Politechnika Wrocławska

Wydział Informatyki i Zarządzania
kierunek studiów: Informatyka

Praca dyplomowa - inżynierska

Aplikacja wykorzystująca mechanizmy Business Intelligence do analizy danych z portali społecznościowych

Przemysław Dudycz

słowa kluczowe:
Portal społecznościowy, Architektura Lambda,
Business Intelligence, Big Data

krótkie streszczenie:
Niniejsza praca przedstawia wykonanie projektu i implementacje aplikacji analizującej dane z portali społecznościowych. Analiza skupia się na danych dotyczących konkretnych osób posiadających konta w jednym z trzech analizowanych portali społecznościowych.

opiekun pracy dyplomowej	.....	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	*ocena*	*podpis*

Ostateczna ocena za pracę dyplomową

Przewodniczący Komisji egzaminu dyplomowego	.....	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	*ocena*	*podpis*

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**

a) kategorii A (akta wieczyste)

b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

* *niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław 2017

Spis treści:

1.Wstęp

Rozwój Internetu jaki można zaobserwować w przeciągu ostatnich 20 lat, spowodował powstanie wiele systemów webowych, gromadzących ogromne ilości danych dotyczących różnych aspektów życia. Każda z aplikacji ma na celu pomoc w podstawowych potrzebach życiowych. W bardzo szybkim tempie, pojawiły się witryny oferujące usługi w dziedzinie handlu, bankowości czy sportu. Jednakże najpopularniejszą grupą aplikacji, z których korzystają użytkownicy, stały się portale społecznościowe. Dane, dostępne za pomocą tych portali, pozwalają na uzyskanie informacji dotyczących życia większości osób, używających sieć internetową. Każdy z użytkowników, posiadających konto w którymś z portali społecznościowych, udostępnia swoje zdjęcia, opinie oraz informacje dotyczące wydarzeń, w jakich bierze udział. Takie działanie, prowadzi do sytuacji, gdzie serwery tych witryn, gromadzą ogromną wiedzę na temat użytkowników w sieci. Jednym z celów tej pracy jest pokazanie, jakie informacje można uzyskać na podstawie analizy danych, które są zawarte na publicznych profilach stron społecznościowych.

Biorąc pod uwagę liczbę kont oraz ilość informacji dostępnych na serwerach, aktualnie najczęściej używanym portalem społecznościowym, jest aplikacja stworzona przez Marka Zuckerberga o nazwie Facebook. Użytkownicy tej aplikacji, tworzą bogaty profil swojej osoby, udostępniając zdjęcia oraz różne informacje na temat swojego życia, takie jak wykształcenie czy rzeczy, które użytkownik lubi. Dodając inne osoby do siatki swoich znajomych, profile zawierają informacje, na podstawie których, można zbudować pełen obraz danej osoby.

Drugą aplikacją internetową, pod względem popularności jest Instagram. Główną funkcjonalnością tego produktu, jest wgrywanie zdjęć, opisanych „hashtagami", pokazujących wydarzenie, które ma lub miało miejsce. Dziennie na serwery Instagrama, wgrywa się więcej niż 80 milionów zdjęć. Liczba ta pokazuje jak dokładnie użytkownicy dokumentują swoje życie poprzez rozbudowę profilu instagramowego. Podobnie jak w Facebooku, istnieje tu możliwość połączenia profilów ze znajomymi. Poprzez obserwacje danego użytkownika, dostaje się aktualną informacje o nowo wgranych zdjęciach.

Ostatnim narzędziem internetowym, który zostanie poddany analizie w aplikacji pisanej w ramach tej pracy, jest Twitter. Aplikacja ta, skupia się na publikowaniu krótkich treści, opisujących różne wydarzenia ze świata. W Twitterze, użytkownicy opisują temat swoich publikacji, podobnie jak w Instagramie, za pomocą hashtagów. Taki opis tematu, pozwala w prosty sposób pogrupować Tweety i ułatwić użytkownikom wyszukiwanie opinii na temat konkretnego wydarzenia. By otrzymywać informacje o nowo udostępnionych Tweetach konkretnej osoby, użytkownik może zacząć obserwować daną osobę, w celu otrzymywania aktualnych informacji na jej temat, na stronie swojego konta. Aktualnie Twittera używa około 650 miliona internautów. W Polsce aplikacja ta nie jest używana na tak dużą skalę, jak za granicą, jednakże treści jakie są publikowane pozwalają na wygenerowanie raportów, które obrazują podejście oraz zdanie ludzi na różne tematy związane z życiem.

Wzrastająca ilość danych w sieci, stała się przyczyną wielu problemów, związanych z przetrzymywaniem oraz analizą tych danych. Liczba transakcji bazodanowych oraz czas w jakim te transakcje są przeprowadzane są nieakceptowalne z perspektywy klienta, używającego danego systemu. W tym celu, zaczęto rozwijać rozwiązania z grupy Big Data oraz Business Intelligence, starające się zdefiniować struktury, które z powodzeniem przetworzą ogromną ilość danych w akceptowalnym czasie. Szybko powstały mechanizmy takie jak Hadoop czy Casandra oraz zdefiniowane zostały różne sposoby zbudowania hurtowni danych, których zadaniem jest przedstawienie raportów na podstawie dostępnej dużej ilości, zróżnicowanych danych. Jednakże, stworzenie takich raportów jest czasochłonne i dlatego jako między innymi odpowiedź na

wspomniane problemy, pojawiła się struktura Business Intelligence nazwana Architekturą Lambda. Twórcą tej architektury jest Nathan Marz. Celem mechanizmów Business Intelligence, zawartych w tej architekturze, jest przyspieszenie dostępu do aktualnych danych oraz raportów, stworzonych na podstawie aktualnych danych.

W ramach niniejszej pracy, zostanie zbudowana aplikacja analizująca dane z trzech, wyżej wymienionych portali społecznościowych – Facebook, Twitter oraz Instagram, za pomocą mechanizmów architektury wywodzącej się z dziedziny Business Intelligence – Architektury Lambda. Posłuży ona do generowania raportów na temat użytkowników tych portali społecznościowych.

2.Cel i zakres pracy

Celem pracy jest projekt i implementacja aplikacji webowej, analizującej dane, pochodzące z trzech portali społecznościowych – Facebooka, Instagrama oraz Twittera, za pomocą architektury, wywodzącej się z dziedziny Business Intelligence, zwanej Architekturą Lambda. Na cele cząstkowe pracy, składają się z następujących faz:

- Przeszukanie i przeanalizowanie rozwiązań konkurencyjnych
- Wybór technologii
- Stworzenie kompletnego projektu aplikacji
- Implementacja i stworzenie systemu

Powyższe cele cząstkowe zostaną wykonane w następującym zakresie:

- Jako rozwiązania konkurencyjne, opisane zostaną najpopularniejsze aplikacje analizujące dane z portali społecznościowych
- Technologiami, pomiędzy którymi nastąpi wybór, będą to technologie webowe oraz odpowiednio dobrane narzędzia bazodanowe
- Projekt aplikacji będzie przygotowany w zakresie zawierającym opis rzeczywistości i wizję systemu, historyjki użytkowników, diagram przypadków użycia, scenariusze przypadków użycia, diagram klas, diagram encji oraz prototyp interfejsu
- Przygotowanie działającej aplikacji, na podstawie wcześniej opisanego projektu

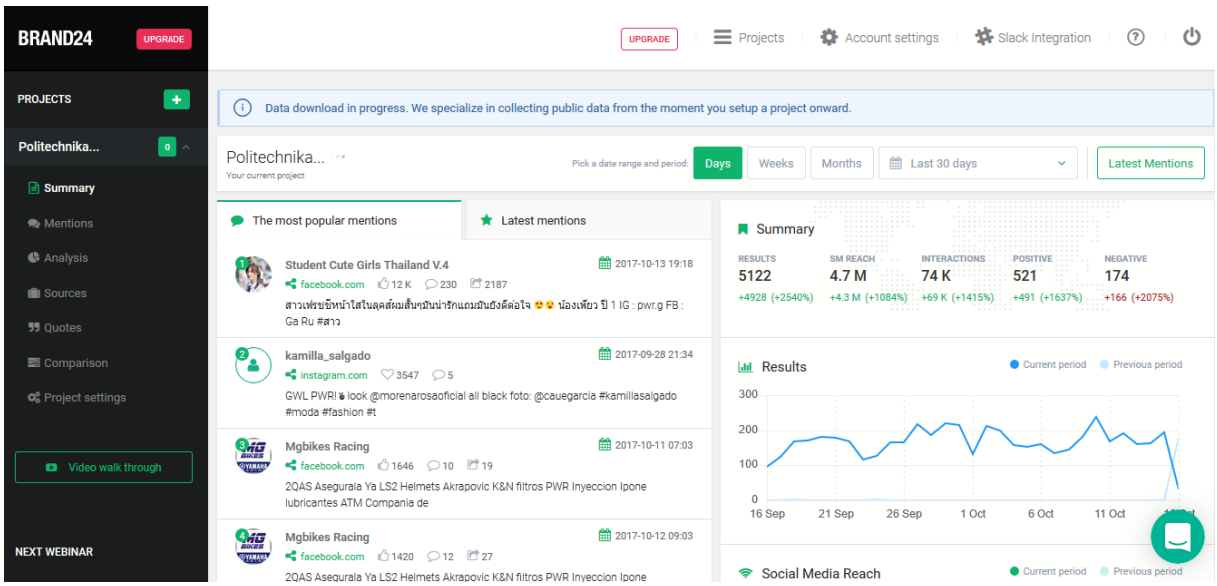
3 Przegląd rozwiązań konkurencyjnych

Niniejszy rozdział ma na celu pokazanie publicznie dostępnych aplikacji istniejących w sieci, które analizują dane z portali społecznościowych oraz przedstawienie różnic jakie występują pomiędzy prezentowanymi w niniejszej pracy rozwiązaniami, a rozwiązaniami ogólnodostępnymi. Przeglądając internet w poszukiwaniu tych konkurencyjnych rozwiązań, można dojść do wniosku, że informacje dostarczone przez portale społecznościowe są głównie wykorzystywane do tworzenia narzędzi analitycznych dla firm oraz korporacji zbierających dane dotyczących klientów. Udostępniają one dane dotyczące popularności danego produktu, wydźwięku opinii konsumentów czy raporty bazujące na lokalizacji. Na rynku istnieją też rozwiązania starające się odpowiadać na różnego rodzaju pytania, wykorzystując analizę tych danych, jednak analiza ta nie jest skierowana na konkretne osoby tak jak ma to miejsce w prezentowanym rozwiązaniu. Z pięciu narzędzi, opisanych poniżej, jedynie jedno stara się dostarczać funkcjonalności podobne do tych, które dostarcza projektowana w ramach niniejszej pracy aplikacja.

3.1 Brand24

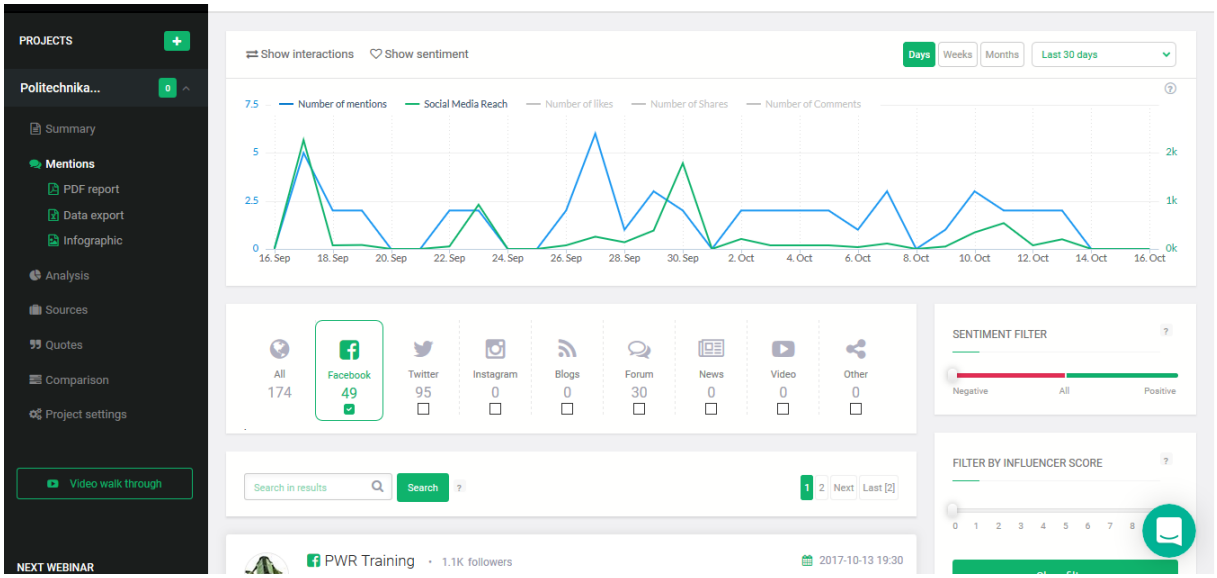
Pierwszym rozwiązaniem, jest polski produkt stworzony przez absolwenta Politechniki Wrocławskiej, Michała Sadowskiego, Brand24. Jest to jedno z największych i najpopularniejszych narzędzi do analizy i monitorowania informacji płynących z sieci, głównie z portali społecznościowych. Skupia się ono na dostarczaniu wszelkich danych dla firm. Dostarcza możliwość oglądania w czasie rzeczywistym tweetów czy postów dotyczących danej organizacji, wątku biznesowego czy konkretnego produktu. W automatyczny sposób rozpoznaje ono czy te informacje są pozytywne czy negatywne. Kolejną znaczącą funkcjonalnością prezentowanego rozwiązania jest ukazanie wpływu opinii konkretnego użytkownika na popularność produktu. Wyżej wymienione analizy są podstawą do tworzenia raportów opisujących zdania klientów na temat danego produktu lub biznesu działającego na rynku. Wykorzystywane jest tu także śledzenie trendów, które można zauważyć na rynku związanych z konkretnym biznesem. Analiza danych użytkowników posiadających konta w sieci, polega jedynie na raportach konsumenckich, które nie dostarczają szczegółowych informacji na temat konkretnego użytkownika.

Pierwszym krokiem by użyć Brand24 jest stworzenie projektu za pomocą słów kluczowych, na podstawie których będzie przeszukiwany internet. Po utworzeniu takiego projektu pokaże się strona z podstawowymi informacjami (Rys.3.1).



Rys.3.1. Strona podsumowania projektu Brand24

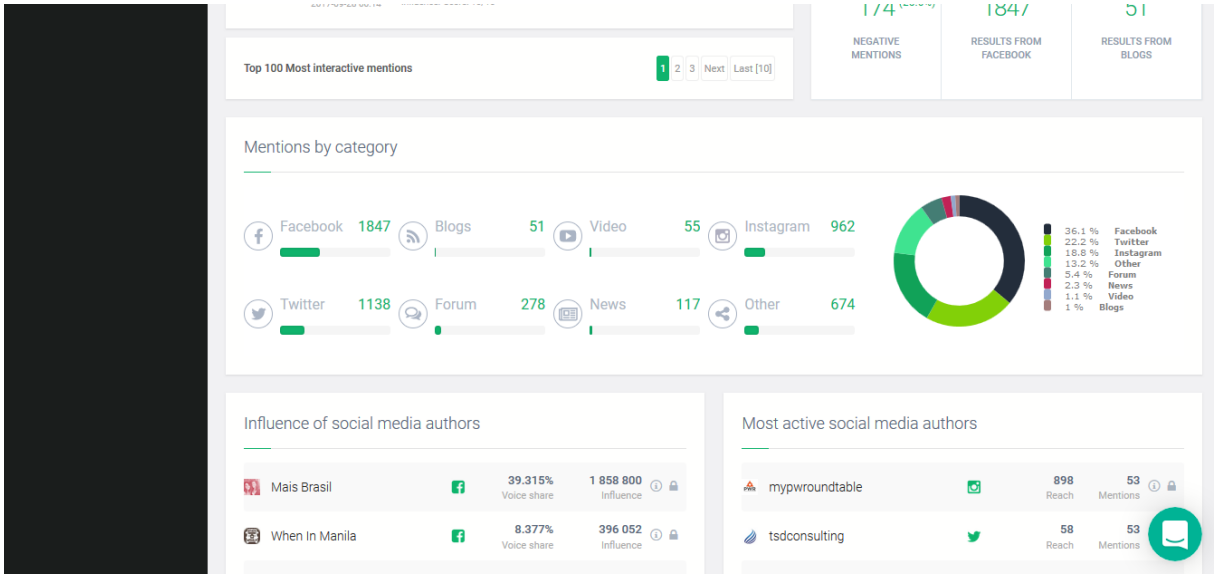
Jak można zauważyć na Rys.3.1, na pierwszej stronie pokazane są ogólne dane dotyczące wspomnień na interesujący użytkownika temat. Drugą częścią są najbardziej popularne wpisy w różnych portalach. Za pomocą menu, można przejść do bardziej szczegółowych raportów wygenerowanych przez aplikację.



Rys.3.2. Zakładka „Mentions" projektu Brand24

Rys.3.2 przestawia zakładkę „Mentions", w ramach której możliwy jest dostęp do informacji pokazujących szczegóły wystąpienia wpisów. Widok tego raportu jest podzielony na trzy części. Pierwszą jest graficzne przedstawienie danych dla użytkownika w formie wykresu, który pozwala na bardzo skuteczną analizę. Drugą częścią jest wypisanie aktualnie udostępnionych wpisów użytkowników na zaznaczony temat. Ostatnią funkcjonalnością na tej stronie jest możliwość określenia filtrów, które zawężają prezentowane informacje na wykresie oraz aktualne udostępnienia wpisów użytkowników w sieci. Filtry jakie można zdefiniować to źródło z jakiego dane pochodzą, określenie czy opinia użytkownika jest pozytywna czy negatywna oraz wpływ danego wpisu na społeczność.

