie.

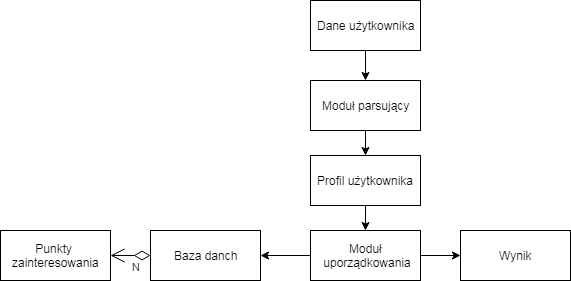
## Punkty zainteresowania

Zwracane jako wynik aplikacji punkty powinny być rozumiane jako miejsca bądź przejściowe lokacje oferujące aktywny sposób spędzania czasu, który spełnia wymagania profilu osobowego użytkownika. Punkty te podlegają kategoryzacji sprzężonej z kategoryzacją profili osobowych użytkowników.

## Diagram przejścia danych

## C:\Users\sg0304270\Downloads\Untitled Diagram (3).png

## Diagram wstępnej architektury



# Opis narzędzi

Aplikacja zostanie przygotowana jako aplikacja webowa udostepniająca klientowi procedury logowania i zarządzania swoim kontem. Od strony użytkownika zostanie przygotowany odpowiedni formularz pozwalający na poprawną konfiguracją konta użytkownika wraz z nadaniem pozwolenia aplikacji na przegląd danych osobowych.

## Język programowania i środowisko programistyczne

Projekt opierać będzie się głównie na języku Python w wersji 3.6. Jego korporacyjna struktura pozwoli na przygotowanie szkieletu rozległej aplikacji którą można będzie w przyszłości rozwijać lub rozszerzać. Za środowisko programistyczne posłuży produkt JetBrains – PyCharm.

## Framework

W celu uzyskania webowego charakteru aplikacji projekt korzystać będzie z Frameworka Flask.

Wykorzystane zostaną odpowiednie funkcjonalności dotyczące serwisow internetowych – zapytan http, parsowania htmli i inne. Oprócz tego Flask oferuje bogatą wersję ORM wraz z automatyczną konfiguracją bazy danych. W celu uproszczenia projektu powinny zostać wprowadzone starania uniknięcia konieczności zastosowania osobnego Front-endowego Frameworka.

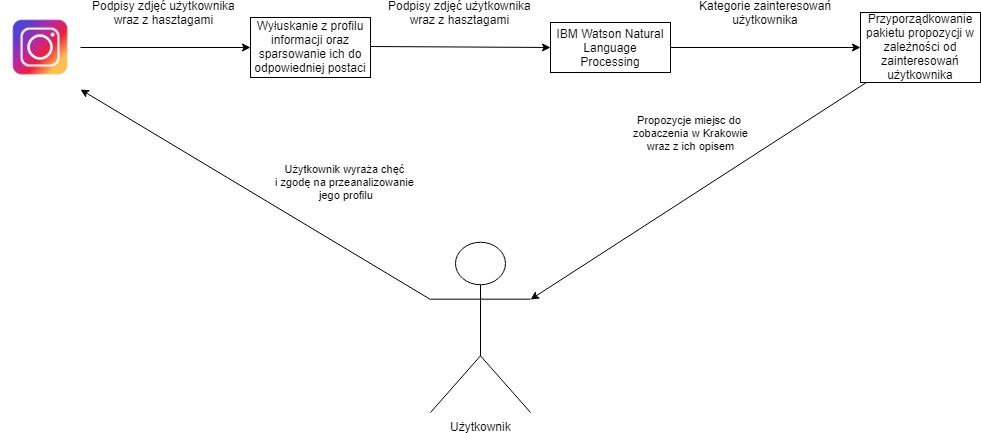
## Baza danych

Za bazę danych może posłużyć dowolna baza danych relacyjna lub nierelacyjna. Ze względu na latwosc konfiguracji, szybkosc dzialania zostala wybrana nierelacyjna baza danych – MongoDB, w której można przechowywac

* Dane o kontach użytkowników
* Profile osobowe użytkowników
* Punkty zainteresowań
* Historyczne dane wprowadzone przez użytkowników – tweety

## Watson

Watson to pakiet narzędzi oferowanych przez IBM, który zawiera między innymi kognitywne usługi serverless. Z pakietu wykorzystane zostanie narzędzie Natural Language Processing, które na podstawie przesłanego tekstu poda informacje o kategoriach zainteresowań użytkownika.



