Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Systemy Informacyjno-Decyzyjne

Narzędzie wspomagające profile biznesowe w sieci społecznościowej Instagram

Zofia Maria Dąbrowska
Numer albumu 276995

promotor
dr inż. Mariusz Kamola

WARSZAWA 2020

Narzędzie wspomagające profile biznesowe w sieci społecznościowej Instagram

Streszczenie. Celem pracy było dostarczenie aplikacji webowej umożliwiającej użytkownikom prywatnym oraz agencjom reklamowym przeskanowanie profili biznesowych w sieci społecznościowej Instagram. Od momentu rejestracji profilu w bazie, aplikacja codziennie dokonuje skanowania konta i zbiera informacje w bazie. Aplikacja w ramach analizy przedstawia średnią liczbę polubień oraz komentarzy pod postem oraz wskaźnik zaangażowania obserwatorów. Ponadto na podstawie historii ruchu na profilu, tworzy wykresy liniowe zawierające liczbę obserwatorów oraz obserwujących. Dodatkowo przy użyciu wykresów słupkowych przedstawione jest zainteresowanie postami danego profilu. Aplikacja zawiera również dedykowane agencjom reklamowym narzędzie przedstawiające wybrane wskaźniki profilu za pomocą wykresu z układem biegunowym. Część serwerowa aplikacji została napisana przy użyciu języka Java, interfejs graficzny powstał przy użyciu języka JavaScript z platformą programistyczną React.js. Pobieranie danych z profili zostało zrealizowane przy użyciu oficjalnego Facebook Graph API, które udostępnia informacje o kontach na Instagramie. Realizacja pracy została przeanalizowana zarówno od strony technicznej, jak i przy wykorzystaniu testów. Przeprowadzono również symulację wyboru profili do kampanii reklamowej.

Słowa kluczowe: aplikacja webowa, Instagram, Facebook Graph API, Java, JavaScript, React.js

Audit tool supporting business profiles on a social media platform Instagram

Abstract. The main purpose of this thesis was to develop a web application for private and commercial users, enabling an audit of every business profile on a social media platform called Instagram. Since the very first profile registration, the application will be scanning this profile everyday and collecting all data to the database. Application with its analysis will be creating an average amount of likes and an average amount of comments per post and profile's engagement rate. Moreover, it prepares line charts for followers and following based on profile history. It also creates a bar chart displaying engagement under each user's post. Application has a tool, dedicated for advertising agencies, displaying profile variables represented by a radar chart. An application server was implemented with Java programming language, user interface was implemented with JavaScript programming language and React.js framework. Getting profiles data was performanced by using the official Facebook Graph API, which shares information about Instagram profiles. Thesis development has been analyzed technically and by using tests. Furthermore, there has been a simulation of choosing a profile for an advertising campaign.

Keywords: web application, Instagram, Facebook Graph API, Java, JavaScript, React.js

załącznik nr 3 do zarządzenia
nr 28 /2016 Rektora PW

Politechnika Warszawska

……..
miejscowość i data

……..
imię i nazwisko studenta

……..
numer albumu

…….…….
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta

Spis treści

1. Wstęp

1.1. Reklama

W dzisiejszych czasach nieodzownym elementem sprzedaży każdego produktu jest reklama. To ona wpływa na jego rozwój i popyt. Kształtuje go i identyfikuje. Za pomocą reklamy opowiada się historię produktu i tworzy wokół niego aurę, która przyciąga odbiorców. Pozwala sięgać do ich najgłębszych pragnień, a nawet jeśli takich jeszcze nie mają, dzięki odpowiedniemu przedstawieniu produktu, kreuje ich marzenia. Jak wyglądałaby sprzedaż bez reklamy? To można sobie tylko wyobrażać, lecz jeśli popatrzy się z perspektywy ostatnich stu lat na rozwój dziedziny marketingu i reklamy, można śmiało powiedzieć, że jej prędkość i wielowymiarowość dorównuje rozwojowi technologii.

Reklama powoli raczkowała już w połowie XIX wieku, jednak to XX wiek przysporzył jej zawrotnego tempa. Zaczynała od ogłoszeń w gazetach, następnie pojawiła się w audycjach radiowych by zaraz później wyświetlać się na ekranach telewizorów. Dzisiaj otacza nas na każdym kroku, w każdym miejscu gdzie się znajdujemy, gdzie żyjemy i gdzie obcujemy. Reklama od zawsze szła ramię w ramię z rozwojem technologii i zawsze znajdowała się w przedmiotach codziennego użytku. Nie powinien więc dziwić fakt, że w związku z tak dużym zainteresowaniem mediami społecznościowymi na przestrzeni ostatnich dziesięciu lat, to tam właśnie reklama znajdzie swoje miejsce.