Informacje, które są zawarte w tej zakładce mogą z sukcesem zobrazować opinie jaką konsumenci mają na dany temat oraz określić na której witrynie nasz produkt cieszy się największą popularnością i dla przykładu tam skierować akcje promocyjne.



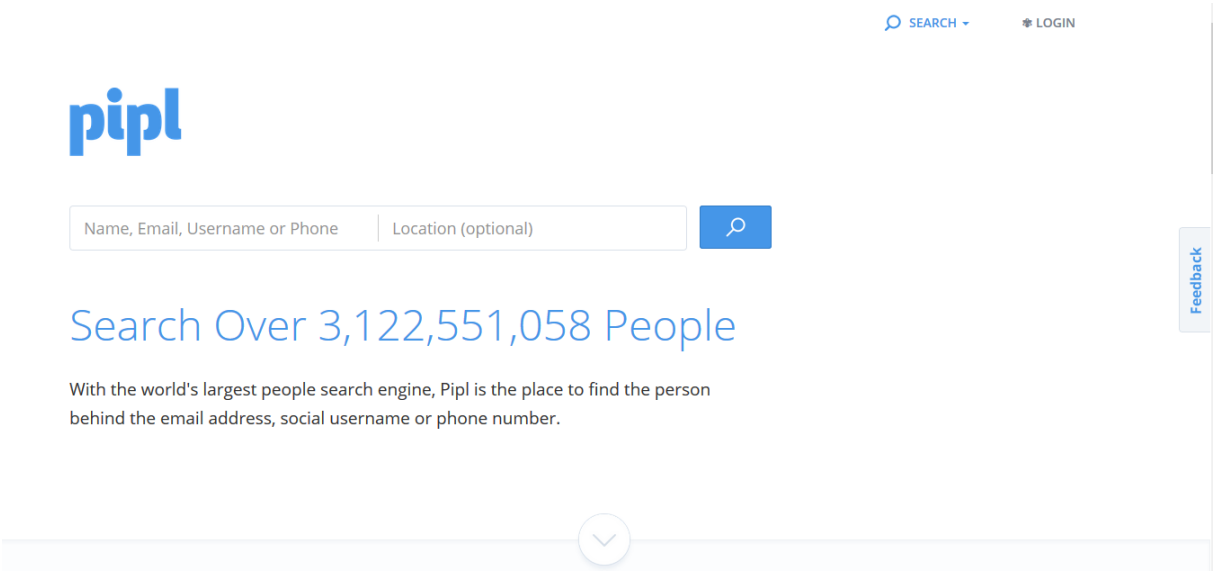
Rys3.3. Zakładka „Analysis" projektu Brand24

Ostatnią omawianą zakładką, znajdującą się w aplikacji Brand24, jest przedstawiona na Rys.3.3 zakładka Analysis. Pokazuje ona analizy i raporty dotyczące różnych witryn, na których został wspomniany analizowany produkt. Dostępne tutaj są też filtry, za pomocą których z łatwością można znaleźć najistotniejsze informacje.

Używając aplikacji Brand24 można wyrobić sobie opinie o klientach firmy oraz dotrzeć do ich opinii o produkcie. Analizy i raporty które znajdują się na tej witrynie, mogą pomóc skierować akcje promocyjne w odpowiednie miejsce oraz dają możliwość uzyskania natychmiastowej informacji co w produkcie należy poprawić, a co jest jego dobrą jego stroną. Patrząc jednak z perspektywy aplikacji tworzonej w ramach tej pracy, analiza informacji z mediów społecznościowych skupia się ogólnie na słowach kluczowych, użytych przez użytkowników. Brand24 nie dostarcza analizy danych na temat konkretnego użytkownika, co jest tematem tej pracy.

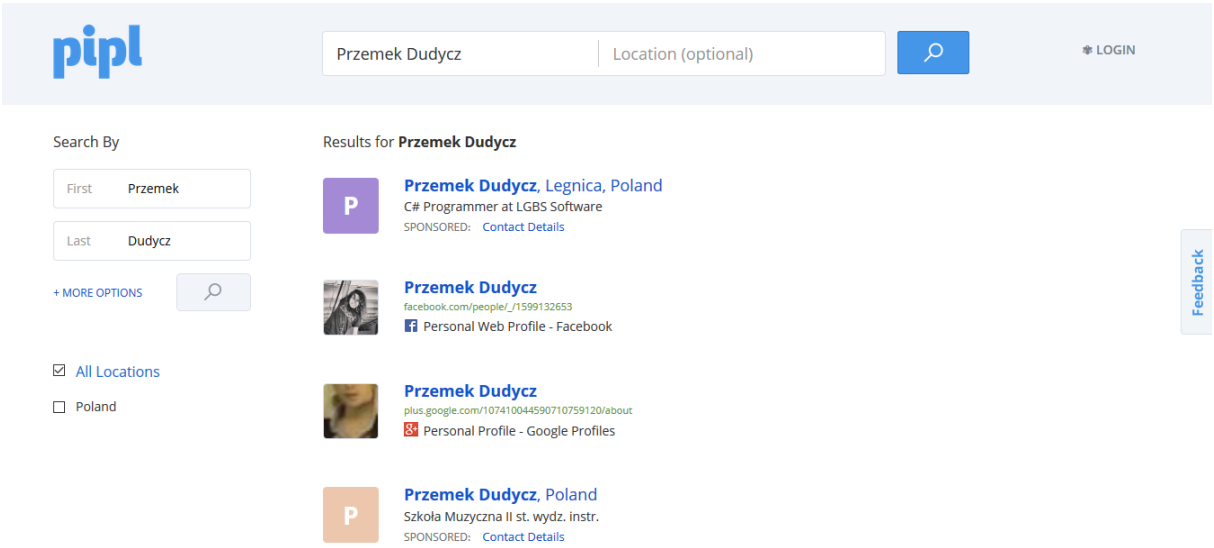
3.2 Pipl

Pipl jest aplikacją z grupy „people search engine". Jest ona w stanie wyszukać konta oraz podać podstawowe informacje dotyczące danej osoby. Po wpisaniu, przykładowo, imienia i nazwiska, dostarcza ona listę kont z różnych serwisów internetowych. Po wybraniu jednego z nich ukazuje się link do danego konta oraz podstawowe informacje zaczerpnięte z wybranego serwisu internetowego. Rozwiązanie to jest jedynie wstępem do tego co dostarcza prezentowana w niniejszej pracy aplikacja. Nie generuje ono szczegółowych raportów, a jedynie odnośniki do kont osoby, która jest wyszukiwana.



Rys.3.4. Strona startowa Pipl

Cała interakcja z aplikacją zaczyna się od strony startowej (Rys.3.4.) W tym miejscu należy wpisać imię i nazwisko poszukiwanej osoby. Następnie, po pewnym czasie, pokazuje się lista kont z różnych serwisów, które mogą należeć do tego użytkownika.



Rys. 3.5. Strona po wyszukiwaniu danego użytkownika

Na Rys.3.5. przedstawiono stronę po wyszukiwaniu danej osoby. Jak widać, pokazane są tam jedynie odnośniki do konkretnych witryn. Jeżeli w internecie są osoby posiadające konta o tym samym imieniu i nazwisku, także będą pokazane w wyniku wyszukiwania.

3.3 Brandwatch

Kolejnym produktem dostępnym na rynku, służącym do analizy portali społecznościowych jest Brandwatch. Wykorzystywany jest on do monitorowania dyskusji oraz opinii wystawianych w internecie na temat danego produktu, wydarzenia czy firmy. Za pomocą hasztagów, słów kluczowych, filtruje on dane by następnie przedstawić raporty konsumenckie. Analizując działanie tej aplikacji, można zauważyć, że jest ona mniej skomplikowana oraz posiada mniejszą liczbę raportów niż aplikacja Brand24. Analiza przeprowadzona za pomocą tego narzędzia, skupia się jedynie na dostarczeniu ogólnych informacji dotyczących konkretnych osób. Narzędzie to jest wykorzystywane jedynie w perspektywie analizy biznesowej. By

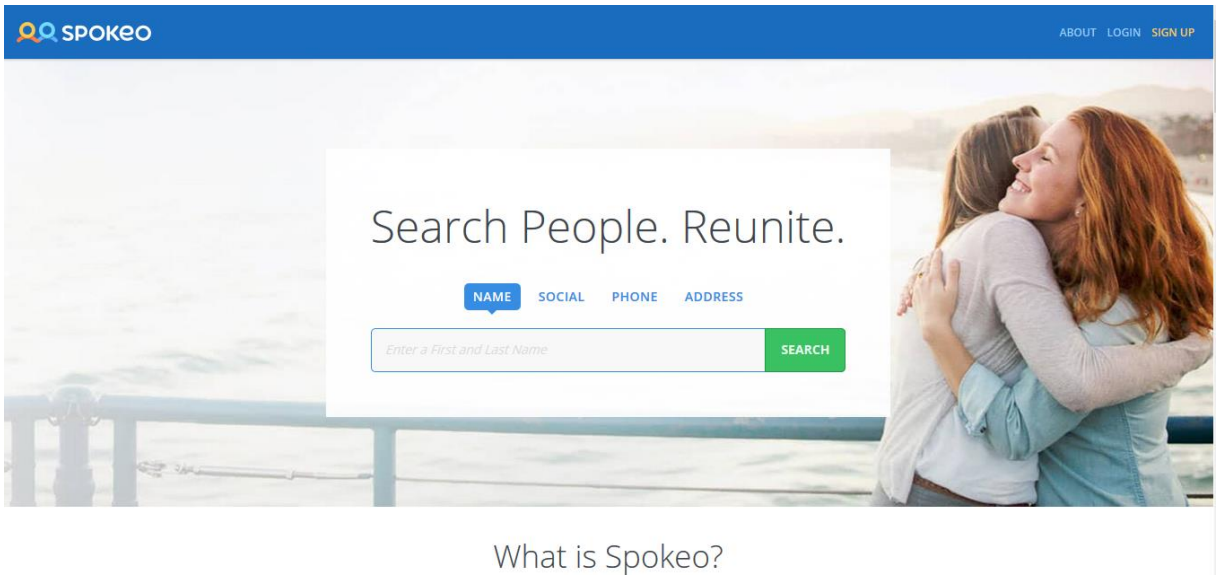
korzystać z Brandwatcha'a, należy podać dane takie jak firmowy email czy nazwa firmy, dlatego opis tej witryny jest przeprowadzony na podstawie opisu dostępnego na stronie Brandwatch'a.

3.4 Crowdbooster

Główną ideą Crowdboostera jest zmaksymalizowanie popularności tweetów czy postów publikowanych przez użytkownika. Wykonywane analizy mają na celu ukazanie informacji pomagających w uzyskaniu jak największej widowni oraz w wytworzeniu jak największej liczby fanów w sieci. Dostarczone raporty ukazują czas, w którym posty osiągają najwięcej polubień czy treści, które powodują największe poruszenie. Możliwe jest monitorowanie profilu użytkownika w celu analizy jego poczynań w sieci. Podobnie jak w Brandwatch'u, nie ma tu informacji szczegółowych na temat konkretnej osoby. Wszystkie zebrane dane są jedynie wykorzystywanie w celu analizy ogólnych trendów.

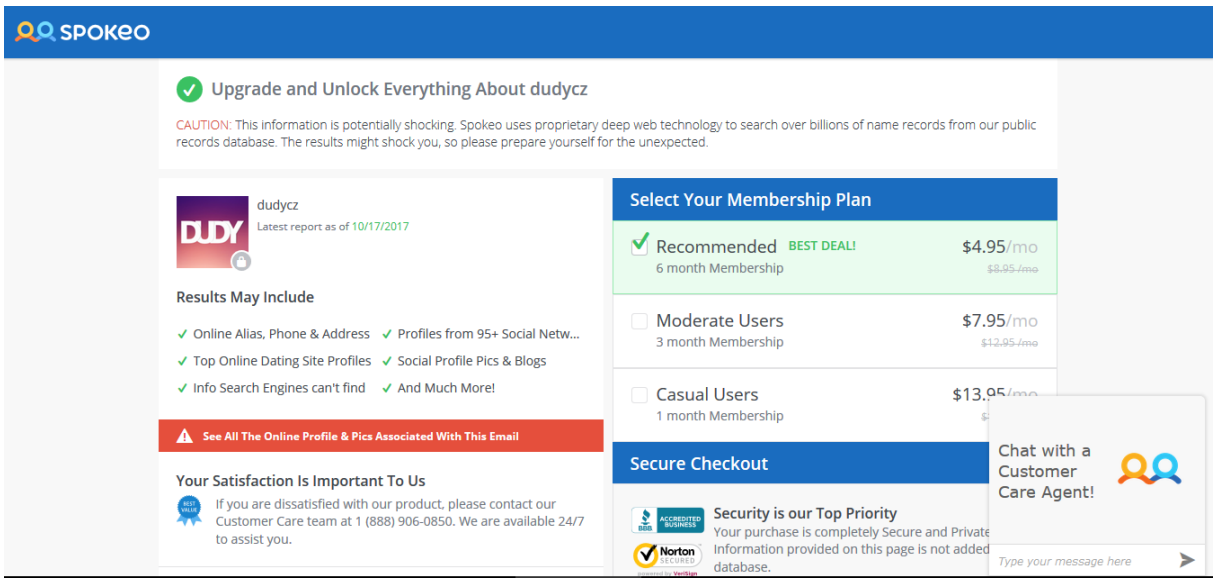
3.5 Spokeo

Ostatnią przedstawianą aplikacją z zakresu niniejszej pracy, jest Spokeo. W teorii dostarcza ona funkcjonalności podobne do proponowanych w aplikacji będącej tematem niniejszej. Z informacji zaczerpniętych na stronie, można wywnioskować, że za pomocą tego narzędzia możliwe jest uzyskanie bardzo szczegółowych informacji na temat użytkowników portali społecznościowych. Raporty mogą przedstawiać dane personalne, historie lokalizacji, w których użytkownik przebywał czy nawet wysokość wynagrodzenia w pracy. Dane mają być zaczerpnięte z różnych baz danych płatnych oraz bezpłatnych.



Rys. 3.6. Strona startowa Spokeo

Na Rys.3.6. można zobaczyć stronę startową, na której należy wpisać imię, nazwisko czy np. numer telefonu. Po rozpoczęciu wyszukiwania użytkownik jest przenoszony na stronę, gdzie pokazuje się informacja czy dana osoba została znaleziona.



Rys.3.7. Strona odblokowująca dane z portalu Spokeo

Jednakże po próbie odblokowania danych, użytkownik jest zmuszony zapłacić za udostępnienie wyszukanych danych (Rys.3.7).

3.6. Podsumowanie

Podsumowując przedstawienie najpopularniejszych produktów dostępnych na rynku, które wykorzystują analizę danych z portali społecznościowych, można dojść do wniosku, że jest bardzo mało narzędzi konkurencyjnych dla realizowanego w ramach niniejszej pracy produktu. Większość rozwiązań analizując dane z portali społecznościowych takich jak Facebook, Tweeter czy Instagram, służy do monitorowania opinii klientów na temat produktu oraz skutecznego reagowania na występujące trendy na rynku. W większości przypadków dostarczone raporty przedstawiają ogólne informację na temat konkretnego użytkownika, ukazując je w perspektywie danego trendu biznesowego. Żadne z tych narzędzi nie skupia się na analizie konkretnego użytkownika w celu jego poznania. Jedynie aplikacja, Spokeo, może być konkurencyjna dla rozwiązania z niniejszej pracy, jednakże wymaga zapłaty za dostarczane usługi.

4. Przegląd technologii

Wybór technologii, jakie będą użyte w procesie wytwarzania projektowanej aplikacji jest bardzo ważnym etapem, wpływającym w znacznym stopniu na kształt tworzonego rozwiązania. Bez podstawowej wiedzy o istniejących narzędziach, których można użyć, oraz możliwościach tych narzędzi, bardzo trudno jest o zaprojektowanie i stworzenie dobrze działającej aplikacji. Wpływ wybieranych technologii, można zaobserwować w architekturze, jakości kodu oraz co najważniejsze, efektywności działania aplikacji. Patrząc z perspektywy użytkownika docelowego, efektywność działania jest najważniejszym czynnikiem wpływającym na zadowolenie z korzystania z dostarczonego produktu. W aplikacjach Business Intelligence, w których występuje przetwarzanie ogromnej liczby danych, efektywność aplikacji jest bardzo trudna do osiągnięcia dlatego wybór odpowiedniego narzędzia, pozwalającego przetworzyć te struktury danych jest bardzo istotna.
Mówiąc o wyborze technologii dla aplikacji przedstawionej w ramach niniejszej pracy, trzeba spojrzeć na problem z perspektywy trzech warstw:

1. Warstwy prezentacyjnej (klienckiej)
2. Warstwy serwerowej
3. Warstwy bazodanowej

Warstwa prezentacyjna dotyczy wszystkiego, co zobaczy użytkownik docelowy. Wybór w tej warstwie jest wyborem z bardzo szerokiego spectrum rozwiązań dostępnych na rynku. Różnice na jakie można napotkać, są to głównie różnice w architekturze tworzonego kodu, dlatego wskazując na któreś z rozwiązań, powinno kierować się wygodą w trakcie pracy z daną architekturą. Drugą warstwą jest warstwa serwerowa, która odpowiada za część logiki biznesowej, tworzenie zapytań do zewnętrznych serwisów oraz kontakt z bazą danych. Wybór w tej części aplikacji powinien być kierowany łatwością stworzenia połączenia z wybranym serwerem bazodanowym oraz efektywnością transferu danych z warstwą kliencką. Ostatnią warstwą, którą należy wyróżnić przy wyborze technologii, jest warstwa bazodanowa. Z perspektywy wcześniej wspomnianej efektywności aplikacji, w tym przypadku jest to wybór najbardziej znaczący, wpływający na prędkość działania dostarczanych funkcjonalności. Jednakże wskazanie technologii w części bazodanowej nie ogranicza się tylko do wskazania odpowiedniego serwera bazodanowego, ale także wyboru architektury, która będzie zaimplementowana za pomocą tego serwera, pozwalającej na przetwarzanie w akceptowalnym czasie dużych zbiorów danych.
W poniższym rozdziale będą opisane technologie, pomiędzy którymi następował wybór, w każdej z trzech, wyżej wymienionych warstw.