# Twitter

Twitter jest serwisem społecznościowym udostępniającym usługę mikroblogowania. Zarejestrowany użytkownik może wysyłać tweety, czyli krótkie wiadomości tekstowe (do 280 znaków), wyświetlane na profilu użytkownika oraz odczytywać tweety użytkowników go obserwujących. Wiadomości dotyczą spraw zwykle powiązanych z życiem zawodem lub zainteresowaniami użytkownika. Na ich podstawie człowiek może jest określić preferencje obserwowanego użytkownika dotyczących muzyki, architektury, kuchni oraz wielu innych. Zadaniem naszego algorytmu będzie określenie preferencji użytkownika na podstawie jego tweetów oraz wykorzystanie ich do polecenia mu miejsc, które potencjalnie mogą wzbudzić jego zainteresowanie.

## Twitter API

Twitter udostępnia rozbudowane API dla deweloperów, które umożliwia wyszukiwanie tweetów między innymi filtrując po tematyce, użytkowniku, popularności tweetu. Przykładowe zapytanie do API bez uwzględnienia nagłówków autoryzujących.

### <https://api.twitter.com/1.1/search/tweets.json?q=nasa&result_type=popular>&<auth\_data>

Zwraca ono najpopularniejsze tweety, zawierające frazę nasa. Wybrane możliwe parametry do zapytań Twitter API:

* q – wymagany, kodowany w UTF-8, zakodowane zapytanie URL o maksymalnej długości 500 znaków. Zapytania mogą być dodatkowo ograniczone złożonością.
* result\_type – opcjonalnu, Określa typ wyników wyszukiwania, które chcesz otrzymywać. Bieżące ustawienie domyślne to „mieszane”. Możliwe wartości to:
  + mieszane: uwzględnij w odpowiedzi zarówno wyniki popularne i nowe.
  + ostatnie: zwróć tylko najnowsze wyniki w odpowiedzi.
  + popularne: zwraca tylko najbardziej popularne wyniki w odpowiedzi.
* geocode – opcjonalny, zwraca tweety użytkowników znajdujących się w danym promieniu od punktu o podanej szerokości i długości geograficznej.

## Biblioteki do obsługi Twitter API

Powstało wiele bibliotek ułatwiających obsługę Twitter API, dla Javy są to:

Twitter4J - to nieoficjalna biblioteka Java dla API Twittera, dzięki które można łatwo zintegrować aplikację Java z Twitter API. Zalety Twitter4J:

* W pełni zgodna z Twitter API 1.1
* Działa na dowolnej platformie Java w wersji 5 lub nowszej
* Obsługuje platformę Android i silnik Google App Engine
* Zależność zerowa: nie są wymagane dodatkowe pliki jar
* Wbudowana obsługa OAuth oraz gzip

## MongoDB

MongoDB to nierelacyjna baza danych opierająca się na dokumentach przechowywanych w formacie JSON. Język zapytań opiera się na języku Javascript który w pełni wspiera operacja na obiektach typu JSON.

W celu uniwersalnego dostępu do bazy danych zastosowano bazę umieszczoną w środowisku Cloud. Dostęp do niej gwarantują klucz API oraz dane uwierzytelniające użytkownika.

Samo środowisko Cloudowe dostarcza bogatego GUI pozwalającego na przegląd i zarządzanie bazą na poziomie administratora. Wśród prezentowanych danych są funkcjonali użytkownicy bazy danych, metryki z dostępu i użycia oraz zarządzanie kilkoma serwerami bazodanowymi w ramach klastrów.

Aplikacja łączy się z bazą za pomocą oficjalnego API dla języka Python. Jest ono dostosowane od działania na obiektach połączeń z bazą danych w intuicyjny skryptowo-proceduralny sposób.

## Flask

Flask jest mikro Frameworkiem napisanym w języku Python w celu tworzenia aplikacji webowych. Nazwa mikro pochodzi od jego konfiguracji, która pozwala uruchomić serwer obsługujący żądania http w kilku linijkach. Brak konieczności konfiguracji rozległych serwisów sprawdza się przy fazie developmentu aplikacji zwłaszcza w jej wczesnych fazach, gdzie serwisy mogą zmieniać swoje wejścia i wyjścia w dynamiczny sposób.