Według badań domu mediowego Starcom dla pierwszego półrocza 2019 udział Internetu w rynku reklamowym wynosił aż 35,8%, zaś telewizji 47,1% [1]. Świadczy to o praktycznie całkowitym przeniesieniu reklamy do urządzeń cyfrowych. Ponadto stwarza również niemałe wyzwanie dla branży marketingu ze względu na wielowymiarowość i występowanie reklam w tak wielu miejscach.

Niewątpliwie próba przeanalizowania wszystkich dostępnych klas mediów mogłaby być niemożliwa, a w samym Internecie znajduje się nieskończona liczba miejsc poprzez, które reklamodawcy mogą trafiać do klientów. Przykładem są serwisy społecznościowe wśród których można wymienić trzy najpopularniejsze: Facebooka, Youtuba oraz Instagrama. To właśnie te domeny zostały zaklasyfikowane do rankingu TOP20 domen, z których korzystało najwięcej internautów, na wszystkich urządzeniach w Polsce na sierpień 2019 rok, według firmy Gemius SA[2]. W związku z tym niniejsza praca skupiać się będzie na klasie mediów Internetowych w specjalizacji sieci społecznościowych Instagram.

1.2. Motywacja i cel pracy

Rynek reklamowy w sieci społecznościowej Instagram opiera się głównie na współpracy z tzw. Influencerami, z języka angielskiego *influencer* czyli osoba, która posiada wpływ. Są to osoby posiadające szerokie grono obserwatorów, którzy śledzą ich działania i wydarzenia z życia. Osoby popularne w Internecie, dzięki swoim fanom mogą wpływać na popyt danych produktów i to co obecnie jest w modzie. Liczby obserwatorów sięgają

milionów. Przykładem jest konto znanego piłkarza Roberta Lewandowskiego, którego grono odbiorców wynosi aż 18 milionów [3], a liczba ta wcale nie jest rekordowa. Dla porównania ludność Polski z czerwca 2019 według GUS to 38 386 000 osób [4]. Gdyby Robert Lewandowski miał tylko obserwatorów z Polski oznaczałoby to, że obserwuje go 46,9% Polaków, a to prawie co drugi statystyczny Polak. Takie porównanie pozwala przybliżyć skalę zasięgów jakie może osiągnąć pojedyncze zdjęcie dodane przez Lewandowskiego.

Posiadanie tak wielu obserwatorów może dać ogromne możliwości, jednak stoją za tym również problemy zarówno dla influencera jak i dla agencji reklamowych. Obie strony chcąc uzyskać jak najlepsze wyniki muszą określić wartość odbiorców. Niestety wśród wielu popularnych osób znajdują się oszuści, których ruch na profilach jest sztucznie generowany przez boty. Zdarzają się też sytuacje, w których sprawiedliwie prowadzone konta padają ofiarą konkurencji i w ten sposób również pozyskują sztucznych obserwatorów. Dla agencji jest to ogromny problem, ponieważ reprezentując swoich klientów, chcą zapewnić im jak najlepsze efekty i sprzedaż. Współpraca z nieuczciwą osobą może doprowadzić nie tylko do zerwania umowy z klientem, ale także do naruszenia wizerunku firmy.

Domy mediowe zajmujące się rynkiem sieci społecznościowych codziennie mierzą się z problemem jakim jest dobór odpowiednich osób do przeprowadzenia kampanii reklamowych. Nasuwa się więc pytanie w jaki sposób domy mediowe oraz influencerzy mogą ocenić jakość przeprowadzanych kampanii i ich wyników? Rozwiązaniem tego problemu jest skorzystanie z narzędzi audytorskich, które skanują instagramowe profile i śledzą ruch na nich.

Celem niniejszej pracy było stworzenie narzędzia, które wspomagałoby prace profili biznesowych oraz agencji kreatywnych w sieci społecznościowej Instagram.