4.1. Warstwa prezentacyjna (kliencka)

Pierwszą wyszczególnioną warstwą jest warstwa prezentacyjna. Jest ona odpowiedzialna za etap działania aplikacji bezpośrednio związany z użytkownikiem docelowym – intuicyjny interfejs oraz wyświetlanie danych w sposób przejrzysty. Całość musi być ułożona tak, żeby dany użytkownik nie był zagubiony korzystając z aplikacji. Zadowolenie użytkownika końcowego bezpośrednio jest związany z warstwą kliencką, dlatego należy przywiązać dużą uwagę w jaki sposób prezentujemy nasze funkcjonalności. Jako że aplikacja wytwarzana w ramach tej pracy, jest to aplikacja webowa, podstawą, która będzie wykorzystywana to HTML oraz CSS z rozszerzeniem SASS. Wykorzystany zostanie także framework o nazwie Bootstrap w celu łatwiejszego stylowania strony za pomocą gotowych komponentów dostarczonych przez ten framework. Wybór, jaki dokonujemy w tej warstwie, jest to wybór pomiędzy mechanizmami zapewniającymi wyświetlanie danych pochodzących z serwera oraz obsługującymi interakcję użytkownika ze stroną. W ramach tego rozdziału wybór ten będzie ograniczony do narzędzi związanych z językiem Javascript.

Javascript jest to język skryptowy, stworzony przez Brendana Eicha z firmy Netscape, z myślą o przeglądarkach internetowych, pozwalający na implementację zachowania strony w związku z interakcją użytkownika. W ostatnich latach, język ten zyskał na popularności, wynikiem czego jest przeniesienie mechanizmów związanych z tym językiem na stronę serwera (NodeJS). Pomimo rozszerzenia swojego wykorzystania na stronę serwerową, język ten ciągle jest kojarzony z kodem klienckim [1]. Pierwszym pytaniem, jakie pojawia się przy wyborze technologii związanej z tą warstwą, jest pytanie o słuszność wyboru jakiegokolwiek frameworku javascriptowego. Biorąc pod uwagę efektywność działania strony, podstawowe badania wykazują, że sam javascript jest dużo szybszy w działaniu, niż jego odpowiedniki [2], jednakże jeżeli weźmiemy pod uwagę współczesne komputery jakimi posługują się użytkownicy, dla większości aplikacji, różnica ta będzie niezauważalna. Jeżeli weźmiemy pod uwagę jakość tworzonego kodu, prędkość wytwarzania aplikacji oraz łatwość z jaką dane są transferowane z serwera na stronę html, użycie frameworka wypada dużo lepiej niż wykorzystanie czystego Javascriptu.

Ilość dostępnych frameworków javascriptowych jest bardzo duża. Porównanie tych narzędzi należy dokonywać w perspektywie architektury, efektywności, łatwości wytwarzania produktu oraz aktywności społeczności powiązanej z danym frameworkiem. Biorąc pod uwagę popularność frameworków w 2017 roku, do porównania wybrane zostały trzy javascriptowe frameworki – Angular, ReactJs oraz EmberJs [3].

Angular jest to jest to open-sourcowy framework, stworzony przez firmę Google. Opiera się on na rozszerzeniu Javascriptu – TypeScriptcie. Dzięki temu jesteśmy w stanie tworzyć kod w pełni obiekowo zoorientowany, używając mechanizmu klas, dziedziczenia czy interfejsów. Angular jest narzędziem multiplatformowym, pozwalającym na tworzenie warstwy klienckiej za pomocą reużywalnych komponentów, tworząc w ten sposób kod czytelny oraz łatwy do utrzymania [4].

ReactJs jest to biblioteka, stworzona przez społeczność facebookową, opierająca się na notacji JSX. Pozwala ona na renderowanie podkomponentów w kodzie HTML, co pozwala na tworzenie reużywalnego kodu. Patrząc z perspektywy architektury warstwy klienckiej, ReactJs nie posiada mechanizmów do tworzenia odpowiedniej struktury. Zazwyczaj dodawane jest jeszcze jedno narzędzie – najczęściej Flux – pozwalające na stworzenie takowej architektury, jednakże połączenie tych dwóch mechanizmów komplikuje naukę tworzenia aplikacji za pomocą Reacta [5][6].

EmberJs jest to open-sourcowy framework, stworzony przez Yehuda Katza**.** Opiera się on na architekturze MVC i pozwala na budowanie aplikacji typu „single page”. Posiadane mechanizmy dwubiegunowego wiązania danych oraz podejścia bazującego na tym, że URL przetrzymuje stan aplikacji pozwala na tworzenie skalowalnych systemów[7][8].
W tabeli Tab. 1.1 zastawione są różnice w znaczących mechanizmach dla trzech, wyżej wymienionych frameworków.

Tab 1. 1 Wady i zalety frameworków javascriptowych (Źródło: [9])

	Angular	**ReactJs**	**EmberJs**
Zalety	• Efektywność działania • Renderowanie po stronie serwera • Posiada wszystkie mechanizmy pozwalające na zbudowanie aplikacji • Wszystkie aktualizacje językowe są od razu dostępne • Stworzony kod jest łatwy do testowania • Łatwość tworzenia reużywalnego kodu	• Efektywność działania • Renderowanie po stronie serwera • Prostota • Podejście biblioteki – wybiera się mechanizmy potrzebne deweloperowi • Wszystkie aktualizacje językowe są od razu dostępne	• Efektywność działania • Renderowanie po stronie serwera • Posiada wszystkie mechanizmy pozwalające na zbudowanie aplikacji • Dobra dokumentacja • Łatwość tworzenia reużywalnego kodu
Wady	• Słaba dokumentacja • Twórcy Angulara nie dbają o kompatybilność nowych wersji frameworka względem poprzednich. Trudności w uaktualnieniu frameworka	• Brak mechanizmów tworzenia architektury aplikacji – użycie przykładowo Fluxa mocno utrudnia naukę wytwarzania aplikacji • Zła renoma notacji JSX	• Mniejsza społeczność internetowa • Skomplikowany w użyciu przy nietypowych przypadkach

Porównując wszystkie trzy frameworki, można dojść się do wniosku, że ciężko jest wybrać najlepsze narzędzie, zapewniające rozwiązanie na problemy, które można spotkać w trakcie implementacji. Każdy z tych trzech frameworków posiada swoje podejście i często wybór danego rozwiązania, będzie kierowany wygodą w użyciu danego frameworka. Biorąc pod uwagę wymienione wady i zalety (Tab. 1.1), w aplikacji tworzonej w ramach niniejszej pracy, do stworzenia warstwy klienckiej zostanie wybrany framework Angular.

4.2. Warstwa serwerowa

Drugą rozpatrywaną warstwą aplikacji tworzonej w ramach niniejszej pracy jest warstwa serwerowa. W tym miejscu należy dobrze zastanowić się nad wyborem technologii z jaką będzie się pracowało ponieważ duża część logiki biznesowej oraz transfer danych z bazy danych do klienta odbywa się właśnie na serwerze. Operacje te mają znaczący wpływ na efektywność aplikacji, dlatego narzędzia jakie wybierzemy, muszą w odpowiedni sposób radzić sobie z tymi zadaniami. Rozwiązaniami jakie będą porównywane w tym miejscu, są to rozwiązania z platform .NET, Java oraz rozwiązanie javascriptowe - Node.js.

Platforma .NET jest to dobrze znana platforma rozwijana przez firmę Microsoft. Opiera się ona głównie na języku C#, jednak dzięki narzędziu Common Language Runtime deweloper ma możliwość programować także w innych językach programowania. Platforma .NET oferuje bardzo szerokie spectrum narzędzi dla tworzenia różnego rodzaju zadań i projektów. Narzędziem do tworzenia aplikacji webowych jest framework ASP.NET, który dostarcza wiele mechanizmów służących do tworzenia aplikacji przeznaczonych do różnego rodzaju zadań.

Biorąc pod uwagę, że warstwa serwerowa ma jedynie przetwarzać dane otrzymane z bazy danych i przesyłać je dalej do warstwy prezentacyjnej, rozwiązaniem jakie najlepiej pasuje do tego scenariusza, jest rozwiązanie RESTowe – aplikacja ASP.NET Core Web API [10].

Drugą platformą, konkurencyjną dla .NET, jest platforma Java. Platforma opiera się głównie na języku Java. Jest to rozwiązanie open-sourcowe z dużą ilością użytkowników biorących czynny udział w rozwoju platformy Java w internecie. Frameworkiem odpowiedzialnym za aplikacje webowe w ramach tej platformy jest Java EE. W ramach tego frameworka, również istnieje możliwość zbudowania RESTowej aplikacji, która jest rozwiązaniem, jakie aplikacja tworzona w ramach niniejszej pracy, będzie wykorzystywała[11].

Node.js jest to środowisko pozwalające na uruchomienia kodu javascriptowego po stronie serwera. Środowisko te zostało stworzone przez Ryana Dahla. Node.js działa na silniku javascriptowym Chrome V8. Opiera się on na zdarzeniach oraz słuchaczach zdarzeń. Pomimo, że w teorii Node.js jest jednowątkowy, dzięki pętli zdarzeń potrafi on wykonać wiele żądań w jednym momencie. Dzięki temu, Node.js jest bardzo efektownym i elastycznym narzędziem, pozwalającym na budowanie dużych i skalowalnych systemów [12].

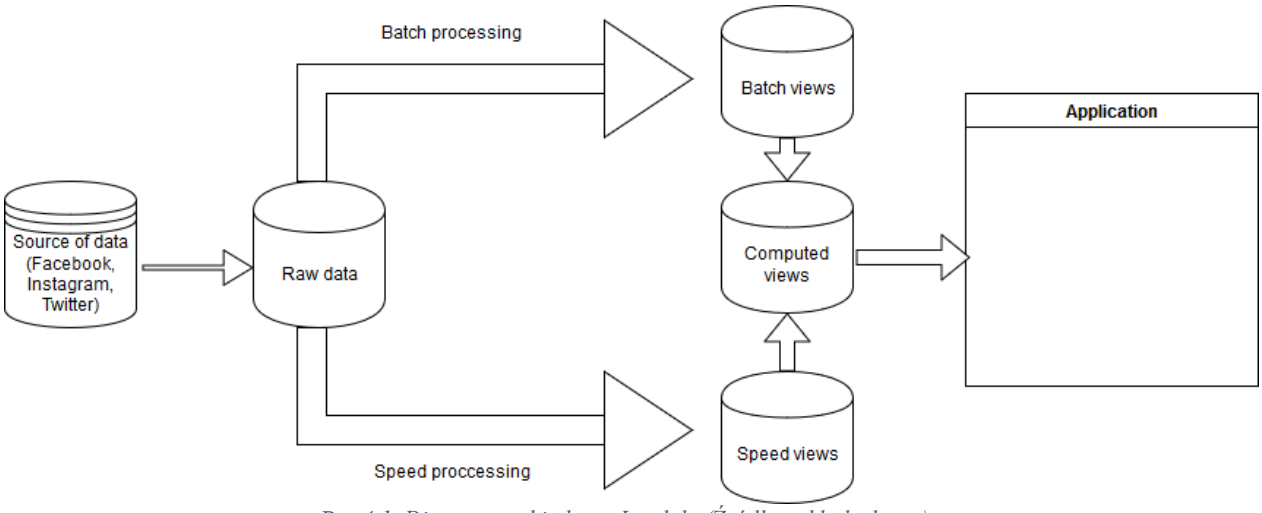
Porównanie trzech, wyżej wymienionych platform jest zadaniem bardzo trudnym. Biorąc pod uwagę różnicę podejścia, użytkownik nie jest w stanie jednoznacznie wskazać środowisko, które w będzie sprawowało się lepiej w każdej sytuacji. Biorąc pod uwagę fakt, że każda z tych platform identyfikuje się konkretnym językiem programowania, z pewnością wybór będzie kierowany wygodą oraz subiektywną oceną języka. Czynnikiem, który z pewnością jest znaczący przy wyborze podejścia, jest wsparcie społeczności internetowej w ramach każdego z narzędzi. Jednakże wszystkie trzy wyżej wymienione, biorąc pod uwagę, że są w tym momencie najbardziej popularnymi rozwiązaniami dla aplikacji serwerowych, posiadają bardzo duże wsparcie w internecie. Dla aplikacji tworzonej w ramach tej pracy, platforma jaka została wybrana, jest to platforma .NET. Wymagania postawione dla warstwy serwerowej, najlepiej będą realizowane za pomocą aplikacji w architekturze REST – ASP.NET Core WEB API.

4.3. Warstwa bazodanowa

Ostatnim wyborem technologicznym, jest wybór w warstwie bazodanowej. W aplikacji tworzonej w ramach niniejszej pracy, procesowanie danych oraz radzenie sobie z ilością wczytywanych informacji do bazy danych jest głównym problemem do rozwiązania.

Sposoby przetrzymywania i zapisywania danych znacznie zmieniły się na przestrzeni ostatnich kilkudziesięciu lat. Wpierw był to zapis i odczyt z pliku, jednakże ze względu na rosnącą liczbę danych nie był to wystarczający sposób. Dlatego w pewnym momencie przedstawiono serwery bazodanowe wraz ze środowiskami zarządzającymi tymi serwerami. Optymalizacje oraz ulepszenia tych serwerów pozwalają na skuteczne radzenie sobie z zapisem i odczytem danych dla większości aplikacji. Jednakże rozrost sieci, serwerów oraz aplikacji, które co raz częściej posiadają wymóg radzenia sobie z ogromną liczbą informacji spowodował, że nawet klasyczne użycie bazy danych nie wystarcza to optymalnego przetwarzania tych informacji. Ze względu na to powstały schematy i rozwiązania tak zwanych hurtowni danych pozwalających na analizę i przetworzenie duży ilości danych. Zaistniały także rozwiązania z grupy nazwanej NOSQL, które w pewnym sensie rozwiązały wiele problemów, jednakże nie są skuteczne we wszystkich przypadkach. Do problemu ilości przetwarzanych danych, dochodzi także kwestia skalowalności tworzonego systemu, która przy klasycznych serwerach SQL jest bardzo trudne do utrzymania. Ze względu na wszystkie, wyżej wymienione kwestie, zaproponowana została architektura nazwaną Architekturą Lambda.

Wzorzec Lambda Architecture opiera się na podziale warstwy bazodanowej na dwie współdziałające części. Architektura taka powinna być efektywna, odporna na błędy oraz skalowalna. Biorąc pod uwagę ilość danych, które mogą trafić do aplikacji, stworzona struktura musi zapewniać dostęp do odpowiednich informacji w akceptowalnym dla użytkownika czasie.



Rys.4.1. Diagram architektury Lambda (Źródło: wkład własny).

Rys.4.1. przedstawia ogólny zarys i idee, która ukazuje na czym polega wyżej wspomniana architektura. Posiada ona dwie główne warstwy - Speed oraz Batch. Pierwsza z nich, jest to miejsce, gdzie ładowane są dane bez przetwarzania, które od razu są gotowe do zaczytania. Warstwa Speed przygotowuje raporty z informacji, które zostały zaczytane od ostatniego przetworzenia przez warstwę Batch do chwili bieżącej. Drugą warstwą jest warstwa Batch, w której zaimplementowane są procedury przetwarzające, łączące oraz przygotowujące raporty dla aplikacji. Odpowiedzią na żądania klienta są dane pochodzące z połączenia widoków warstw Speed oraz Batch. Podejście takie, pomaga w dostępie do danych w czasie rzeczywistym, za pomocą informacji z części Speed oraz do zaawansowanych raportów przygotowanych przez część Batch. Informacje takie są jednak dostępne z pewnym opóźnieniem pozwalającym te raporty wygenerować.

Implementacja Architektury Lambda może zostać dokonana na wiele sposobów. Ważną rzeczą, o której należy pamiętać, jest dobranie odpowiednich technologii do warstwy Batch i warstwy Speed. Warstwa Batch potrzebuje narzędzia posiadającego zoptymalizowane operacje procesowania i wyliczania potrzebnych widoków na podstawie dostępnych informacji. Natomiast warstwa Speed, powinna zostać zaimplementowana za pomocą narzędzia, które udostępnia dane w jak najkrótszym czasie.