Serwer zostaje utworzony poprzez linię komend przez którą należy podać domyślny plik konfiguracyjny serwera. Plik ten zawiera parametry takie jak mapowania żądań, politykę cookies oraz wiele innych parametrów mogących się znaleźć w nagłówkach żądań oraz dotyczące ograniczeń np. CORS.

Wirtualna informacja turystyczna

# Opis aplikacji

Aplikacja ma udostępnić użytkownikowi zautomatyzowaną metodę wyszukiwania interesujących miejsc na obszarze Krakowa które odpowiadają preferencjom użytkownika. Uzytkownik podaje nazwe jego konta na Twitterze, aplikacja pobiera historie jego tweetow, które mają dostarczyć informacji na temat docelowego przedziału miejsc, które użytkownik jest chetnie odwiedzilby. Następnie za pomocą danych zgromadzonych z profilu społecznościowego użytkownika dotyczących ogólnych zainteresowań, spośród zbioru miejsc zdefiniowanych w aplikacji zostanie wyodrębniony i zaprezentowany użytkownikowi jak najbardziej trafny wynik działania programu.

# Środowisko aplikacyjne

Aplikacja domyślnie powinna być dostępna jako aplikacja webowa umożliwiająca komunikację poprzez przeglądarkę internetową. Podczas tworzenia środowiska powinny zostać uwzględnione czynniki krytyczne dla poprawnego działania aplikacji

* Możliwość szybkiej komunikacji za pomocą przesylania tekstu przez użytkownika oraz sprawnej odpowiedzi ze strony serwera
* Wysoka zdolność obliczeniowa dla
  + Przetwarzania nazwy uzytkownika na historie jego tweetow
  + Przetwarzania tweetow uzytkownika na kategorie
  + Algorytm wyszukania odpowiednich miejsc
* Odpowiednia przepustowość łącza internetowego

# Środowisko programistyczne

Aplikacja powinna być wdrażana w środowisku pozwalającym na korporacyjną architekturę. Modułowość aplikacji pozwoli na jednoczesny rozwój każdego aspektu aplikacji równocześnie i bezkolizyjnie. Jednocześnie środowisko powinno udostępniać framework który odciąży programistów od zbędnego tworzenia obsługi protokołów komunikacyjnych. Wszystkie powyższe oczekiwania powinny zostać spełnione przy jednoczesnym zachowaniu minimalnej ilości użytych technologii.

# Moduły programu

## Pobieranie hisotrii tweetow uzytkownika

## Analizowanie tej historii

## Otrzymanie kategorii zainteresowan uzytkownika.

## Wybór najbardziej trafnych miejsc

## Architektura programu

W celu utworzenia rozbudowanej aplikacji potrzebne będzie środowisko o architekturze korporacyjnej, w którym moduły aplikacji mogłyby być rozwijane równolegle i bezkolizyjnie. Taka architektura gwarantowałaby dalszy rozwój aplikacji przez kolejne zespoły, ze względu na jej złożoność suboptymalnym pod względem jakości otrzymanego produktu jest, aby została ukończona i dopracowana przez jeden zespół programistyczny.

# Opis przebiegu implementacji modułów

## Pobranie od użytkownika historii jego tweetów

Jednym z powodów wybrania Twittera jest programowalny interfejs aplikacji (API), który oferuje. Autentyfikacja jest realizowana przez podanie w nagłówkach zapytania kierowanego do API odpowiednich kluczy, są to:

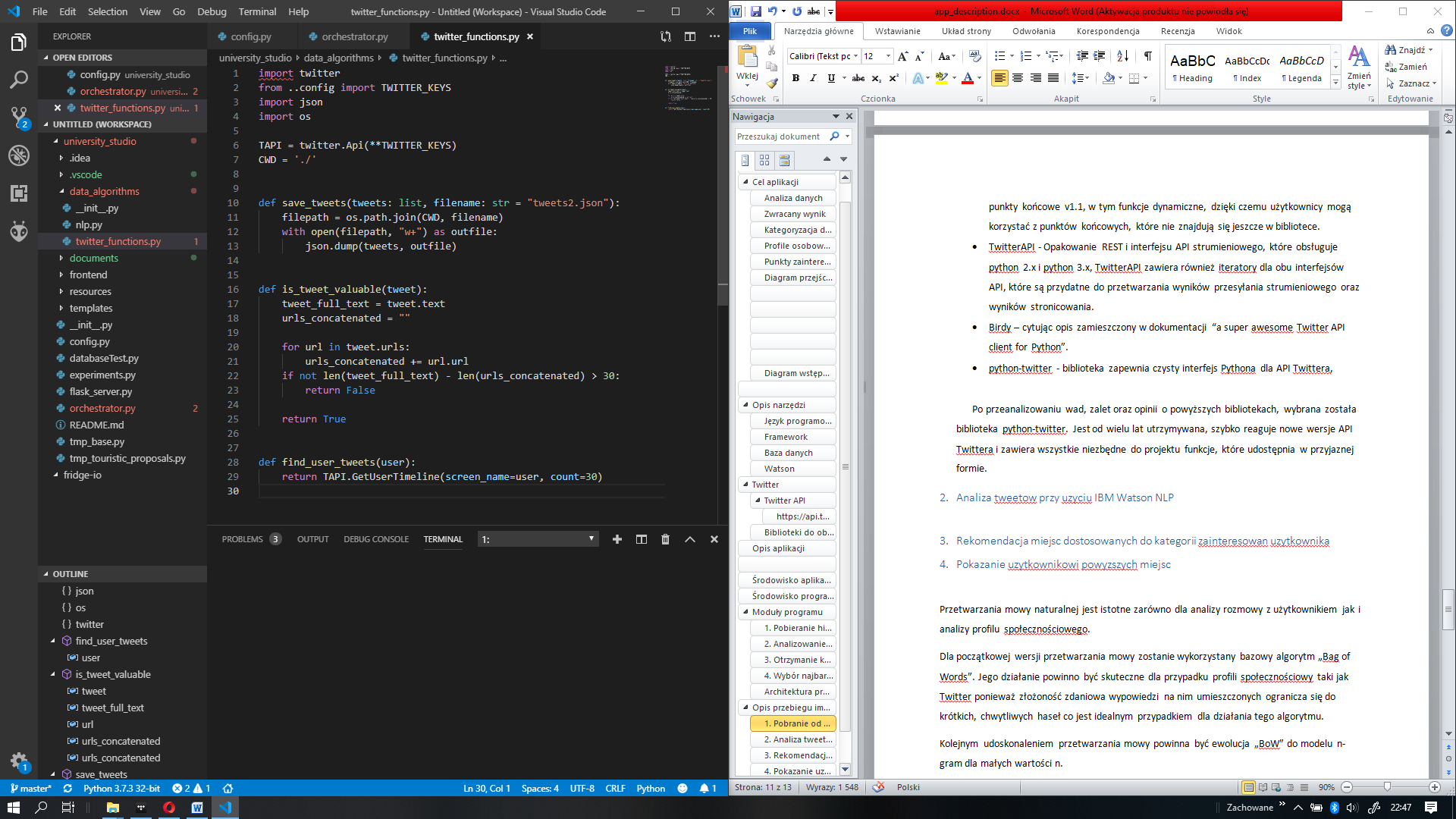
* Consumer key
* Consumer secret
* Access token key
* Access token secret

Powyższe klucze są możliwe do uzyskania przez stronę Twittera dla deweloperów. Po uzupełnieniu danych osobowych, kilku informacji o projekcie oraz opisu motywów jego stworzenia można wygenerować powyższe dane autentyfikacyjne. Istnieje wiele bibliotek napisanych na Pythona, które obsługują interfejs Twittera, wybrane z nich:

python-twitter maintained by @bear — this library provides a pure Python interface for the Twitter API (documentation)

* Tweepy - wrapper Pythona dla API Twittera.
* TweetPony - biblioteka Pythona mająca na celu prostotę i elastyczność.
* Python Twitter Tools - obszerna biblioteka Pythona do łączenia z Twitter REST i API do przesyłania strumieniowego Posiada również klienta Twittera z linii poleceń. Obsługuje Python 2.6, 2.7 i 3.3+.
* twitter-gobject - umożliwia dostęp do interfejsu API REST Twittera za pośrednictwem zestawu obiektów opartych na GObject w celu łatwej integracji z kodem opartym na GLib2.
* TwitterSearch - Interfejs oparty na Pythonie dla interfejsu API wyszukiwania 1.1.
* Twython - Aktywnie utrzymywany, czysty wrapper Pythona dla API Twittera. Obsługuje zarówno normalne, jak i strumieniowe API Twittera. Obsługuje wszystkie punkty końcowe v1.1, w tym funkcje dynamiczne, dzięki czemu użytkownicy mogą korzystać z punktów końcowych, które nie znajdują się jeszcze w bibliotece.
* TwitterAPI - Opakowanie REST i interfejsu API strumieniowego, które obsługuje python 2.x i python 3.x, TwitterAPI zawiera również iteratory dla obu interfejsów API, które są przydatne do przetwarzania wyników przesyłania strumieniowego oraz wyników stronicowania.
* Birdy – cytując opis zamieszczony w dokumentacji “a super awesome Twitter API client for Python”.
* python-twitter - biblioteka zapewnia czysty interfejs Pythona dla API Twittera,