1.3. Istniejące rozwiązania

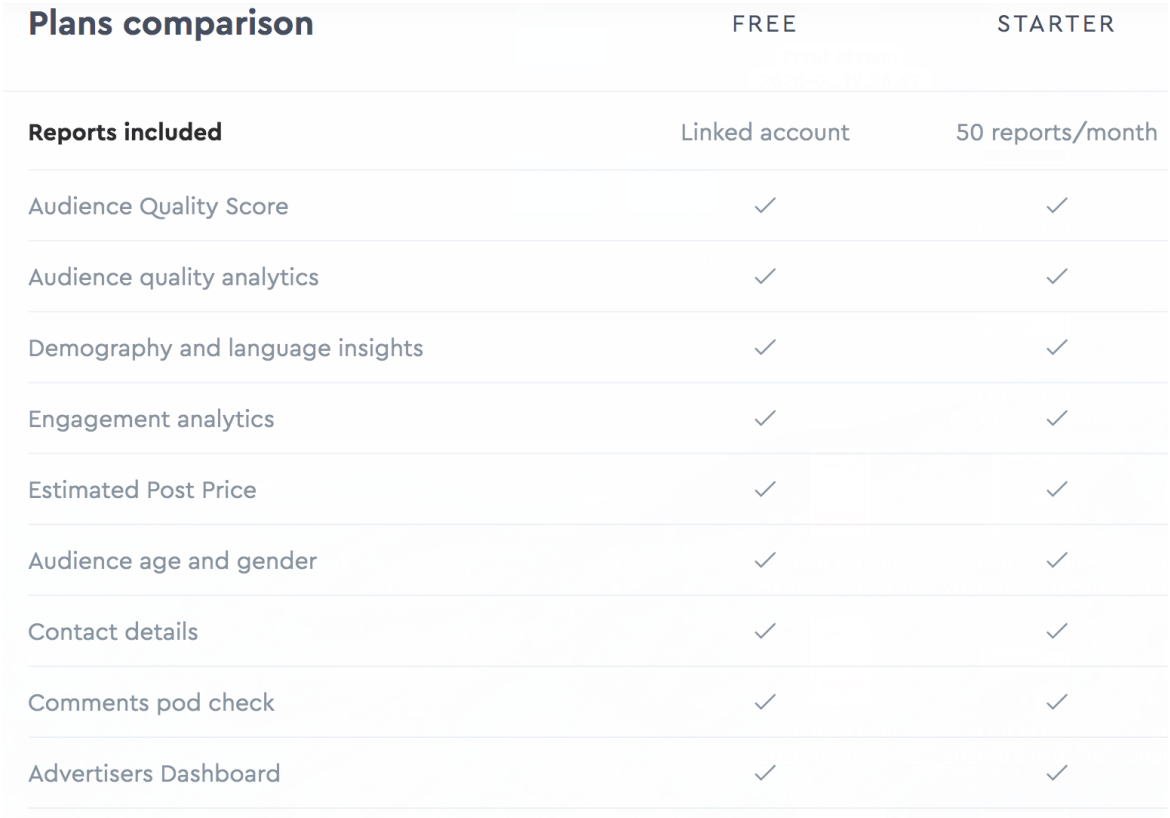
Przed realizacją pracy należy sprawdzić i przeanalizować dostępne rozwiązania. Pomoże to odkryć na co jest zapotrzebowanie, a także jakich rozwiązań i narzędzi brakuje. Analiza pozwoli zweryfikować wstępne pomysły, a ponadto może zainspirować do stworzenia niebranych wcześniej pod uwagę funkcjonalności.

Sieć społecznościowa Instagram została założona 6 października 2010 roku [5] i posiada miliard aktywnych użytkowników w ciągu miesiąca [6]. Naturalnym jest więc, że w ciągu ostatniej dekady powstało wiele urządzeń służących zarówno do analizy jak i do zarządzania profilami w tym serwisie. Niestety w gąszczu różnych narzędzi bardzo często można natknąć się na aplikacje wykorzystujące dane użytkownika. Jednym z takich narzędzi okazała się bardzo popularna aplikacja InstaAgent [7], która miała za zadanie śledzić jacy użytkownicy odwiedzają dany profil. Aby korzystać z aplikacji, należało zalogować się podając dane do konta na Instagramie - login oraz hasło. Jak później wykryto, dane potrzebne do zalogowania wysyłane były na zewnętrzny serwer. W ten sposób tysiącom

użytkowników skradziono hasła do kont, co bezpośrednio naraziło ich na ryzyko kradzieży wraz z setkami obserwatorów. Google i Apple usunęło aplikacje ze swoich sklepów, jednak szkoda wyrządzona użytkownikom pozostała nieodwracalna.