Procesowanie oraz radzenie sobie z dużą ilością danych, ciągle jest dużym problemem dla tworzonych systemów informatycznych. By ułatwić to zadanie, zostały stworzone mechanizmy takie jak Hadoop, Cassandra czy framework MapReduce. Rozwiązaniem dla podobnych problemów, jest Apache Spark. Apache Spark jest open-sourcową platformą, stworzoną na Uniwersytecie Kalifornii. Platforma ta stosuje klastrowe przetwarzanie danych, opierając się na rozproszonych zbiorach danych RDD (Resilient Distributed Dataset). Posiada ona także interfejsy API dla różnych języków programowania, w których można pisać operacje przetwarzania, procesowania czy transformacji dostępnych danych.
W aplikacji tworzonej w ramach niniejszej pracy, do przetrzymywania danych będzie wykorzystany serwer PostgreSQL. Procesowanie w warstwie Batch, będzie dokonywane za pomocą Apache Sparka oraz języka Scala, natomiast warstwą Speed, będą widoki tworzone za pomocą PostgreSQLa.

4.4 Podsumowanie

Wybór technologii, za pomocą których zostanie wykonana aplikacja, jest bardzo ważnym etapem, w którym podejmuje się decyzję o kształcie budowanego rozwiązania. Zły dobór może skutkować nieoptymalnie działającą aplikacją. W ramach tego rozdziału, zostały przedstawione trzy warstwy, które wymagają doboru technologii. Pierwszą warstwą, jest warstwa kliencka, która odpowiada za komunikację z użytkownikiem. Biorąc pod uwagę, że tworzona aplikacja będzie aplikacją webową, wybór w tej warstwie ograniczył się do wyboru frameworka javascriptowego. Po porównaniu najpopularniejszych frameworków, do implementacji warstwy klienckiej, został wybrany Angular. Drugą rozpatrywaną warstwą, jest warstwa serwerowa. Bardzo ważnym aspektem przy stronie serwera, jest optymalna szybkość komunikacji, pomiędzy serwisami zewnętrznymi, bazą danych oraz klientem. Technologią wybraną do implementacji strony serwera jest .NET Core. Ostatnim etapem wyboru technologii jest wybór dla warstwy bazodanowej. W tym przypadku, potrzeba było dobrać odpowiednio zbudowaną architekturę, pozwalającą na przetwarzanie dużej ilości danych w akceptowalnym przez użytkownika czasie oraz udostępniającą część aktualnych danych. Po przeanalizowaniu możliwych rozwiązań, wybrana została Architektura Lambda. Do jej implementacji posłużą PostgresSQL oraz Apache Spark

5. Projekt aplikacji

Tworzenie aplikacji internetowej jest zadaniem złożonym z wielu elementów. Brak wcześniej zdefiniowanych wymagań względem projektu, powoduje bardzo duże trudności w utworzeniu poprawnych funkcjonalności oraz może prowadzić do popełnienia niechcianych błędów. Spojrzenie na projekt jako całość oraz zamodelowanie współdziałania wszystkich elementów systemu jest niezbędne do prawidłowego zaimplementowania mechanizmów zawartych w produkcie.

Utworzenie projektu aplikacji składa się z kilku części. Każda następna część jest zależna od założeń i wniosków wcześniej stworzonej. W pierwszej kolejności należy wykonać analizę rzeczywistości, której aplikacja dotyczy. Służy to zrozumieniu działania narzędzi oraz zjawisk występujących w obrębie danej dziedziny. Drugim krokiem jest opisanie ogólnej wizji aplikacji, gdzie ma miejsce wyodrębnienie ról użytkowników działających w systemie oraz zakresu uprawnień, które do tych ról są przypisane. Następnym elementem jest opisanie tych uprawnień w perspektywie ról w formie historyjek użytkowników. Stworzone wcześniej etapy są podstawą do wygenerowania diagramu przypadków użycia, który w bardzo przejrzysty sposób pokazuje wszystkie funkcjonalności systemu powiązane z danymi rolami użytkowników. Ostatnim krokiem jest stworzenie prototypu interfejsu, który jest podstawą dla widoków dostępnych użytkownikowi docelowemu. Posiadając tak zdefiniowany projekt, z łatwością można zabrać się za implementację wszystkich mechanizmów opierając się na elementach opisanych powyżej. W tym rozdziale opisany będzie projekt aplikacji tworzonej w ramach tej pracy, w wyżej wymienionych krokach.

5.1. Opis rzeczywistości

Portale społecznościowe są to aplikacje webowe ułatwiające komunikację pomiędzy ludźmi za pomocą Internetu. Z rozwojem sieci, portale z tej grupy, automatycznie zyskały na popularności. Użytkownicy takich witryn, udostępniają wiele informacji, które na swój sposób dokumentują oraz określają daną osobę, co może pozwolić na zbudowanie pewnego obrazu osoby, nie znając jej w realnym świecie. Aplikacja tworzona w ramach tej pracy, analizuje dane pochodzące z trzech portali społecznościowych – Facebooka, Twittera oraz Instagrama.

Facebook jest to portal społecznościowy stworzony w 2004 roku przez Marka Zuckenberga pod oryginalną nazwą TheFacebook. Z początku miał być to portal ułatwiający komunikację studentów szkół wyższych w Stanach Zjednoczonych, jednakże Facebook z czasem rozrósł się do aplikacji społecznościowej używanej na całym świecie. Za pomocą tej aplikacji, każda osoba może zarejestrować swoje konto, tworząc profil, który uzupełnia informacjami, jakimi chce podzielić się z innymi użytkownikami. Użytkownik może zapraszać profile należące do innych osób, tworząc w ten sposób sieć osób, z którą chce dzielić się wydarzeniami, filmikami czy zdjęciami zamieszczonymi w Internecie.

Twitter jest to druga aplikacja, analizowana w ramach tej pracy, należąca do grupy portali społecznościowych, stworzona przez Jacka Dorseya w 2006 roku. Opiera się ona na krótkich wiadomościach, zwanymi „tweetami", za pomocą których użytkownicy mogą wyrażać opinie oraz dzielić się nimi z innymi. Tworząc profil, udostępnia się go dla każdego użytkownika posiadającego konto, mogącego obserwować daną osobę, powodując, że wszystkie opinie będą widoczne na jego „ścianie" aktywności. Bardzo ważną częścią Twittera są hashtagi. Hashtag jest to słowo kluczowe wskazujące na jakiś temat, poprzedzone znakiem hasha. Przykładem może być #PWR czy #Mistrzostwa. Za ich pomocą, opinia może zostać zakwalifikowana do danego tematu, co ułatwia obserwację reakcji ludzi na dany temat. Co prawda Twitter nie jest najbardziej popularną witryną społecznościową, jednakże ilość danych jakie są wysyłane przez

użytkowników na serwery Twittera oraz charakter tej witryny powoduje, że można z niej „wyciągnąć” bardzo interesujące informacje pokazujące stosunek użytkownika Twittera do różnych aspektów życia.

Instagram jest to ostatni portal społecznościowy, rozpatrywany w ramach tej pracy, stworzony przez Kevina Systroma oraz Mike’a Kriegera w 2010 roku. Głównym zasobem, jaki jest udostępniany przez użytkowników Instagrama są zdjęcia, które za pomocą wyżej opisanych hashtagów tworzą wpisy na danym profilu. Użytkownik, w zależności od ustawień prywatności konta, może śledzić innych użytkowników dostając powiadomienia o najnowszych wpisach. Każdy profil jest to historia wpisów, które zostały załadowane. Wpis może posiadać dane takie jak lokalizacja, w której zdjęcie zostało zrobione, czas zrobienia zdjęcia czy właśnie hashtagi opisujące czego zdjęcie dotyczy.

Dane, zebrane z trzech wyżej wymienionych portali, mają być podstawą do tworzenia raportów na temat użytkownika. Łącząc wszystkie informacje pobrane z tych portali można wygenerować profil osoby, opisujący jego upodobania, charakter czy historię pobytu w różnych lokalizacjach.

5.2. Wizja systemu

Projekt który jest tworzony w ramach tej pracy, ma służyć do analizy danych z trzech portali społecznościowych – Facebooka, Twittera oraz Instagrama. Załadowane dane z tych witryn mają być podstawą do stworzenia raportów z czterech kategorii – dane personalne, lokalizacja, aktywność internetowa oraz charakter. W systemie tworzonym w ramach tej pracy wyodrębnia się dwa rodzaje użytkowników – zalogowany oraz niezalogowany.

Niezalogowany użytkownik ma dostęp jedynie do akcji logowania do aplikacji, za pomocą jednego z trzech, wyżej wymienionych portali. Logowanie będzie przeprowadzone poprzez zewnętrzne mechanizmy należące do użytych aplikacji społecznościowych. System budowany w ramach tej pracy, zapisuje dane logowania przesłane przez zewnętrzne aplikacje, przetrzymując je w sesji użytkownika.

Zalogowany użytkownik dostaje dostęp do funkcjonalności dodania osoby, którą zamierza obserwować. Gdy taka osoba jest już zdefiniowana, widoczne są strony raportów przedstawiające różne, zebrane dane dotyczące obserwowanej osoby. By uzyskać dostęp do aktualnych danych należy odświeżyć je dla danego portalu społecznościowego. W sytuacji gdy użytkownik jest niezalogowany w ramach odświeżenia danych dla którejś z zewnętrznych aplikacji, jest on przekierowywany na stronę logowania. Raporty są podzielone na cztery kategorie:

- Dane personalne – podstawowe informacje takie jak imię, nazwisko, data urodzenia.
- Lokalizacja – dane dotyczące miejsca przebywania danego użytkownika. Pokazuję miejsce urodzenia, zamieszkania oraz przedstawione na mapie miejsca w których obserwowany użytkownik przebywał.
- Aktywność internetowa – raporty przedstawiające informacje na temat aktywności danego użytkownika. Raporty z tej kategorii przedstawiają dane takie jak godziny największej aktywności, czy najpopularniejsze tematy interesujące danego użytkownika.
- Charakter – raporty przedstawiające informacje na temat charakteru użytkownika. W tej kategorii raportów, istnieją raporty takie jak zgodność charakteru pomiędzy zalogowanym użytkownikiem i obserwowaną osobą czy stosunek negatywnych opinii do pozytywnych udostępnianych w Internecie.

Każda obserwowana osoba przez zalogowanego użytkownika będzie miała przypisane do siebie strony z wyżej wymienionymi raportami oraz służący do nawigacji dashboard, który będzie zawierał przyciski do aktualizacji danych. Zalogowany użytkownik będzie w stanie nawigować pomiędzy tymi raportami oraz usuwać i dodawać nowych obserwowanych użytkowników.

5.3. Historyjki użytkowników

Na podstawie wyżej opisanej wizji systemu, zostały zdefiniowane następujące historyjki użytkowników:

Klient niezalogowany:

- Jako **klient niezalogowany** chciałbym mieć możliwość zalogowania się do swojego konta

Klient zalogowany:

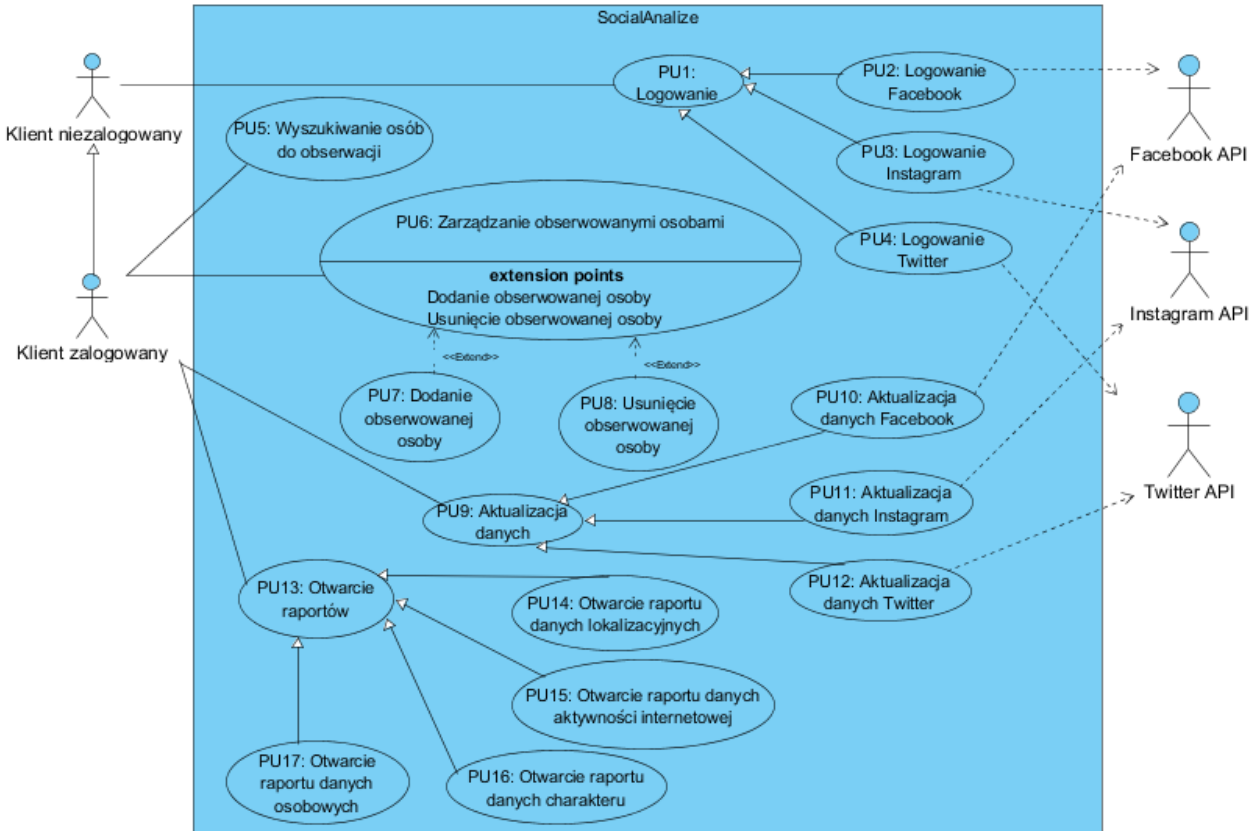
- Jako **klient zalogowany** chciałbym mieć możliwość wyszukania osoby do obserwacji
- Jako **klient zalogowany** chciałbym mieć możliwość dodania nowej obserwowanej osoby
- Jako **klient zalogowany** chciałbym mieć możliwość usunięcia obserwowanej osoby
- Jako **klient zalogowany** chciałbym mieć możliwość otworzenia raportu danych personalnych osoby obserwowanej
- Jako **klient zalogowany** chciałbym mieć możliwość otworzenia raportu dotyczącego danych lokalizacji osoby obserwowanej
- Jako **klient zalogowany** chciałbym mieć możliwość otworzenia raportu dotyczącego danych aktywności internetowej osoby obserwowanej
- Jako **klient zalogowany** chciałbym mieć możliwość otworzenia raportu dotyczącego danych charakteru osoby obserwowanej
- Jako **klient zalogowany** chciałbym mieć możliwość nawigacji pomiędzy reportami oraz osobami obserwowanymi
- Jako **klient zalogowany** chciałbym mieć możliwość aktualizowania danych
- Jako **klient zalogowany** chciałbym mieć możliwość wylogowania się z aplikacji

Historyjki opisane powyżej, są podstawowym opisem uprawnień w perspektywie ról w aplikacji budowanej w ramach tej pracy.

5.4. Diagram przypadków użycia

Diagram przypadków użycia poprzez graficzne symbole opisuje budowany system oraz jego wymagania funkcjonalne. Podstawowym elementem takiego diagramu jest przypadek użycia, który składa się ze zbioru działań dostępnych dla danego użytkownika. Za jego pomocą, zdefiniowane są wymagania funkcjonalne tworzonego systemu. Drugim elementem wykorzystywanym w diagramie przypadków użycia jest aktor. Aktor reprezentuje użytkownika w konkretnej roli. Za pomocą asocjacji definiuje się, do których przypadków użycia dany aktor ma uprawnienia. Możliwe jest także dziedziczenie pomiędzy aktorami. Ostatnim elementem użytym w niżej przedstawionym diagramie, jest system zewnętrzny reprezentujący mechanizmy spoza ram aplikacji.

Zdefiniowanie wizji systemu oraz uprawnień dla ról w postaci historyjek użytkowników, daje podstawę do stworzenia diagramu przypadków użycia, który dla aplikacji tworzonej w ramach tej pracy przedstawiono na rysunku 5.1:



Rys.5.2. Diagram przypadków użycia dla aplikacji tworzonej w ramach tej pracy.