Po przeanalizowaniu wad, zalet oraz opinii o powyższych bibliotekach, wybrana została biblioteka python-twitter. Jest od wielu lat utrzymywana, szybko reaguje nowe wersje API Twittera i zawiera wszystkie niezbędne do projektu funkcje, które udostępnia w przyjaznej formie.

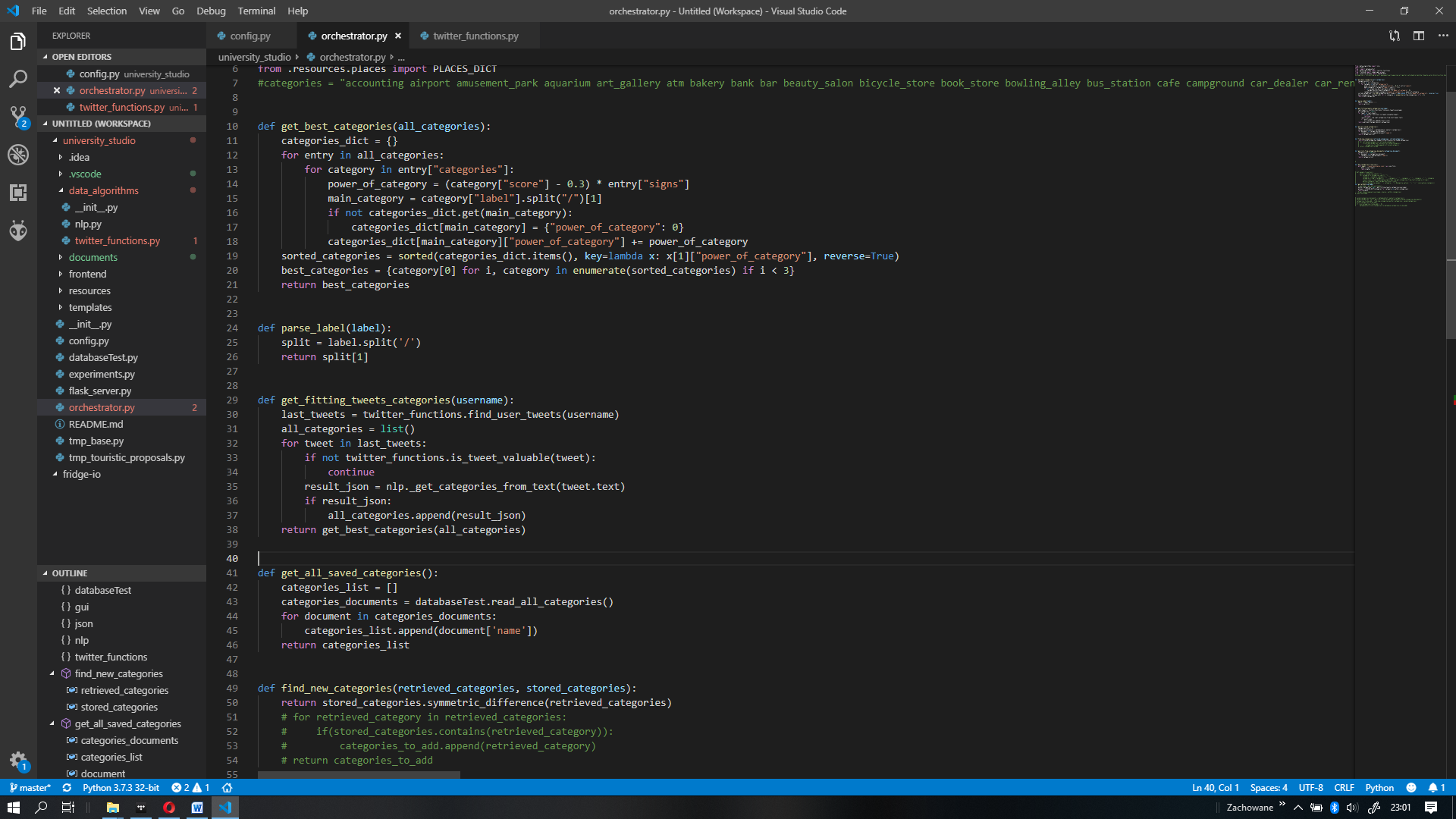


Skrypt odpowiedzialny za operacje na tweetach.

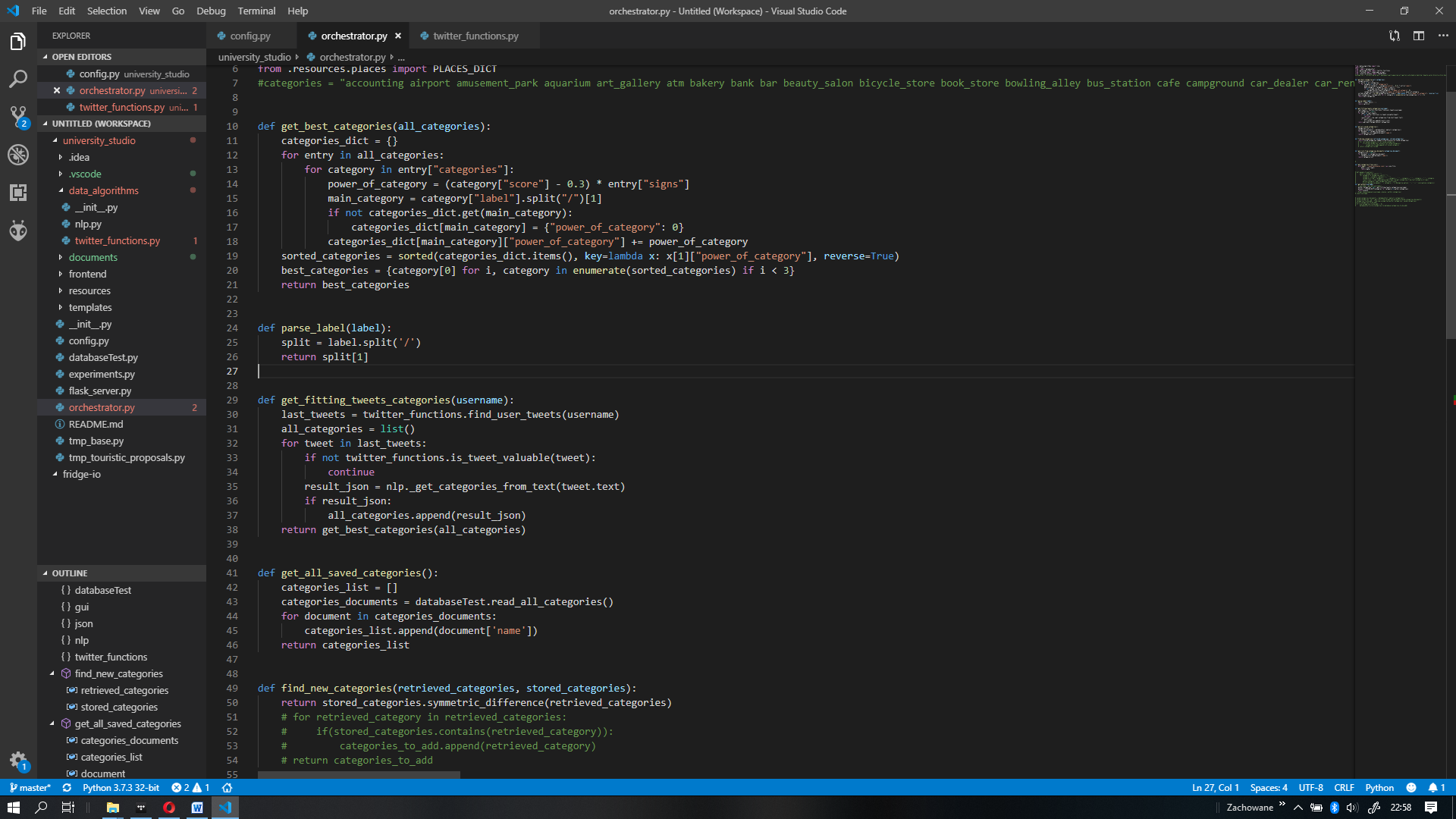
Aby otrzymać ostatnie 30 tweetów danego użytkownika wystarczy wysłanie jednego zapytania, które jest realizowane prze funkcje biblioteki python-twitter (importowanej po prostu jako twitter” – GetUserTimeline(). Powyższy moduł jest odpowiedzialny za operacje na tweetach i zawiera możliwość sprawdenia czy tweet niesie wartościową dla analizy informacje, a także funkcje zapisującą tweety lokalnie.