Wśród dostępnych narzędzi istnieją również portale, na których analiza konta jest bezpieczna i nie wymaga podawania hasła do konta na Instagramie. Jedną z firm, która proponuje takie narzędzie jest HypeAuditor [8]. Posiada bardzo dużą bazę użytkowników a rozwiązania, które proponują są innowacyjne i mogą dostarczyć bardzo istotnych informacji. Oprócz porównywania profili i ich analizy, można śledzić przebieg wgranych kampanii. HypeAuditor korzysta również z autorskich narzędzi do wykrywania czy dany użytkownik nie posiada fałszywego ruchu na koncie. Jedną z funkcjonalności, która zapewnia detekcję oszustów, jest przedstawiony na Rys. 1.1 *Comments pod check*. Funkcja ta śledzi komentarze pod postami użytkownika i ocenia ich jakość na podstawie tego przez kogo zostały opublikowane. Jeżeli pod każdym zdjęciem większość komentarzy pisana jest przez te same osoby, występuje wysokie prawdopodobieństwo, że użytkownik korzysta z tzw. *pod groups* czyli grup, w których użytkownicy komentują sobie na wzajem posty.

Niestety nie każdy jest w stanie sobie na to pozwolić. Miesięczny dostęp do analizy pięćdziesięciu profili kosztuje 300$ co dla polskiego rynku może okazać się wygórowaną ceną [9]. Korzystanie z takiego narzędzia pozostaje rozwiązaniem dla dużych przedsiębiorstw, lecz nie rozwiązuje to problemu małych lub dopiero startujących agencji i marek.



Plans comparison	FREE	STARTER
Reports included	Linked account	50 reports/month
Audience Quality Score	✓	✓
Audience quality analytics	✓	✓
Demography and language insights	✓	✓
Engagement analytics	✓	✓
Estimated Post Price	✓	✓
Audience age and gender	✓	✓
Contact details	✓	✓
Comments pod check	✓	✓
Advertisers Dashboard	✓	✓

Rysunek 1.1. Dostępne funkcjonalności HypeAuditor [9]

Rozwiązaniem dla mniejszych i dopiero rozwijających się agencji i marek, może być niniejsza praca. Narzędzie pozwoliłoby pozyskać cenną analizę kont bez dodatkowych opłat. Ponadto aplikacja umożliwiałaby skanowanie każdego konta o profilu biznesowym, dzięki czemu korzystać z niej mogłyby nie tylko firmy, ale także osoby prywatne, które chciałyby dowiedzieć się więcej na temat swojego konta.

2. Realizacja pracy

2.1. Wstępne założenia

Pierwszym podstawowym etapem realizacji każdego projektu jest określenie wstępnych założeń. Pozwala to twórcy zgłębić dany problem i zidentyfikować jego słabe punkty. Już na poziomie wstępnych założeń można zauważyć co będzie trudniejsze do wykonania, na co należy poświęcić więcej czasu i które części zadania mogą okazać się wyjątkowo problematyczne. To właśnie na tym etapie kształtuje się plan realizacji pracy. Bez niego ciężko byłoby dokonać wyboru architektury rozwiązania oraz odpowiednio dobrać wykorzystywane technologie.

Głównym założeniem pracy jest stworzenie aplikacji webowej umożliwiającej użytkownikowi przeskanowanie profili biznesowych w sieci społecznościowej Instagram, która spełnia poniższe założenia:

1. Użytkownik może wyszukać i przeskanować dowolne konto o profilu biznesowym
2. Aplikacja musi informować użytkownika jeżeli konto, które chce wyszukać nie istnieje lub nie posiada profilu biznesowego
3. Aplikacja musi wyświetlać podstawowe dane konta, które skanuje
4. Aplikacja musi obliczać i wyświetlać średnią liczbę polubień, komentarzy oraz średni wskaźnik zaangażowania obserwatorów
5. Aplikacja musi przeszukiwać konto w celu znalezienia użytych hasztagów, zliczać je oraz zliczać liczbę ich użycia
6. Aplikacja musi porównywać listę użytych hasztagów przez użytkownika z listą zabronionych hasztagów i wyświetlać te hasztagi
7. Aplikacja musi obliczać liczbę nowych obserwatorów w ciągu ostatnich 4 tygodni oraz ich przyrost
8. Aplikacja od dnia zarejestrowania do bazy danego konta musi śledzić liczbę obserwatorów i osób obserwowanych, a następnie na tej podstawie tworzyć wykresy przyrostu obserwatorów i osób obserwowanych
9. Aplikacja musi obliczać liczbę dodanych postów w ciągu ostatnich 4 tygodni oraz ich przyrost
10. Aplikacja musi śledzić dzienny przyrost polubień i komentarzy w każdym poście oraz tworzyć wykres słupkowy z tymi danymi
11. Aplikacja musi tworzyć wykres biegunowy z pięcioma wskaźnikami dla każdego profilu
12. Aplikacja musi raz dziennie skanować każde konto w bazie w celu uaktualnienia danych
13. Aplikacja powinna dostosowywać się automatycznie do szerokości przeglądarki tak aby móc z niej korzystać również na tabletach czy telefonach komórkowych