Opis elementów zamieszczonych na diagramie przypadków użycia dotyczącego aplikacji tworzonej w ramach tej pracy znajduje się w tabeli (Tab. 5.1)

Tab. 5.1. Opis diagramu przypadków użycia

Nazwa elementu	**Znaczenie**
Przypadki użycia	
Logowanie	Zbiorczy przypadek użycia reprezentujący akcję logowania
Logowanie Facebook	Przypadek użycia reprezentujący akcję logowania za pomocą Facebooka
Logowanie Instagram	Przypadek użycia reprezentujący akcję logowania za pomocą Instagrama
Logowanie Twitter	Przypadek użycia reprezentujący akcję logowania za pomocą Twittera
Wyszukiwanie osób do obserwacji	Przypadek użycia reprezentujący akcję wyszukiwania osób do obserwacji
Zarządzanie obserwowanymi osobami	Zbiorczy przypadek użycia reprezentujący akcję zarządzające osobą obserwowaną
Dodanie obserwowanej osoby	Przypadek użycia reprezentujący akcję dodania nowej obserwowanej osoby
Usunięcie obserwowanej osoby	Przypadek użycia reprezentujący akcję usunięcia obserwowanej osoby
Aktualizacja danych	Zbiorczy przypadek użycia reprezentujący akcję aktualizacji danych za pomocą jednego z trzech portali społecznościowych

Aktualizacja danych Facebook	Przypadek użycia reprezentujący akcję aktualizacji danych za pomocą Facebooka
Aktualizacja danych Instagram	Przypadek użycia reprezentujący akcję aktualizacji danych za pomocą Instagrama
Aktualizacja danych Twitter	Przypadek użycia reprezentujący akcję aktualizacji danych za pomocą Twittera
Otwarcie raportów	Zbiorczy przypadek użycia reprezentujący akcję otwarcia reportu
Otwarcie raportu danych lokalizacyjnych	Przypadek użycia reprezentujący akcję otwarcia reportu danych lokalizacyjnych
Otwarcie raportu danych aktywności internetowej	Przypadek użycia reprezentujący akcję otwarcia raportu danych aktywności internetowej
Otwarcie raportu danych charakteru	Przypadek użycia reprezentujący akcję otwarcia raportu danych dotyczących charakteru
Otwarcie raportu danych osobowych	Przypadek użycia reprezentujący akcję otwarcia reportu danych osobowych
Aktorzy	
Klient niezalogowany	Aktor reprezentujący klienta niezalogowanego
Klient zalogowany	Aktor reprezentujący klienta po zalogowaniu za pomocą jednego z portali społecznościowych
Systemy zewnętrzne	
Facebook API	System zewnętrzy Facebooka. Służy do logowania oraz pobierania danych.
Instagram API	System zewnętrzny Instagrama. Służy do logowania oraz pobierania danych
Twitter API	System zewnętrzny Twittera. Służy do logowania oraz pobierania danych

W celu stworzenia pełnego opisu diagramu przypadków użycia należy zdefiniować scenariusze z nim powiązane. Pozwalają one na dokładne pokazanie kroków, wchodzących w skład konkretnej funkcji aplikacji. Scenariusze przypadków użycia zostały pokazane w tabeli 5.2.

Tab 5. 2. Scenariusze przypadków użycia

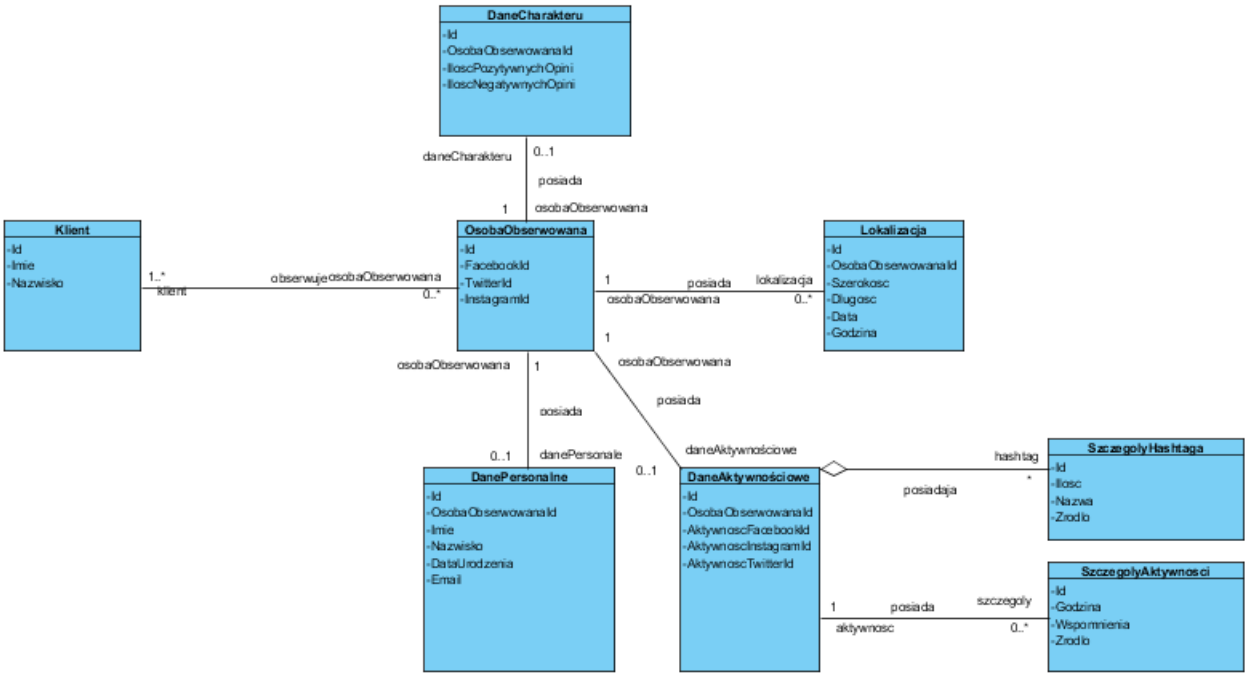
	Akcja aktora	Akcja systemu
1	Użytkownik klika na jeden z przycisków logowania (**PU1**)	
2		Następuje przekierowanie do zewnętrznego, wybranego serwisu logowania portalu społecznościowego
3		Po wpisaniu loginu i hasła na zewnętrznym serwisie, system przekierowuje użytkownika na stronę dla zalogowanego użytownika
4	Użytkownik naciska przycisk dodania nowej obserwowanej osoby	
5		System przekierowuje użytkownika na stronę dodawania
6	Użytkownik wpisuje identyfikator szukanej osoby (**PU5**)	
7		System przedstawia wyniki wyszukiwania
8	Użytkownik naciska przycisk dodania nowej obserwowanej osoby obok odpowiedniego wyniku wyszukiwania (**PU7**)	
9		System dodaje obserwowaną osobę w ramach użytkownika
4a1	Użytkownik naciska na awatar jednej z obserwowanych osób	
4a2		Następuje przekierowanie do widoku nawigacyjnego obserwowanej osoby
4a3	Użytkownik odświeża dane za pomocą jednego z przycisków, odpowiadającemu konkretnemu portalowi społecznościowemu (**PU9**)	
4a4		System odświeża dane
4a4a1		W przypadku gdy użytkownik jest niezalogowany w ramach konkretnego portalu następuje przekierowanie do zewnętrznej strony logowania
4a4a2		Po wpisaniu loginu i hasła na zewnętrznym serwisie, system przekierowuje użytkownika na stronę nawigacyjną wcześniej wybranej obserwowanej osoby
4a3a1	Użytkownik naciska przycisk związany z otwarciem konkretnego raportu(**PU13**)	
4a3a2		System przekierowuje użytkownika na stronę wybranego raportu

4a3b1	Użytkownik naciska przycisk „Usuń osobę obserwowaną”(**PU8**)	
4a3b2		System wyświetla okno dialogowe potwierdzające wybór
4a3b3	Użytkownik wybiera „Tak”	
4a3b4		System usuwa osobę obserwowaną
4a3b3a1	Użytkownik wybiera „Nie”	
4a3b3a2		Następuje przekierowanie na stronę nawigacyjną wybranej, obserwowanej osoby

Rys. 5.1 przestawiający diagram przypadków użycia dla aplikacji budowanej w ramach tej pracy, jest podstawowym diagramem, na który opierana będzie wizja działania systemu. Posiada on wszystkie wymagania funkcjonalne względem aplikacji oraz uprawnienia zdefiniowanych ról do konkretnych przypadków użycia.

5.5. Diagram klas

Następnym krokiem w tworzeniu kompletnego projektu systemu, jest przedstawienie diagramów, ukazujących techniczne aspekty systemu. Implementacja wymagań funkcjonalnych jest to zadanie, które musi zostać odpowiednio zaplanowane, w celu uniknięcia ewentualnych błędów. Narzędziem służącym do wstępnego przedstawienia systemu z perspektywy obiektów i relacji występujących pomiędzy nimi, jest diagram klas. Dla aplikacji tworzonej w ramach tej pracy, diagram klas został przedstawiony na Rys.5.2.



Rys.5.3. Diagram klas dla aplikacji

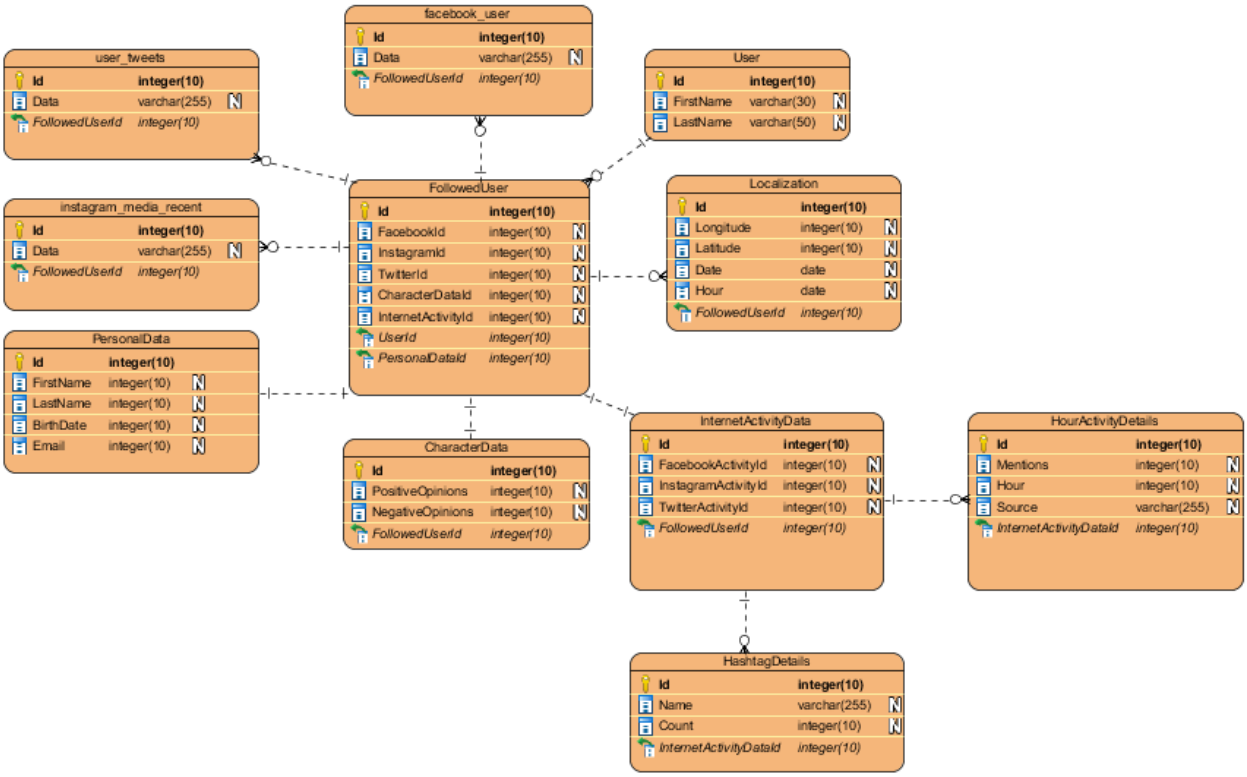
Powyższy diagram (Rys.5.2) pokazuje główne klasy z jakich będzie składała się aplikacja. Pierwszą klasą na którą należy zwrócić uwagę, jest klient reprezentujący zalogowanego użytkownika. Następnie najważniejszą, centralną klasą jest osoba obserwowana. Posiada ona atrybuty składające się z identyfikatora osoby w systemie oraz identyfikatorów dla poszczególnych portali społecznościowych. Obiekt osoby obserwowanej jest połączony z klasami zawierającymi dane dotyczące tej osoby, będące prezentowane na odpowiednich raportach. Takimi klasami są DaneCharakteru, Lokalizacja, DanePersonalne oraz DaneAktywnościowe. Ostatni obiekt jest powiązany ze szczegółami dotyczącymi raportu, czyli

SzczegolyHashtaga oraz SzczegolyAktywnosci. Dane powiązane z tymi klasami posłużą do wyświetlenia odpowiadających wykresów.

Diagram klas bardzo dobrze pokazuje system w sposób, pozwalający na łatwą jego implementację w języku obiektowo zorientowanym. Pozwala to na wyeliminowanie wstępnie występujących błędów i nieścisłości, występujących w trakcie implementacji [13].

5.6. Diagram encji

Po opisaniu wymagań funkcjonalnych oraz stworzeniu diagramu klas, który pozwala na pokazanie systemu za pomocą obiektów i relacji między nimi, następnym etapem jest zamodelowanie struktury bazy danych. W tym celu, zostanie wykorzystany diagram encji. W dużej liczbie przypadków, diagram encji jest podobny do diagramu klas, jednakże w systemie tworzonym w ramach tej pracy, biorąc pod uwagę rozbudowaną warstwę bazodanową, będzie on nieznacznie się różnił. Diagram encji wygenerowany dla aplikacji tworzonej w ramach tej pracy jest przedstawiony na Rys.5.3.

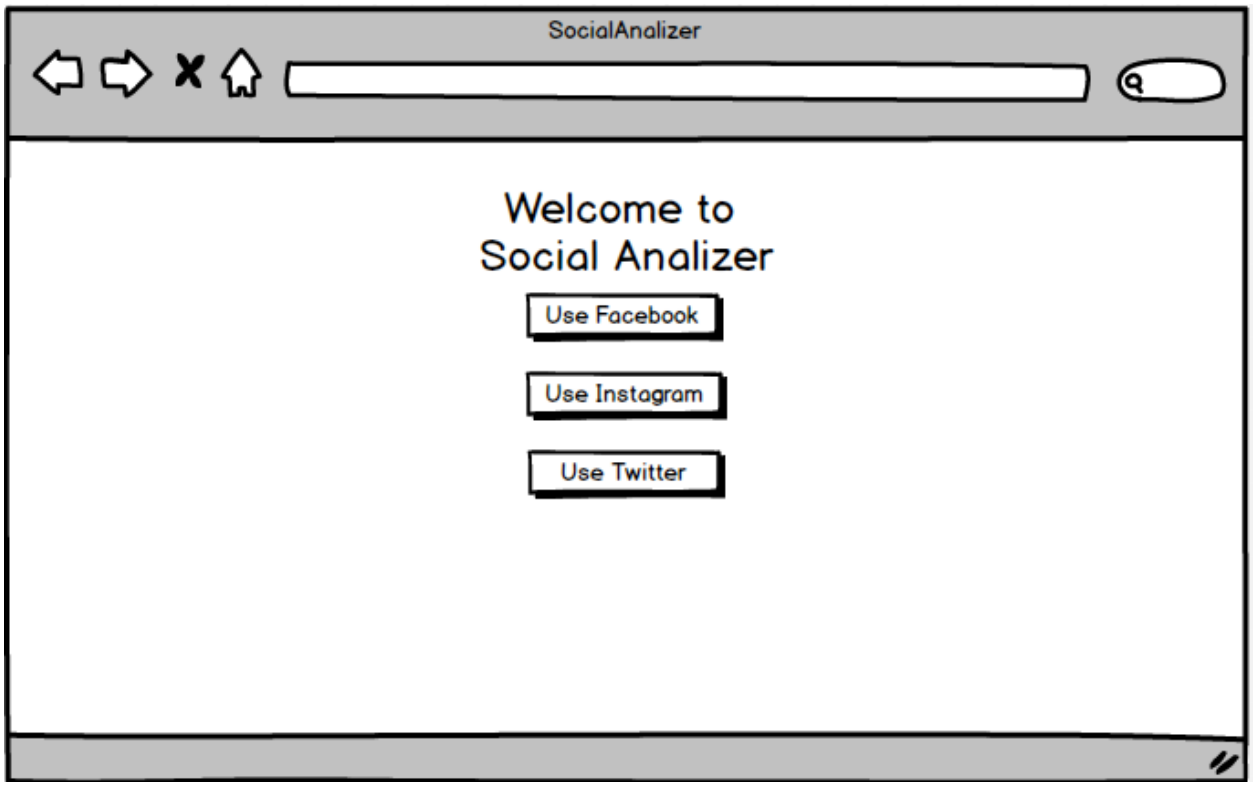


Rys.5.4. Diagram encji dla aplikacji

Podstawowym pojęciem, zawiązanym z wyżej przedstawionym diagramem (Rys.5.3.) jest encja. Encją nazywany jest „istniejący obiekt, rozróżnialny od innych bytów tego samego typu”[14]. Każda encja jest powiązana z innymi encjami związkiem, składającym się z nazwy, liczebności i opcjonalności. Na rysunku 5.3 przedstawione zostały encje wraz z relacjami, występujące w aplikacji. Podobnie jak w diagramie klas, występuje tutaj encja User, która reprezentuje użytkownika zalogowanego do aplikacji. Centralnym punktem diagramu encji przedstawionego na rysunku 5.3 jest encja FollowedUser, definiująca osobę obserwowaną. Trzema obiektami, które nie były uwzględnione w diagramie klas, są user_tweets, instagram_media_recent oraz facebook_user. Są to obiekty przechowujące dane pochodzące wprost z zewnętrznych serwerów portali społecznościowych. Na podstawie tych danych Apache Spark przygotowuje raporty, które następnie zapisuje w kolejnych widocznych encjach. Tymi encjami są CharacterData, Location oraz InternetActivityData wraz z powiązanymi szczegółami.