## Analiza tweetów przy użyciu IBM Watson NLP

Po pobraniu i wstępnym przefiltrowaniu tweetów użytkownika są one przesyłane do Watsona. Każdy tweet z osobna jest analizowany, następnie kategorie zainteresowań użytkownika są zwracane przez narzędzie IBMu. Po otrzymaniu wielu zbiorów potencjalnych kategorii zainteresowań użytkownika skrypt wybiera trzy najlepsze kategorie. Ten mechanizm jest oparty na współczynniku pewności zwracanym przez Watsona, a także ilości danych do niego wrzuconych.



Funkcja zwracająca kategorie zainteresowań użytkownika.



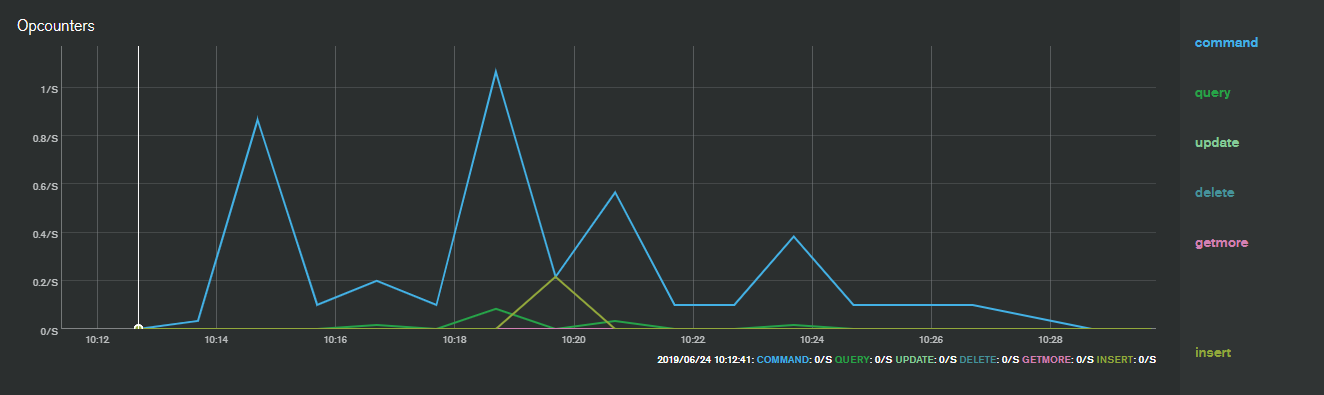
Funkcja zwracająca najlepsze kategorie ze zbioru

## 

Screenshot pokazujący schemat działania NLP Watsona.

## Rekomendacja miejsc dostosowanych do kategorii zainteresowań użytkownika

Użytkownik wysyła zapytanie o rekomendacje miejsc dla danego użytkownika aplikacji Twitter. Komunikacja odbywa się za pomocą REST korzystając z AJAX. Po odebraniu żądania aplikacja przegląda określoną ilość Tweetów użytkownika i korzystając z Watson language processing API oraz własnych algorytmów odnajduje najbardziej trafne kategorie dla danego użytkownika.