14. Aplikacja powinna prezentować wskaźniki średniej liczby polubień, średniej liczby komentarzy, zaangażowania obserwatorów, liczby obserwujących oraz liczby dodanych postów w ciągu ostatnich 4 tygodni w układzie biegunowym

2.2. Wybór rozwiązania

Jednym z podstawowych i pierwszym do rozwiązania problemem pracy jest sposób pobierania informacji z kont Instagrama. W ramach analizy dokonano porównania dwóch najbardziej popularnych metod pobierania danych - korzystania z oficjalnego API wystawionego przez aplikację Instagram oraz metody tzw. "Web Scrapingu".

2.2.1. Instagram API

Wykorzystanie oficjalnego API niewątpliwie niesie ze sobą wiele korzyści. Pobierając dane z API można mieć pewność, że wszystkie otrzymane informacje są dozwolone do pobrania. Dodatkowo takie dane są już odpowiednio przygotowane i nie trzeba poddawać ich formatowaniu, co przyspiesza czas realizacji pracy. Wadą korzystania z API jest dostępność i niestabilność. Trzeba zawsze pamiętać o tym, że z dnia na dzień dostęp do pobieranych informacji może zostać zamknięty, a co za tym idzie, w pełni działająca aplikacja przestanie funkcjonować. Jest to bardzo duże ryzyko i w przypadku rozwijania komercyjnego biznesu opartego o integrację z API, należy zastanowić się nad awaryjnymi rozwiązaniami.

2.2.2. Web Scraping

Drugą metodą pobierania danych jest tzw. "Web Scraping". Polega ona na bezpośrednim ściąganiu danych z html danej strony lub platformy. Ściągnięcie danych można przeprowadzić ręcznie, jednak zazwyczaj służą do tego specjalnie napisane skrypty np. w języku Python, które pobierają cały kod html strony i wyciągają interesujące odbiorcę informacje. Metoda ta pozwala ściągnąć wszystkie dane, które wyświetlają się na danej stronie w związku z czym odbiorca nie jest w żaden sposób ograniczony. Niestety bardzo często może okazać się nielegalna, ponieważ wykorzystywanie jej nie zawsze jest jednoznacznie określone prawnie. Jeżeli osoba scrapująca stronę wejdzie w posiadanie chronionych danych, może liczyć się z przykrymi konsekwencjami. Tak stało się w przypadku firmy Brand24, która za pobranie zabronionych danych została zbanowana przez grupę sieci społecznościowych Facebook oraz Instagram [10].

W związku z powyższą analizą, do pobierania danych o kontach zostało wykorzystane oficjalne API serwisu Instagram.

2.3. Wybór technologii

Wybór technologii to zaraz po określeniu wstępnych założeń i wymagań, drugi najważniejszy etap tworzenia aplikacji. Jest to kluczowy moment podczas, którego zapada

decyzja jak będzie wyglądać aplikacja. To właśnie na tym etapie rozstrzygnięte zostaje jak poszczególne moduły aplikacji będą się ze sobą komunikować. Dokładna analiza wybranych technologii jest kluczowa.

2.3.1. Języki programowania

Ze względu na założenie tworzenia aplikacji webowej wybór technologii odpowiedzialnych za moduł interfejsu użytkownika zawęża się. Naturalnym rozwiązaniem wydaje się skorzystanie z języka Javascript o standardzie ECMAScript 6, który został wydany 17 czerwca 2015 roku [11]. Jest to język skryptowy o wielu zaletach, który według rankingu StackOverflow po raz siódmy zajmuje pierwsze miejsce w rankingu najpopularniejszych technologii z wynikiem sięgającym aż 67.8% [12]. Nic dziwnego bowiem Javascript jest interpretowany przez wszystkie przeglądarki internetowe, dzięki czemu aplikacja może pozostać uniwersalna, a twórca nie musi martwić się o kompatybilność na różnych platformach.