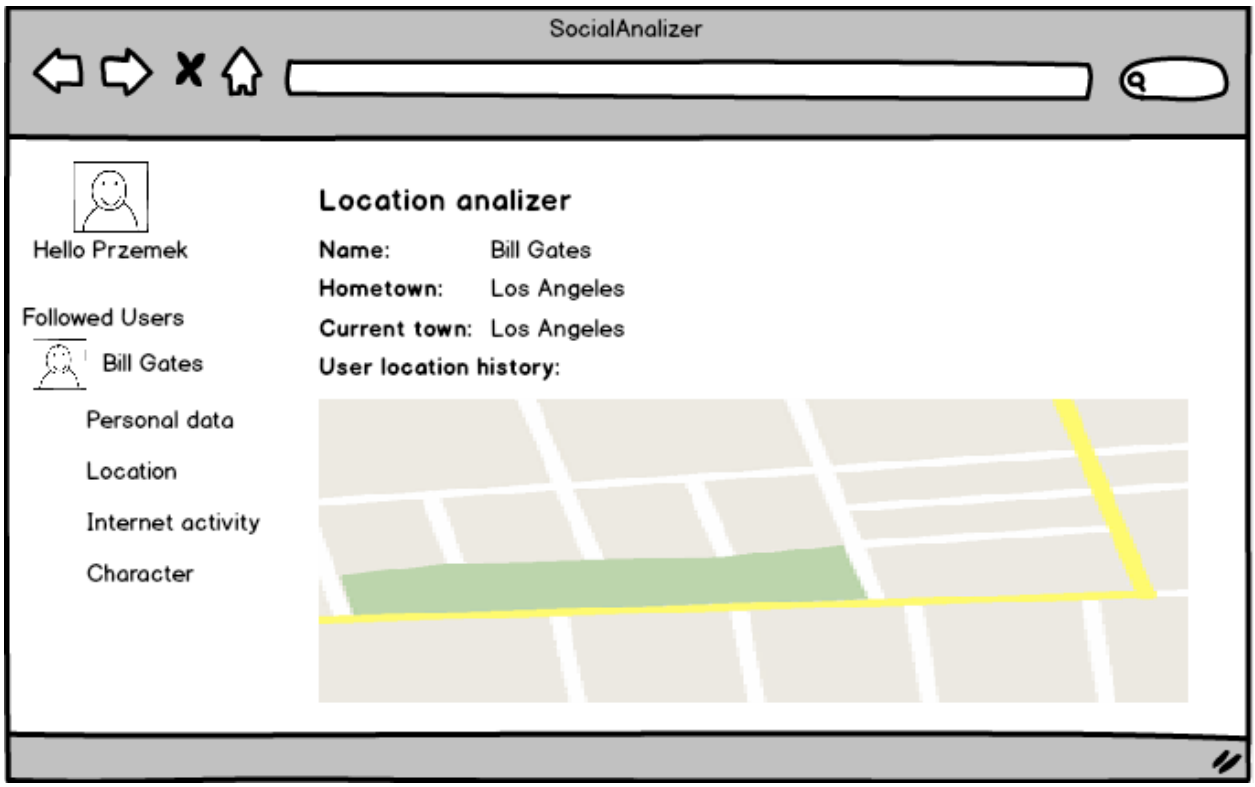
5.7. Prototyp interfejsu

Ostatnim etapem projektowania, jest przedstawienie prototypu interfejsu w formie graficznej. Biorąc pod uwagę, że zadowolenie klienta korzystającego z aplikacji jest mocno powiązane z interfejsem, stworzenie takiego prototypu jest bardzo pomocne. Posiadając przykłady widoków tworzonej witryny, przed rozpoczęciem implementacji, można zdobyć opinie potencjalnych użytkowników aplikacji oraz na ich podstawie dokonać popraw i usprawnień. Dzięki temu, można znacznie zwiększyć prawdopodobieństwo zadowolenia użytkowników tworzonego systemu. Poniżej (Rys.5.4, Rys.5.5, Rys.5.6) są przedstawione przykładowe prototypy trzech stron aplikacji tworzonej w ramach tej pracy.



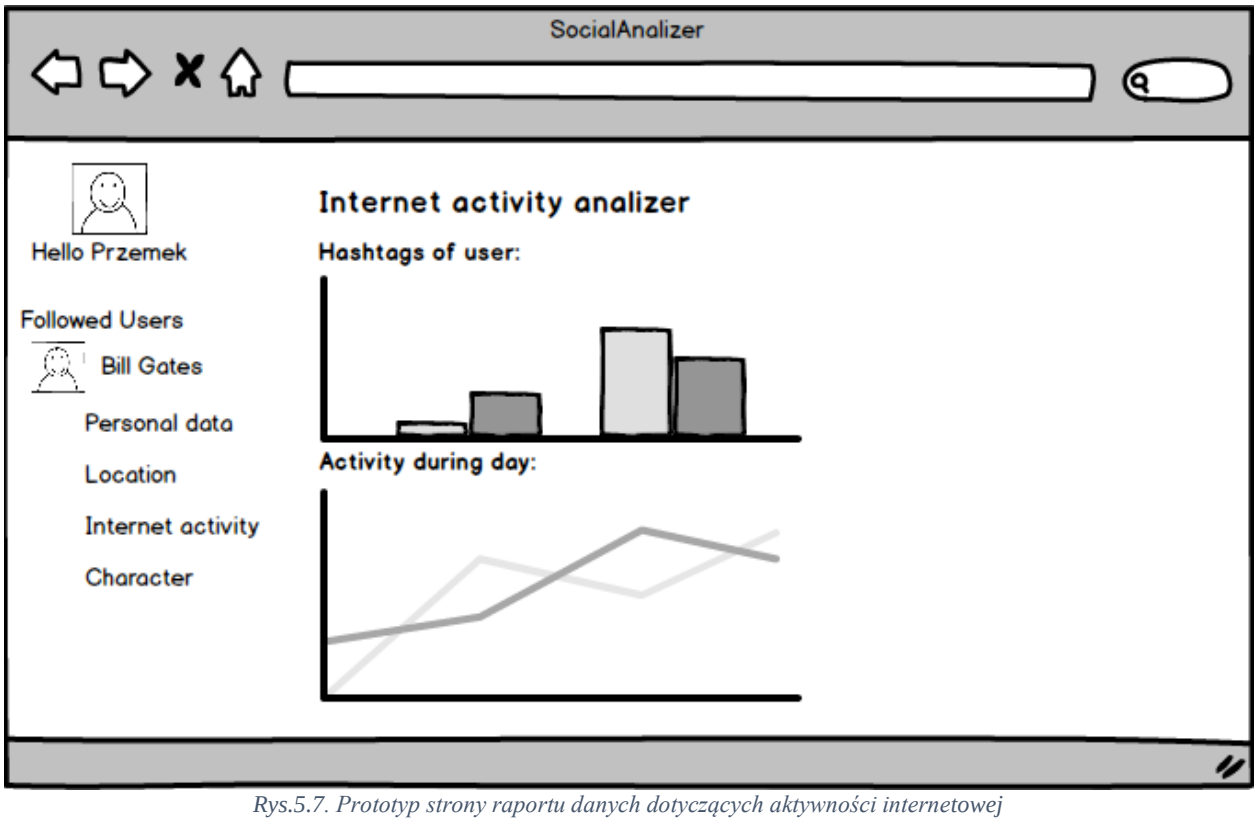
Rys.5.5. Prototyp strony logowania

Pierwszy prototyp przedstawia stronę widoczną dla każdego klienta korzystającego z systemu. Jest to prosta strona zawierająca powitanie, oraz możliwość zalogowania się za pomocą jednego z trzech portali społecznościowych, analizowanych przez system. Po wybraniu jednego z portalu, użytkownik zostaje przekierowany do zewnętrznej strony, a następnie po wpisaniu loginu i hasła, zostaje on przekierowany do strony dla zalogowanego użytkownika, który może otworzyć jeden z raportów dotyczących obserwowanych użytkowników.



Rys.5.6. Prototyp strony raportu danych lokalizacyjnych

Jednym z raportów, jaki jest dostępny dla zalogowanego użytkownika, jest raport dotyczący lokalizacji, którego prototyp jest przedstawiony na Rys.5.5. Komponentem wspólnym dla stron dostępnych po zalogowaniu jest boczny panel nawigacyjny, pozwalający z łatwością otworzyć interesujące klienta raporty. Strona z danymi dotyczącymi lokalizacji, zawiera nagłówek, krótkie informacje dotyczące wybranego użytkownika oraz mapę, na której będą nałożone znaczniki przypisane do konkretnej lokalizacji, posiadające czas i datę przebywania obserwowanej osoby w zaznaczonym miejscu.



Rys.5.7. Prototyp strony raportu danych dotyczących aktywności internetowej

Ostatnim przedstawionym prototypem, jest prototyp strony raportu (Rys.5.6), dotyczącego aktywności użytkownika w internecie. Strona ta zawiera wcześniej wspomniany boczny panel nawigacyjny oraz dwa komponenty, składające się z wykresów, przedstawiające informacje dotyczące użytych przez użytkownika hashtagów oraz godzinowej aktywności w ciągu dnia.

5.8. Podsumowanie

Tworzenie aplikacji składającej się z wielu elementów jest zadaniem bardzo złożonym. Trudno jest zacząć od razu implementację bez wcześniejszego zaplanowania i przemyślenia pracy. Dobre zaprojektowanie systemu, pozwala na uniknięcie wielu błędów oraz na stworzenie spójnej wizji systemu. Analityka biznesowa dostarcza wielu narzędzi, które mogą posłużyć na przedstawienie różnych aspektów systemu, patrząc z wielu perspektyw. W tym rozdziale został przedstawiony zarys projektu aplikacji, zawierający opis rzeczywistości, wizję systemu, historyjki użytkownika, diagram przypadków użycia, diagram klas, diagram encji oraz prototyp interfejsu. Każdy z etapów projektowania opierał się na wcześniejszym etapie opisującym dany aspekt. Przygotowana w ten sposób dokumentacja, dotycząca aplikacji tworzonej w ramach tej pracy, pozwala na przejście do implementacji, która dzięki przedstawionym w tym rozdziale diagramom, jest znacznie ułatwiona.

6.Opis aplikacji

Zdefiniowanie wymagań funkcjonalnych oraz diagramów opisujących architekturę systemu, pozwala na rozpoczęcie implementacji aplikacji. Tworzenie kodu opierającego się o wcześniej stworzony projekt, pozwala na uniknięcie dużej liczby błędów oraz stworzenie optymalnego rozwiązania. Spójna wizja jest bardzo pomocna w pisaniu kodu czytelnego oraz pozwalającego na przyszłe rozszerzenie funkcjonalności, dzięki wzorcom projektowym, zastosowanym w implementacji. By aplikacja mogła dobrze działać, należy przygotować środowisko wykonawcze, łatwe do zarządzania oraz działające w jak najmniejszej liczbie konfiguracji, potrzebnej do uruchomienia danego systemu. Bardzo ważnym punktem, z perspektywy użytkownika, jest nie tylko dobrze działająca aplikacja, ale interfejs, pozwalający na intuicyjne korzystanie z funkcjonalności. W tym celu, w projekcie aplikacji, został stworzony prototyp interfejsu, na podstawie którego, zaimplementowane zostały widoki w warstwie klienckiej. Ostatnim elementem opisanym w tym rozdziale są testy zdefiniowane i przeprowadzone w ramach sprawdzania działania systemu. Istnieją różne rodzaje testów, które są istotnym czynnikiem wpływającym na kontrolę nad błędami pojawiającymi się w trakcie działania. Testy jednostkowe z łatwością mogą pokazać miejsce, w którym zmiana mogła spowodować błąd, natomiast testy akceptacyjne, przeprowadzone w formie testów interfejsowych, sprawdzają główne funkcjonalności systemu. Rozdział ten opisuje wszystkie, wyżej wymienione fazy tworzenia aplikacji.

6.1. Środowisko wykonawcze

Biorąc pod uwagę, że aplikacja tworzona w ramach tej pracy, jest to aplikacja webowa, jej środowisko wykonawcze można podzielić na dwie części – środowisko wykonawcze po stronie klienta oraz serwera.

Środowiskiem wykonawczym po stronie klienta dla aplikacji webowej, jest dowolna przeglądarka internetowa, posiadająca silnik, uruchamiający kod pisany w Javascripcie. Najpopularniejszymi przeglądarkami są Mozilla, Chrome, Internet Explorer oraz Safari. Ważnym aspektem dobrze działającej aplikacji webowej, jest możliwość uruchomienia jej na każdej z tych przeglądarek. Różnice jakie można napotkać w interpretowaniu kodu przeznaczonego dla warstwy klienckiej, mogą generować problemy w napisaniu rozwiązania działającego na wszystkich dostępnych interpreterach. Jedną z zalet użycia frameworka Javascriptowego jest zapewnienie, że problem ten nie wystąpi dla tworzonej aplikacji. Angular, który został użyty przy implementacji, zapewnia spójny wygląd oraz prawidłowe działanie funkcjonalności na każdej z wyżej wymienionych przeglądarek.

Środowisko wykonawcze, którego wybranie jest zadaniem dla osoby tworzącej aplikację, jest to środowisko po stronie serwera. Biorąc pod uwagę silne połączenie serwerów z wybraną technologią, w jakiej napisana jest warstwa serwerowa, wybór narzędzia jest dokonywany wcześniej. Implementacja serwera używając platformy .NET, determinuje użycie serwera IIS w celu hostowania tworzonej aplikacji webowej.

IIS jest serwerem webowym stworzonym przez firmę Microsoft, działający na systemach Windows, akceptującym zdalne żądania klientów, wysyłając do nich odpowiednią treść odpowiedzi. Odpowiedzią na żądanie klienta może być statyczna strona HTML, tekst, obrazki oraz inne zasoby dostępne w internecie. Jednakże IIS dostarcza mechanizmy nie tylko generujące odpowiednie odpowiedzi, ale także zapewniające bezpieczeństwo oraz optymalne działanie serwera. Biorąc pod uwagę fakt, że każde żądanie jest obsłużone przez IIS, administrator jest w stanie sprawdzić tożsamość klienta oraz jego uprawnienia do zasobów, wymienionych w żądaniu. W celu ułatwienia zarządzania serwerem, wprowadzone zostały

pojęcia puli aplikacji oraz procesu wykonawczego. Proces wykonawczy odpowiada za wygenerowanie wszystkich żądań i odpowiedzi pochodzących z serwera, będąc głównym punktem działania aplikacji serwowanej przez IIS. Pula aplikacji jest to zbiór procesów wykonawczych, pozwalających na organizację oraz łatwiejsze przypisanie uprawnień dla danego systemu działającego w ramach IIS'a.

Ostatnim elementem, wchodzącym w skład środowiska wykonawczego, jest Docker. Docker jest to open-sourcowy projekt, stworzony w 2014 roku. Pozwala on na definicję środowiska uruchomieniowego dla aplikacji, jako przenośnych kontenerów, posiadających całą konfigurację zdefiniowaną wewnątrz. Kontenery stworzone za pomocą Dockera, uruchomione na różnych maszynach, działają w ten sam sposób. W aplikacji tworzonej w ramach tej pracy, Docker został użyty do uruchomienia serwera bazodanowego oraz stworzenia architektury Lambda za pomocą tego serwera oraz Sparka. Dzięki takiemu podejściu, możliwe jest użycie stworzonej architektury bazodanowej dla różnych aplikacji oraz na różnych maszynach, bez wymogu zbędnej konfiguracji.

6.2. Najważniejsze rozwiązane problemy

Tworząc aplikacje webową, deweloper napotyka się na wiele problemów związanych z implementacją. Globalnym problemem, bardzo ważnym z perspektywy czytelności oraz przyszłych modyfikacji, są zasady, jakimi deweloper kieruje się przy tworzeniu swojego kodu. Pisząc aplikację tworzoną w ramach tej pracy, kierowano się zasadami SOLID oraz CQRS.
SOLID, oznacza 5 zasad, określonych przez Roberta C. Martina, którymi osoba wytwarzająca oprogramowanie powinna się kierować. Brzmią one następująco [17]:

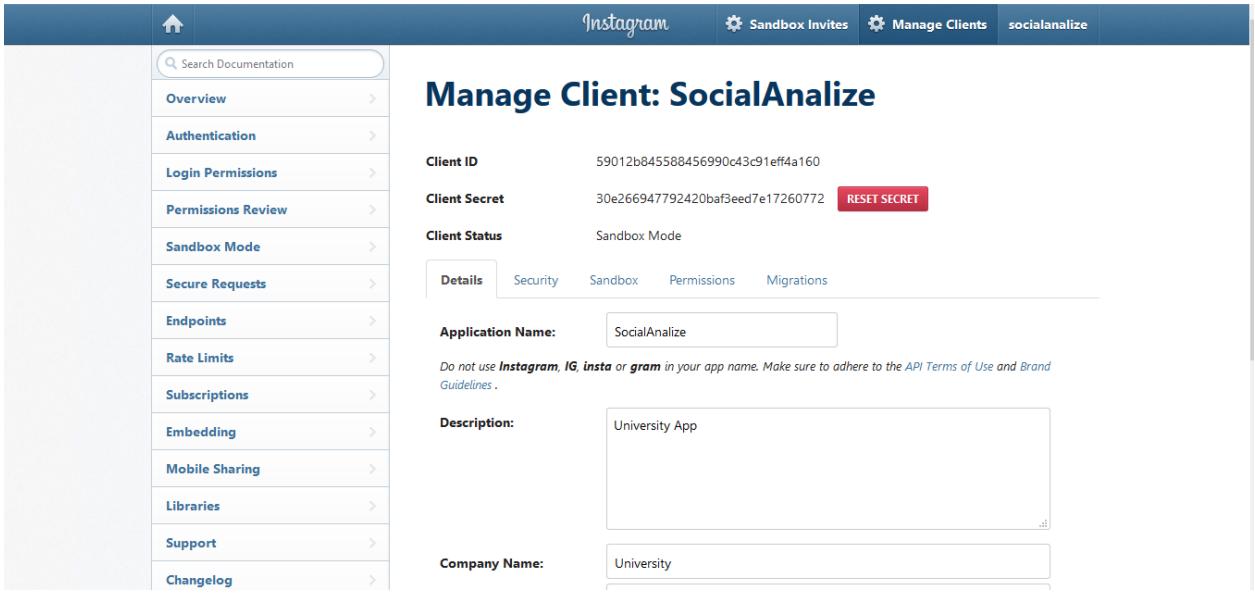
- Zasada jednej odpowiedzialności – klasa powinna zajmować się jedną rzeczą np. aktualizacją bazy danych. Prowadzi to do podziału złożonych funkcji na funkcje wykonującą atomowe, pojedyncze działania
- Zasada otwarte – zamknięte – klasy powinny być otwarte na rozszerzenia lecz zamknięte na modyfikację. Prowadzi do użycia interfejsów i klas abstrakcyjnych w jak największej liczbie przypadków
- Zasada podstawienia Liskova – obiekty klas powinny być zastąpione instancjami klas pochodnych, bez generowania błędów w aplikacji
- Zasada segregacji interfejsów – interfejsy powinny dotyczyć konkretnej dziedziny. Nie powinno się tworzyć interfejsów posiadających opis różnych dziedzin
- Zasada odwrócenia zależności – klasa powinna bazować na klasach abstrakcyjnych, interfejsach, nie na konkretnych implementacjach

Drugim użytym wzorcem, jest CQRS (Command Query Responsibility Segregation). Wzorzec ten pozwala na wykorzystanie innego modelu danych do pobierania danych oraz ich aktualizacji. Podstawowym podejściem używanym przy tworzeniu modelów, służących do pracy z danymi, jest podejście CRUD'owe. Polega ono na tworzeniu wspólnych modeli dla wszytkich operacji oraz za ich pomocą oraz pomocą repozytoriów, na komunikacji ze źródłami danych. Dla prostych scenariuszy jest to podejście prawidłowe, jednakże gdy wymagania względem aplikacji są większe, kod tworzony w ten sposób może stać się nieczytelny. W celu uniknięcia takiej sytuacji, stworzony został wzorzec CQRS, definiujący modele związane z zapytaniami oraz komendami. Modele związane z zapytaniami służą jedynie do odczytania danych z bazy, natomiast w celu aktualizacji lub dodania nowych danych używa się komend. W sytuacji gdy, po stronie serwera jest duża ilość logiki związanej ze źródłem danych, wzorzec CQRS bardzo dobrze sprawdza się w celu utworzenia czytelnego i zrozumiałego kodu.