Panel administracyjny MongoDB, metryki operacji

Kolejnym krokiem jest wyszukanie wszystkich miejsc w bazie danych MongoDB którym zostały przypisane wyszukiwane kategorie. Dokument JSON zawierający nazwę miejsca oraz jego kategorię zostaje zwrócony do użytkownika w przypadku poprawnego wyszukiwania. Kod błędu 404 w przypadku nie znalezienia wyniku lub błędów w trakcie działania

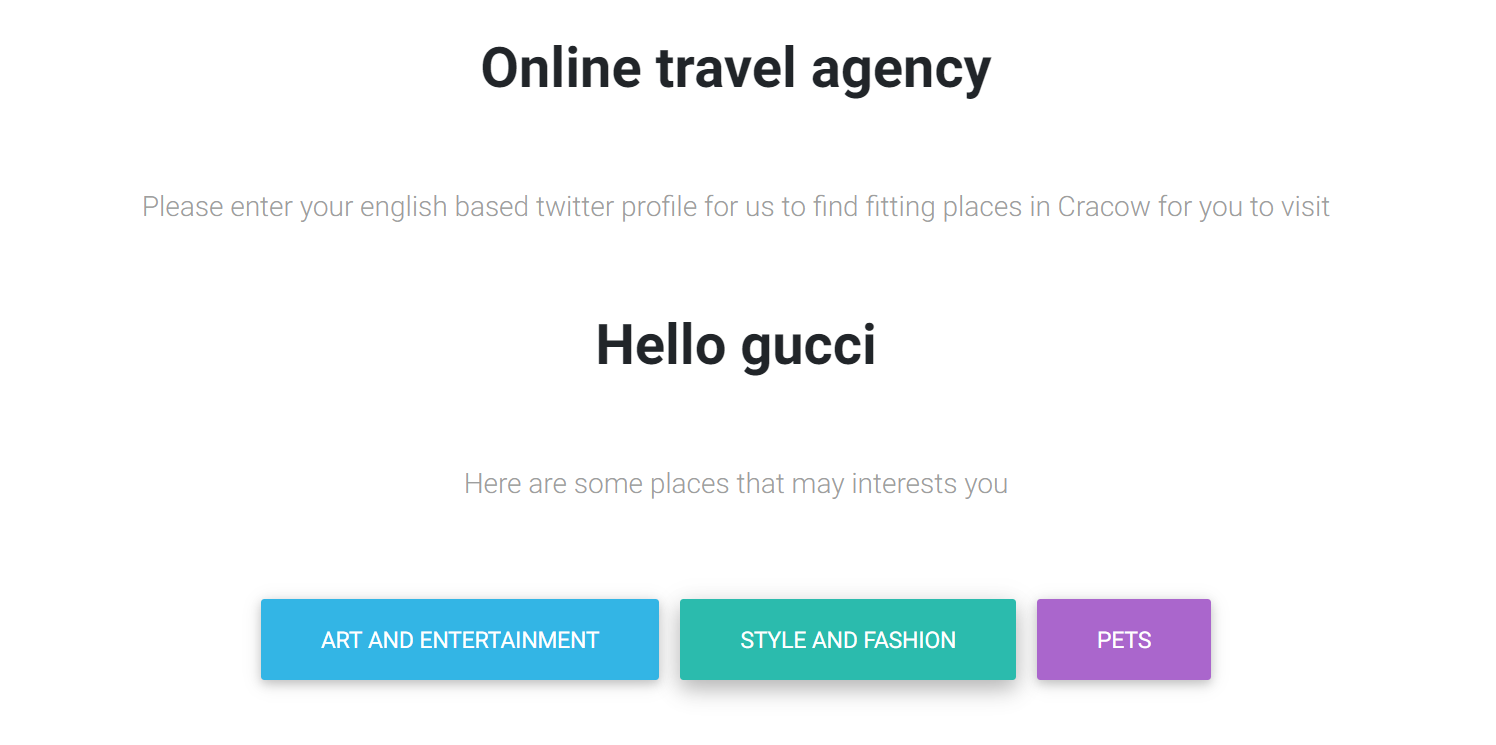
## Pokazanie użytkownikowi powyższych miejsc

Przeglądarka użytkownika odpowiada za odebranie asynchronicznych wyników rekomendacji.

Następnie zostaje uruchomiony skrypt JS który jest odpowiedzialny za dynamiczne wyświetlenie nowych elementów zawierających polecane miejsca. W skład tej procedury wchodzą tworzenie modali odpowiedzialnych za wyświetlenie map oraz konfiguracja startowych położeń znaczników na danych mapach.



Początek dynamicznego generowanie widoku w JS



Przykładowy wynik analizy tweetów

Arkusze stylów zostały dołączone do projektu poprzez bibliotekę Bootstrap która umożliwia użycie gotowych szablonów wszystkich domyślnych elementów HTML.