React.js jest jedną z najpopularniejszych platform programistycznych (z języka angielskiego *framework*), która służy do budowy szkieletów aplikacji. W ten sposób struktura aplikacji zdefiniowana jest poprzez komponenty o określonych zadaniach. Główną zaletą języka Javascript i jego platformy React.js jest pokaźny zbiór bibliotek i tzw. *paczek* NPM (*Node Package Manager*), które zawierają gotowe komponenty i rozwiązania. Jedną z wykorzystanych bibliotek do stworzenia modułu interfejsu będzie npm Nivo, która zawiera zbiór gotowych komponentów do rysowania wykresów [13]. Samodzielne tworzenie takich modułów byłoby bardzo czasochłonne, a dzięki takiemu rozwiązaniu, efekty pracy można przyspieszyć. Drugą z zastosowanych bibliotek będzie Axios [14], która służy do obsługi API i generowania zapytań. Jest to bardzo popularny moduł JavaScript, opierający się na obietnicach (z języka angielskiego *promise*), które upraszczają obsługę asynchronicznie działającego kodu. Konkurencyjną metodą dla Axiosa jest Fetch, który również opiera się na promisach [15]. Jaka jest więc różnica między obiema metodami? Używając Axiosa możemy uniknąć tzw. *bolierplate code*, czyli nadmiarowego kodu, który powtarza się w wielu miejscach, lecz nie można z niego zrezygnować. Nadmiarowy boilerplate code powstaje zwłaszcza przy zapytaniach POST. Porównanie przykładowych fragmentów kodu z zapytaniem POST przedstawiają Rys. 2.1 oraz Rys. 2.2. Wykorzystując metodę *axios.post()* obsługa zapytania wraz z przechwyceniem błędu zawiera się tylko w trzech liniach kodu. W przypadku metody *fetch()*, aby prawidłowo obsłużyć zapytanie, potrzebne jest określenie metody, nagłówków oraz sformatowanie obiektu przesyłanego w zapytaniu. Korzystając z Axiosa nie jest to wymagane.

```
76      const response = await fetch(url, {
77          method: 'POST',
78          headers: {
79              'Content-Type': 'application/json;charset=utf-8'
80          },
81          body: JSON.stringify(element)
82          }).catch( (error) => {
83          return alert("Connection refused. Can't add new list");
84      })
85          if(response!=null){
86              return response.json();
87      }
88  }
```

Rysunek 2.1. Przykładowe zapytanie POST przy użyciu metody fetch()

```
90      axios.post(url, { element })
91          .then(response => response)
92          .catch(error => alert("Connection refused. Can't add new list"))
93
```

Rysunek 2.2. Przykładowe zapytanie POST przy użyciu metody axios.post() z biblioteki Axios

Do implementacji modułu serwerowego wybrany został język Java 8 wraz z platformą programistyczną Spring Boot, która dzięki swojej strategii *konwencji ponad konfiguracją* (z języka angielskiego *Convention Over Configuration*), pozwala omijać kompleksową konfigurację XML, która była nieunikniona w przypadku platformy Spring. Ponadto korzystając ze Spring Boot, twórca nie musi już martwić się posiadaniem zewnętrznego serwera do uruchamiania aplikacji. Spring Boot posiada również liczne adnotacje, które bardzo skracają zapisany kod i przyspieszają prace nad nim. Wystarczy jedynie dodać bibliotekę i uruchomić adnotację, aby stworzyć w pełni działający kod.

Moduł przechowywania danych został oparty o relacyjną bazę danych MySQL. Baza ta charakteryzuje się bardzo dobrą skalowalnością. Ponadto może obsłużyć nawet bardzo dużą liczbę zapytań. Kolejną zaletą MySQL jest uniwersalność i kompatybilność przez co bez problemu może działać zarówno na systemie Windows jak i Linux czy OS X. Aby ułatwić śledzenie, zarządzanie i stosowanie zmian w schematach bazy danych, wykorzystana została biblioteka Liquibase. Liquibase używa tzw. *changesetów*, które reprezentują pojedynczą zmianę w bazie. Pliki z changesetami mogą zawierać się w czterech formatach: SQL, XML, YAML, oraz JSON. W niniejszej pracy wykorzystany został format XML.