Przy tworzeniu aplikacji w ramach tej pracy, pierwszym, konkretnym, problemem była autentykacja użytkownika, odbywająca się za pomocą jednego z trzech, analizowanych portali

społecznościowych. Dla każdego z nich, proces autentykacji wygląda w ten sam sposób i składa się z części rejestrującej aplikację na stronie deweloperskiej, związanej z portalem oraz części, w której aplikacja musi wykonać trzy żądania w celu uzyskania tokenu, pozwalającego wykonywać żądania do serwera.

Pierwszym etapem tworzenia mechanizmu autentykacji, za pomocą zewnętrznej aplikacji społecznościowej, jest utworzenie konta, a następnie klienta reprezentującego aplikacje na deweloperskiej stronie dla danego portalu społecznościowego.



Rys.6.8. Strona deweloperska Instagrama

Rysunek 6.1 przedstawia klienta, utworzonego w ramach tej pracy. Posiada on dane, takie ja client id oraz client secret, będące kluczami, potrzebnymi w później wysłanych żądaniach, mających na celu uzyskania tokenu dostępu dla użytkownika. Następnym elementem, jaki należy zdefiniować, jest adres powrotny do budowane w ramach niniejszej pracy aplikacji, po potwierdzeniu uprawnień przez użytkownika.

Posiadając informacje ze strony deweloperskiej, można przejść do implementacji autentykacji w budowanym systemie. Najpierw jednak należy zdefiniować słuchacza zdarzeń po stronie klienta, wywołującego akcje autentykacji po stronie serwera. Zawartość wywołanej funkcji przedstawiono na Listingu 6.1:

```
loginInstagramUser() {
       window.location.href                                                    =
       'http://localhost:50985/api/instagramAuthentication/authenticate';
}
```

Listing.6.9.Funkcja logująca użytkownika

Warstwą serwerową aplikacji, tworzonej w ramach tej pracy, jest to projekt Web API, więc wywołana funkcja przekierowuje użytkownika na konkretną akcję kontrolera po stronie serwera. Listing 6.2 pokazuje akcje kontrolera po stronie serwera, obsługującą autentykację.

```
[HttpGet]
[Route("authenticate")]
public ActionResult Get()
{
	var clientId = "59012b845588456990c43c91eff4a160";
	var clientSecretId = "30e266947792420baf3eed7e17260772";
	var redirectUri = @"http://localhost:50985/api/instagramAuthentication";

	var config = new InstagramConfig(clientId, clientSecretId, redirectUri, realtimeUri);

	var scopes = new List<OAuth.Scope>();
	scopes.Add(OAuth.Scope.Likes);
	scopes.Add(OAuth.Scope.Comments);
	scopes.Add(OAuth.Scope.Basic);
	scopes.Add(OAuth.Scope.Follower_List);
	scopes.Add(OAuth.Scope.Relationships);
	scopes.Add(OAuth.Scope.Public_Content);

	var link = OAuth.AuthLink(config.OAuthUri + "authorize", config.ClientId,
config.RedirectUri, scopes, InstaSharp.OAuth.ResponseType.Code);
	return Redirect(link);
}

[HttpGet]
public ActionResult Get(string code)
{
	var accessToken =  queryBus.Send<GetInstagramAccessToken, string>(new
GetInstagramAccessToken() {
		ClientId = "59012b845588456990c43c91eff4a160",
		ClientSecretId = "30e266947792420baf3eed7e17260772",
		Code = code,
		ReturnUrl = @"http://localhost:50985/api/instagramAuthentication"
	}).Result;

return Redirect("http://localhost:50985?accessToken=" + accessToken);
}
```

Listing.6.2.Funkcja obsługująca autentykację po stronie serwera

Pierwsza funkcja odpowiada za stworzenie przekierowania na poprawną stronę logowania portalu społecznościowego. Url stworzony w ramach tej funkcji posiada, klucze pobrane ze strony deweloperskiej oraz zdefiniowany tam adres powrotu. Druga funkcja jest wywołana przez portal społecznościowy za pomocą adresu powrotu, dodając do niego parametr zawierający kod, który służy do wysłania żądania HTTP w celu uzyskania tokenu. W ramach wzorca CQRS, wywoływany jest handler, wykonujący żądanie po token dostępu, a następnie przekierowuje użytkownika na stronę aplikacji frontowej, z tokenem jako parametrem adresu.

Przedstawiony na Listing 6.3 kod, tworzący parametry zapytania HTTP typu POST w celu uzyskania tokenu dostępu. Dzięki temu kluczowi, aplikacja w ramach tego użytkownika, jest w stanie pobrać dane z serwerów danego portalu społecznościowego.

```
private async Task<string> GetAccessToken(GetInstagramAccessToken authData)
{
       var client = new HttpClient();
       var postValues = new List<KeyValuePair<string, string>>
              new KeyValuePair<string, string>("client_id",authData.ClientId),
              new KeyValuePair<string, string>("client_secret",authData.ClientSecretId),
              new KeyValuePair<string, string>("grant_type","authorization_code"),
              new KeyValuePair<string, string>("redirect_uri",authData.ReturnUrl),
              new KeyValuePair<string, string>("code", authData.Code)
       };

       // now encode the values
       var content = new FormUrlEncodedContent(postValues);
       var authLinkUri = new Uri(@"https://api.instagram.com/oauth/access_token");
       // make request for auth token
       var response = await client.PostAsync(authLinkUri, content);

       var parsedResponse = await response.Content.ReadAsStringAsync();
       var json = JObject.Parse(parsedResponse);
       var accessToken = json["access_token"].ToString();
       return accessToken;
}
```

Listing.6.3.Kod uzyskujący Access Token dla użytkownika

Głównym problemem, w ramach aplikacji tworzonej w tej pracy, było zaimplementowanie mechanizmu Business Inteligence - Architektury Lambda. Biorąc pod uwagę to, że warstwą Speed tej architektury są to widoki tworzone w PostgreSQL, w tym rozdziale poniżej (Listing 6.4) przedstawiony jest prosty przykład kodu w języku Scala, wykonywany przez Sparka, w celu przetworzenia danych. Mechanizm ten, w zaimplementowanej architekturze, służy jako warstwa Batch, która zajmuje się przetworzeniem danych oraz utworzeniem gotowych do odczytania raportów na temat osoby obserwowanej.

```
def transformLocations(spark: SparkSession, df: DataFrame): DataFrame = {
    import spark.implicits._

    val jsons = df
      .map(t => t.getString(2))

    val locations = spark.read.json(jsons)
      .select(explode($"data").alias("data"))
      .select(
        monotonically_increasing_id().as("Id"),
        $"data.user.id".cast("long").as("UserId"),
        $"data.user.username".as("UserName"),
        from_unixtime($"data.created_time").cast("timestamp").as("Created"),
        $"data.location.latitude".cast("decimal(9,6)").as("Latitude"),
        $"data.location.longitude".cast("decimal(9,6)").as("Longitude")
      )

    locations.show()

    return locations
}
```

Listing.6.4.Kod w Scali przetwarzający dane

Kod definiuje prostą transformację danych lokalizacyjnych, za pomocą Scali, wykonywaną przez Sparka. Wyciągnięte dane są następnie zapisane do tabeli, którą aplikacja wykorzystuje do budowy raportu związanego z lokalizacją. Kod pobierający surowe dane, wywołujący metodę „transformLocations” oraz zapisujący wynik tej metody do bazy jest pokazany na Listingu 6.5:

```
def processLocations(spark: SparkSession) {
    val df = Postgres.read(spark, "import.instagram_media_recent")

    val locations = transformLocations(spark, df);

    Postgres.write(locations, "data.UserLocation")
}
```

Listing.6.5.Kod zapisujący przetworzone dane do bazy danych

Patrząc z perspektywy problemów występujących przy implementacji, można je podzielić na problemy globalne, związane z czytelnością oraz dobrym poukładaniem kodu oraz na konkretne problemy połączone z funkcjonalnościami. Na pierwszy typ problemów, rozwiązaniem jest podążanie za dobrymi praktykami kodowania, natomiast drugi typ problemów, często wymaga zaimplementowania specyficznego mechanizmu, pozwalającego na optymalne spełnienie danego wymagania funkcjonalnego.

6.3. Testy jednostkowe, akceptacyjne oraz środowiskowe

Dobrze działająca aplikacja, pozwalająca na łatwe rozszerzenie funkcjonalności, musi być aplikacją dobrze przetestowaną. Istnieje wiele rodzajów testów, sprawdzających różne etapy wytwarzania aplikacji. Najczęściej spotykanymi typami testów, są testy jednostkowe. Dobrze napisany test jednostkowy, odpowiada za sprawdzenie tylko jednej funkcji. Jeżeli testuje zbyt obszerną część funkcjonalności, powoduje to duże trudności w utrzymaniu oraz słabą czytelność jego wyników. Drugim typem testów, wykonanych w ramach tej pracy, są to testy akceptacyjne. Sprawdzają one działanie aplikacji w perspektywie wymagań funkcjonalnych, nie wchodząc w detale techniczne. W przypadku tego systemu, testami akceptacyjnymi będą to testy „end-to-end", czyli testy Selenium.

Selenium jest to framework, pozwalający na komunikację z przeglądarką. Dzięki niemu, programista jest w stanie automatycznie sterować tym, co dzieje się na stronie internetowej. Istnieją dwa sposoby tworzenia testów Selenium – poprzez nagrywanie oraz poprzez pisanie kodu. Nagrywanie testu, polega na „wyklikaniu" tego, co dany test ma robić. W tym sposobie framework Selenium automatycznie generuje kod testu. Drugim sposobem, jest manualne napisanie testu przez dewelopera.

Ostatnim typem testów, przeprowadzanych w końcowym okresie wytwarzania aplikacji, są to testy środowiskowe. Polegają one na udostępnieniu systemu kilku klientom, w celu sprawdzenia przez nich funkcjonalności oraz poprawności stworzonego interfejsu. W aplikacji tworzonej w ramach tej pracy, do definicji testów, zostaną użyte narzędzia związane z Angularem – do testów jednostkowych Jasmina wraz z Karmą oraz do testów akceptacyjnych Protractor.

Testy jednostkowe zostaną utworzone po stronie klienta, za pomocą frameworka javascriptowego o nazwie Jasmine wraz ze środowiskiem testowym o nazwie Karma. Za pomocą tych narzędzi, deweloper jest w stanie zdefiniować zestaw testów, sprawdzających działanie całej warstwy klienta, a następnie za pomocą Karmy przedstawić raport, pokazujący skuteczność za pomocą wskaźnika pokrycia kodu. W kodach źródłowych(Listingi 6.6 – 6.8) zostaną przedstawione przykładowe testy napisane za pomocą Jasmine.

```
beforeEach(async(() => {
    TestBed.configureTestingModule({
      providers: [
          AuthenticationService,
          {
              provide: window,
              useClass: MockWindow
          }
      ]
    })
    .compileComponents();
}));
```

Listing.6.6.Funkcja wywoływana przed uruchomienie testu jednostkowego

Pierwszym krokiem w napisaniu testów, jest zdefiniowanie operacji, które zostaną wykonane przed każdym z nich, przygotowując zamienniki dla klas, odwołujących się do zewnętrznych serwisów i stron. W tym przypadku, został stworzony zamiennik dla zmiennej globalnej window. Biorąc pod uwagę, że w serwisie używamy tyko zmiennej href, zamiennik przedstawiony jest na Listingu 6.7:

```
export class MockWindow {
    public location: {
        href: string;
    };
}
```

Listing.6.7.Zamiennik dla globalnego obiektu window

W tym momencie, deweloper jest przygotowany do zdefiniowania testów, sprawdzających serwis logowania (Listing 6.8).

```
it('should create', () => {
    expect(service).toBeTruthy();
});

it('should redirect to proper page when instagram login', () => {
    service.loginInstagramUser();
    expect(window.location.href)
    .toEqual('http://localhost:50985/api/instagramAuthentication/authenticate');
});
```

Listing.6.8.Testy jednostkowe sprawdzające autentykacje

Powyżej zostały przedstawione dwa przykładowe testy, sprawdzające serwis logowania. Pierwszy sprawdza, czy dany serwis został stworzony, natomiast drugi czy po wywołaniu akcji logowania do Instagrama, użytkownik został przekierowany na odpowiednią, zewnętrzną stronę.

Następnym typem testów są testy Selenium. Zostaną one zaimplementowane za pomocą frameworka angularowego Protractor i będą sprawdzały większą część funkcjonalności. Poniżej pokazany jest ich przykład (Listingi 6.9 – 6.10).

Listing 6.9 przedstawia kod, za pomocą którego test Selenium jest stworzony. Służy on do sterowania działaniem przeglądarki, odpowiednio nawigując oraz pobierając elementy dostępne na stronie.

```
navigateToDashboard() {
  return browser.get('/dashboard');
}

getDashboardRootContainer() {
  return element(by.className('dashboard-root'));
}

getDashboardDataBtn() {
  return element(by.className('refresh-btn'));
}

getDashboardUserData() {
  return element(by.className('user-data'));
}
getDashboardNavigationBtn() {
  return element(by.className('nav-btn'));
}
```

Listing.6.9.Funkcje sterujące zachowaniem przeglądarki

Przykładowy test został przedstawiony na listingu 6.10. Jak można zauważyć, wykorzystuje wcześniej zdefiniowane funkcje w obiekcie page. Za ich pomocą, na początku przekierowuje przeglądarkę na stronę dashboardu, następnie wyciągając po kolei znaczące elementy, sprawdza czy pojawiły się one na stronie.

```
it('should redirect to dashboard and display proper page', () => {
  page.navigateToDashboard();
  expect(page.getDashboardRootContainer()).toBeTruthy();
  expect(page.getDashboardUserData()).toBeTruthy();
  expect(page.getDashboardDataBtn()).toBeTruthy();
  expect(page.getDashboardNavigationBtn()).toBeTruthy();
});
```

Listing.6.10.Przykładowy test selenium

Ostatnimi testami przeprowadzonymi w ramach tej pracy, były to testy środowiskowe. Polegały one na udostępnieniu aplikacji potencjalnym klientom, w celu przetestowania działania oraz poprawności interfejsu. Pierwszą rzeczą, jaka została zauważona przy przeprowadzeniu tych testów, był problem z mechanizmem odświeżenia danych. Jak można było się dowiedzieć, mechanizm ten jest mało intuicyjny i niewygodny w użyciu. Jednakże biorąc pod uwagę, trudności jakie występują podczas pobierania danych, bez potrzeby ciągłego zalogowania użytkownika, niestety mechanizm ten jest niezbędny do działania aplikacji. Drugim często zadawanym pytaniem było pytanie o rodzaj informacji, jakie aplikacja stworzona w ramach tej pracy otrzymuje na temat zalogowanego użytkownika. Często zewnętrzne serwisy, po przekierowaniu na ich zewnętrzną stronę, wyświetlają informacje do jakich danych aplikacja będzie miała dostęp. Było to niepokojące dla użytkowników korzystających z systemu. Jednakże biorąc pod uwagę, że ta wyświetlana informacja jest sterowana przez zewnętrzny system, nie ma możliwości zmiany jej treści. Problem ten został wliczony w koszty używania systemu.