2.3.2. Narzędzia wspomagające rozwój pracy

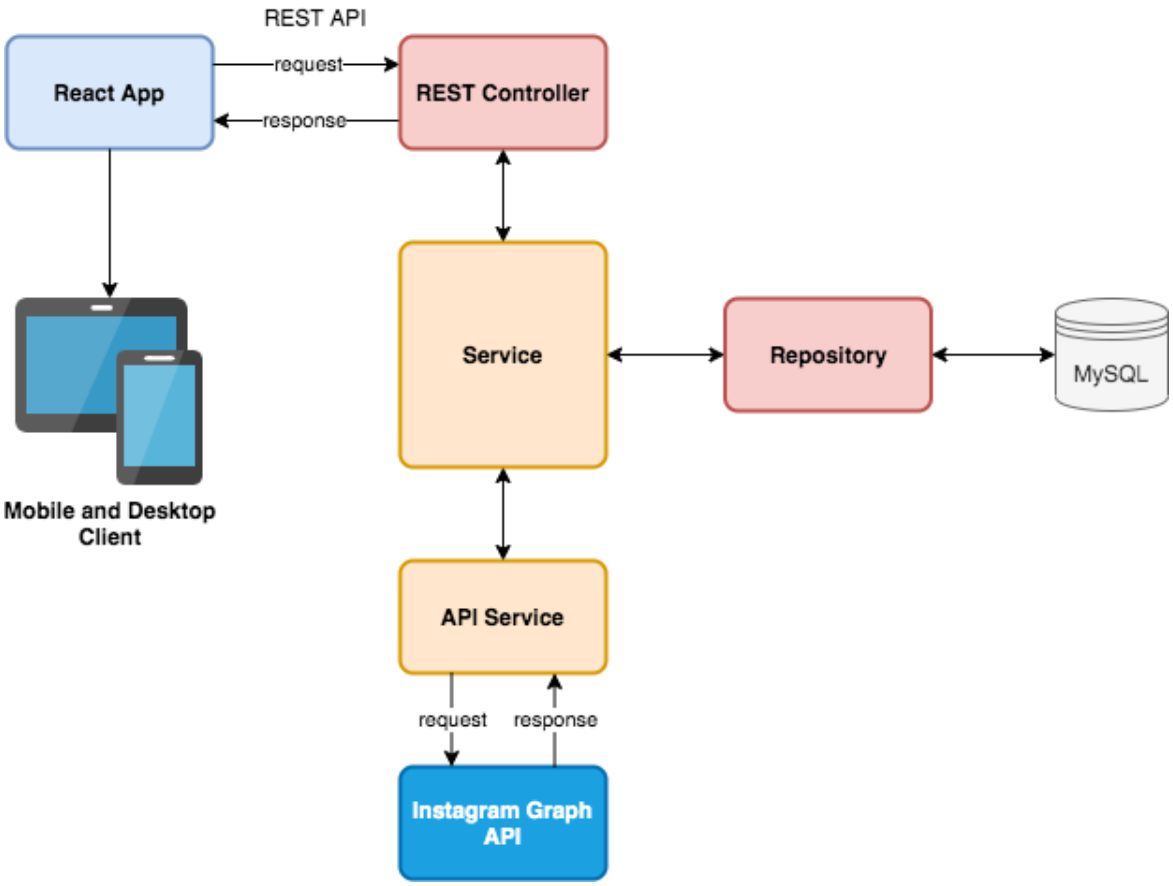
Przed przystąpieniem do pracy warto przemyśleć jakie narzędzia zostaną wykorzystane. Niejednokrotnie odpowiedni dobór narzędzi pozwala przyspieszyć prace programisty, a także uniknąć niechcianych komplikacji.

Przykładem narzędzia, bez którego nie może obejść się każdy programista, jest system kontroli wersji Git. Pozwala on na bezpieczne przechowywanie kodu, równoległą prace wielu programistów nad jednym projektem oraz ułatwia kontrolę nad zmianami. Dzięki swojej przenośności może być używany we wszystkich systemach operacyjnych co również ułatwia pracę. W związku z tym do stworzenia poniżej pracy wykorzystany zostanie hostingowy serwis GitHub, który bazuje na systemie kontroli wersji Git.