6.4. Przykład działania

W ramach niniejszej pracy, została utworzona aplikacja, udostępniająca raporty na temat użytkowników korzystających z portali społecznościowych. Portalami, które zostały poddane analizie były Facebook, Twitter i Instagram. W tym rozdziale, pokazane jest działanie oraz wygląd aplikacji końcowej. Przykłady widoków, dotyczą głównie raportów na temat obserwowanych osób.

Pierwszym widokiem (Rys.6.2) jest widok startowy aplikacji, dostępny dla niezalogowanego klienta. Udostępnia on trzy możliwości za pomocą których, użytkownik może dostać się do aplikacji. Po wybraniu jednej z nich, użytkownik zostaje przekierowany na zewnętrzną stronę potwierdzającą jego tożsamość, a następnie z powrotem do aplikacji.

Rys.6.10. Strona startowa aplikacji

Rysunek 6.3 przedstawia widok strony nawigacyjnej w ramach obserwowanej osoby. Jest to bardzo ważna strona, ponieważ są tutaj przyciski, za pomocą których następuje odświeżanie danych, czyli pobranie ich z portali społecznościowych i przetworzenie przez Architekturę Lambda.

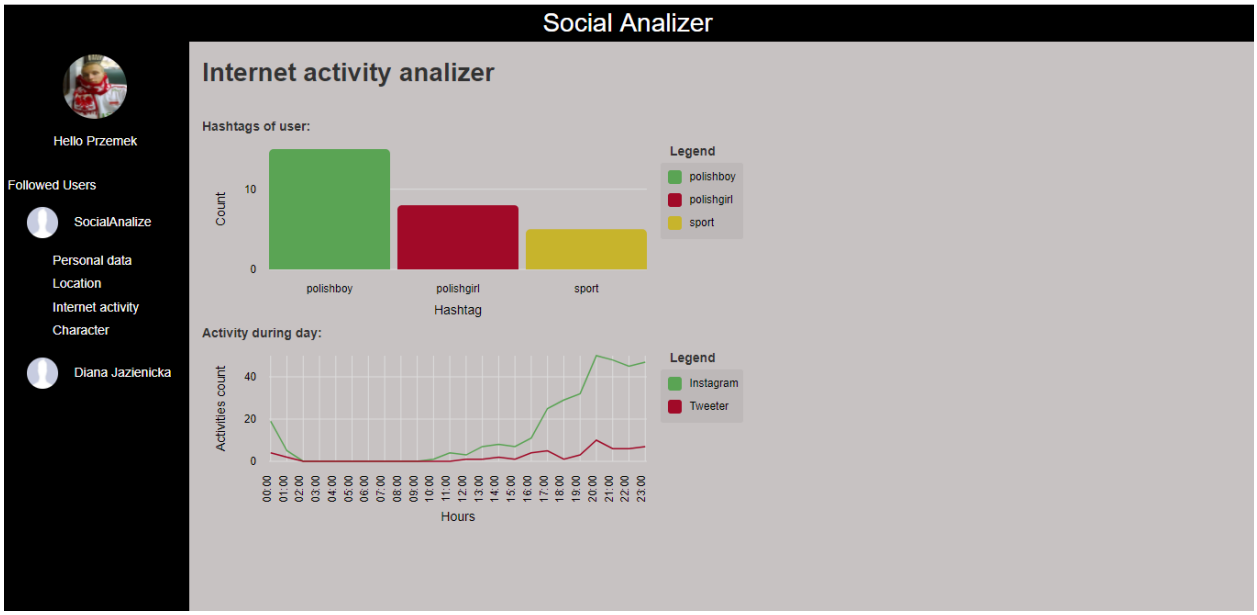
Rys.6.11. Strona nawigacyjna osoby obserwowanej

Na rysunku 6.4 przedstawiony został raport dotyczący danych lokalizacyjnych. Na początku raportu podane są podstawowe dane takie jak imię, nazwisko oraz miasta z których użytkownik pochodzi oraz w którym aktualnie przebywa. Poniżej pokazana jest mapa, na której markerami

zaznaczone są miejsca, w których użytkownik przebywał. Po naciśnięciu na marker, pojawiają się szczegóły dotyczące lokalizacji.

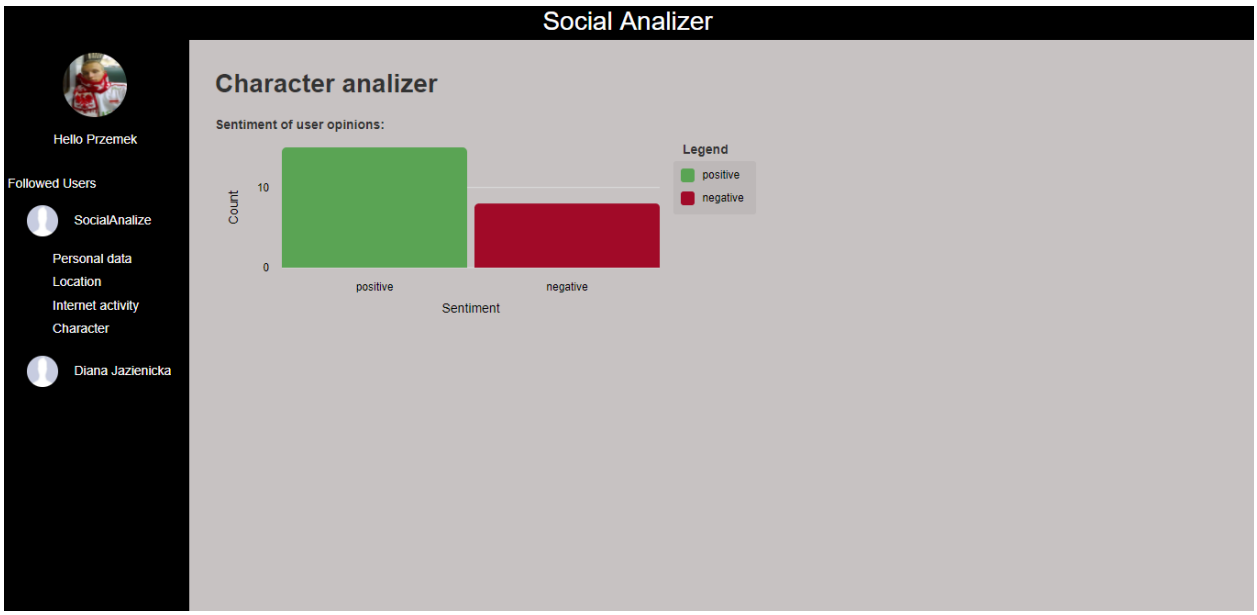
Rys.6.12. Strona raportu opartego o dane lokalizacyjne

Następna strona (Rys.6.5) przedstawia informacje zebrane przez aplikacje Social Analizer dotyczące aktywności internetowej. Przedstawia dwa wykresy – wykres hashtagów oraz wykres aktywności w trakcie dnia. Wykres hashtagów, przedstawia tematy oraz słowa kluczowe z nimi związane, jakimi użytkownik najbardziej się interesuje oraz którymi posługuje się na portalach społecznościowych. Natomiast wykres aktywności w trakcie dnia, pokazuje liniowy wykres, wskazujący na godziny w których użytkownik jest najbardziej aktywny.



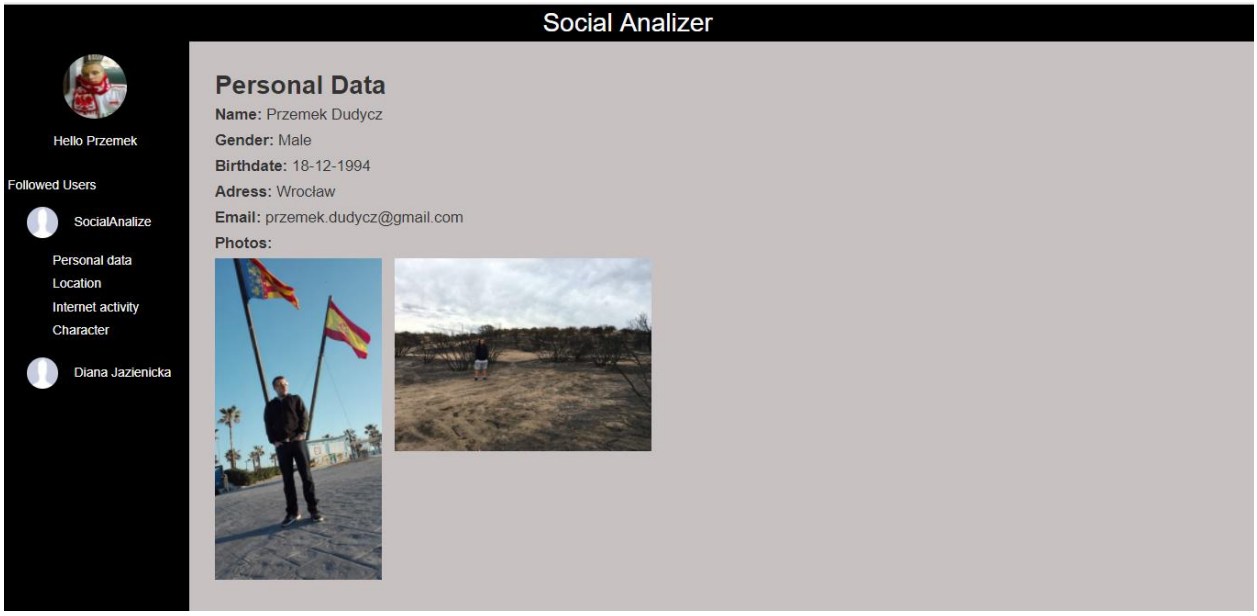
Rys.6.13. Strona raportu opartego o dane aktywności internetowej

Raport dotyczący danych związanych z charakterem, jest przedstawiony na rysunku 6.6. Widok ten, przedstawia wykres słupkowy, ukazujący stosunek treści pozytywnych do treści negatywnych udostępnianych przez użytkownika.



Rys.6.14. Strona raportu opartego o dane związane z charakterem

Ostatnim raportem, dostępnym w tworzonej w ramach tej pracy aplikacji, jest raport danych personalnych. Raport ten przedstawia podstawowe informacje na temat osoby, takie jak imię i nazwisko, płeć czy data urodzenia, oraz zebrane zdjęcia powiązane z profilami danego użytkownika.



Rys.6.15. Strona raportu opartego o dane personalne

6.5. Podsumowanie

Zbudowanie dobrze działającej aplikacji jest zadaniem bardzo trudnym. Biorąc pod uwagę liczbę występujących problemów, deweloper potrzebuje kierować się odpowiednimi zasadami oraz wzorcami, w celu napisania dobrze poukładanego kodu, pozwalającego na modyfikacje i odczytanie go w przyszłości. W tym celu zostały wykorzystane odpowiednie wzorce projektowe takie jak zasady SOLID czy CQRS. Korzystając z tych wzorców, rozwiązano problemy dotyczące autentykacji, zapisu i odczytu danych z bazy oraz komunikacji przeprowadzonej z warstwą kliencką. Bardzo ważną cechą korzystania z wzorców jest łatwość, z

jaką można napisać testy do konkretnej funkcjonalności. W ramach tego rozdziału, opisane zostały podstawowe testy z trzech zdefiniowanych typów – testów jednostkowych, akceptacyjnych i środowiskowych. Testy jednostkowe i akceptacyjne posłużyły do automatycznego sprawdzenia poprawności działania systemu, natomiast testy środowiskowe posłużyły do otrzymania informacji zwrotnej od potencjalnych klientów, na temat funkcjonalności oraz poprawności interfejsu.

7.Podsumowanie

Portale społecznościowe, są to aplikacje ułatwiające komunikację ze znajomymi oraz znanymi użytkownikowi grupami osób. Dane zawarte na takich portalach, dotyczą różnych dziedzin życia oraz opisują wydarzenia jakie miały miejsce czy opinie na dany temat. Dzięki tym systemom, można wyrobić sobie zdanie o konkretnej osobie, nigdy jej nie spotykając, co jest ważne przy różnych sytuacjach życiowych, przykładowo próbie otrzymania pracy. Złe treści udostępniane na publicznych profilach, mogą realnie wpłynąć na przyszłość użytkownika. Aplikacja stworzona w ramach tej pracy, pokazuje jak szczegółowe dane, może wydobyć osoba, łącząc informacje dostępne z wielu witryn społecznościowych. Przykładowy raport dotyczący lokalizacji, pokazuje szczegółowość zaimplementowanej analizy oraz zagrożenia płynące z nierozsądnego udostępniania w internecie treści, związanych z życiem danej osoby.

Biorąc pod uwagę fakt, że dane zawarte na serwerach Facebooka, Instagrama czy Twittera są często danymi poufnymi, systemy te posiadają wiele mechanizmów je zabezpieczających. Pierwszym dostępnym mechanizmem jest profil prywatny, który użytkownik może włączyć wchodząc w ustawienia. Oznaczenie swojego profilu jako prywatny powoduje, że jedynie określona grupa osób może oglądać dane związane z użytkownikiem. Dane z takich profili, są niedostępne w aplikacjach, ale także w wystawionych przez portale społecznościowe serwisach API, więc analiza takich osób jest nieskuteczna. Jednym z wniosków, pojawiających się po analizie utworzonych raportów, jest potwierdzenie używania profilu prywatnego, dostępnego tylko dla znanych użytkowników. Pozwala to osiągnąć odpowiedni stopień bezpieczeństwa i anonimowości. Drugim mechanizmem zabezpieczającym przed analizą poufnych danych, są pozwolenia dla zewnętrznych aplikacji do pobierania odpowiednich danych. Najbardziej restrykcyjnym portalem społecznościowych z trzech analizowanych w ramach tej pracy, jest Facebook, który udostępnia jedynie podstawowe dane na temat użytkownika.

Pomimo wyżej opisanych mechanizmów bezpieczeństwa, ilość danych, które są pobierane, w szczególności dotyczących znanych osób, powoduje nieoptymalne działanie prostych serwerów bazodanowych. Zaproponowana w ramach tej pracy struktura, pochodząca z dziedziny Business Intelligence, Architektura Lambda jest rozwiązaniem pozwalającym analizować te duże ilości danych oraz przedstawiać aktualne informacje bez oczekiwania. Rozwiązanie to jest rozwiązaniem ogólnym i może zostać zaimplementowane na wiele sposobów. W ramach tej pracy został użyty Spark oraz widoki postgresowe jako dwie najważniejsze warstwy tej struktury. W przypadku aplikacji tworzonej w ramach tej pracy, zaimplementowana architektura zadziałała prawidłowo, dostarczając przeliczone przez Sparka raporty oraz aktualne dane, dostarczone przez te widoki. Najważniejszym rozwiązanym problemem podczas tworzenia tej architektury, było zdefiniowanie przeliczeń w języku Scala, w ramach Sparka.

Cele postawione na początku tej pracy, zostały w całości osiągnięte. W pierwszej kolejności, został dokonany przegląd najpopularniejszych konkurencyjnych rozwiązań internetowych, analizujących dane z portali społecznościowych. Można zauważyć, że większość aplikacji, przedstawia raporty bazujące na informacjach z witryn społecznościowych, w celu analizy rynku, biznesu oraz opinii klientów na temat danego produktu. Jedynie jedna aplikacja stara się przedstawiać informacje na temat konkretnego użytkownika, jednakże cena tych raportów nie pozwoliła na bardziej szczegółową analizę tego rozwiązania. Następnym celem tej pracy był wybór technologii, za pomocą których zostanie stworzona aplikacja. Wybór ten dotyczył trzech warstw aplikacji:

- Warstwy klienckiej
- Warstwy serwerowej
- Warstwy bazodanowej

Po analizie cech i wad różnych podejść, dla warstwy klienckiej wybrany został Angular, w warstwie serwerowej użyty został .NET Core, natomiast wartwa bazodanowa została stworzona za pomocą PostgreSQL'a oraz Sparka. W ostatniej warstwie, zaimplementowana została Architekturą Lambda. Architektura ta ma za zadanie przetwarzać dane, tworząc skomplikowane raporty, równocześnie natychmiastowo udostępniając aktualne dane. Pozwala ona także na bardzo skuteczne skalowanie rozwiązania, poprzez możliwość dodania nowych widoków w odpowiednich warstwach. Dzięki temu tworzona aplikacja może być w łatwy sposób rozszerzona o nowe raporty. Następnym zadaniem wykonanym w ramach tej pracy, było stworzenie projektu systemu, który składał się z opisu rzeczywistości, wizji systemu, historyjek użytkowników, diagramu przypadków użycia, diagramu klas, diagramu encji oraz prototypu interfejsu. Projekt tak zdefiniowany, pozwolił na wyeliminowanie dużej liczby błędów, występujących w trakcie implementacji oraz stworzenie dokumentacji, która pokazuje wszystkie wymagania funkcjonalne oraz niefunkcjonalne dla aplikacji. W ostatnim etapie, została dokonana implementacja systemu na podstawie wcześniej stworzonej dokumentacji.

W przyszłości, przewiduje się dodanie większej liczby raportów ukazujących bardziej szczegółowy obraz obserwowanej osoby. Serwery, związane z portalami społecznościowymi udostępniają większą ilość danych aplikacjom, które otrzymały pozytywną ocenę oraz dostały subskrypcję tych portali. Aplikacja stworzona w ramach tej pracy, zostanie przesłana do oceny oraz po jej pozytywnym wyniku zostaną zdefiniowane raporty, zawierające bardziej szczegółowe dane. Następnym krokiem, będzie próba udostępnienia aplikacji w internecie dla wszystkich zainteresowanych użytkowników.