Oprócz wybrania serwisu z systemem kontroli wersji, przed rozpoczęciem pracy warto wybrać również zintegrowane środowisko programistyczne, na którym praca będzie pisana. Ze względu na podział aplikacji na część z interfejsem graficznym i część serwerową, praca nad kodem również została podzielona na dwa środowiska. Do implementacji interfejsu graficznego zostało wybrane Visual Studio Code, które jest jednym z popularniejszych środowisk frontendowych. Posiada on wiele przydatnych wtyczek takich jak Beautify, Prettier czy ESLint, które pomagają wykrywać błędy pod kątem podejrzanych instrukcji oraz formatują jego składnię tak aby kod wyglądał przejrzyście i był utrzymany w jednym standardzie. Do implementacji części serwerowej zostało wybrane środowisko IntelliJ. Jest to jeden z najlepszych programów do implementacji kodu w języku Java, które ułatwia i przyspiesza pracę dzięki inteligentnemu uzupełnianiu kodu oraz podpowiedzi.

Do dokumentacji API bardzo pomocnym narzędziem jest Swagger UI [16]. Dzięki Swagger UI w łatwy sposób można otrzymać prosty i przejrzysty opis dla każdego wystawionego punktu końcowego komunikacji (z języka angielskiego *endpoint*). Narzędzie to jest bardzo pomocne zwłaszcza kiedy przy projekcie pracują dwa zespoły odpowiadające za interfejs graficzny i za aplikacje serwerową. Dzięki temu komunikacja między nimi zachodzi dużo szybciej i łatwiej jest zweryfikować zawarty między nimi kontrakt. Szczególnie przydatną funkcją jest opcja wysłania zapytania i wyświetlania odpowiedzi serwera w zależności od statusu na żywo.

2.4. Architektura rozwiązania



Rysunek 2.3. Schemat działania aplikacji

Określając wstępne założenia i dokonując wyboru rozwiązania oraz technologii dużo łatwiej jest sformułować architekturę rozwiązania. Schemat działania aplikacji został przedstawiony na Rys. 2.3. Warstwę prezentacji stanowić będzie reactowa aplikacja, komunikująca się z serwerem za pomocą REST API. W module serwerowym wyróżnić można trzy rodzaje komponentów: *kontrolery, serwisy* oraz *repozytoria*. Wśród serwisów odpowiedzialnych za logikę biznesową aplikacji, najważniejszym serwisem będzie API Service stanowiący komunikację z Instagram Graph API. To on będzie pobierał wszystkie potrzebne dane wybranych profili i zapisywał je do obiektów przechowujących je. Jest to kluczowy serwis, bez którego nie działałaby aplikacja.

2.4.1. Koncepcja DTO

DTO to tzw. *Data Transfer Objects*. Jest to wzorzec projektowy, który opiera się na założeniu, że w związku z tym, że każde połączenie z dowolnym zdalnym interfejsem jest kosztowne, odpowiedź na każde połączenie powinna zawierać jak najwięcej informacji. DTO jest obiektem służącym do transferu danych pomiędzy warstwami aplikacji, jej

modułami, ale także transferem między aplikacjami i systemami. Można go wykorzystać w każdej sytuacji, gdzie transfer jest konieczny.

Zaletą korzystania z takich obiektów z perspektywy młodego twórcy jest przede wszystkim przejrzysty podział i struktura projektu, które DTOsy wprowadzają. W niniejszej pracy w związku z ciągłym transferem danych między serwisami, repozytoriami i kontrolerami, obiekty transferu danych zostały wykorzystane w każdym rodzaju komponentów. Na Rys. 2.4 przedstawiony został obiekt *BusinessInfo* przechowujący wszystkie informacje o profilu użytkownika.

```
@Data
@NoArgsConstructor
public class BusinessInfo {
    private String biography;
    @JsonProperty("followers_count")
    private long followersCount;
    @JsonProperty("follows_count")
    private long followsCount;
    @JsonProperty("media_count")
    private long mediaCount;
    private String username;
    private String name;
    @JsonProperty("profile_picture_url")
    private String profilePictureUrl;
    private MediaDto media;

}
```

Rysunek 2.4. Przykład stworzonego obiektu transferu danych dla podstawowych informacji o profilu użytkownika, wykorzystywany w serwisach.

2.4.2. Facebook Graph API

Aby pobierać dane z API Instagramu, najpierw należy zapoznać się z budową i działaniem Facebookowego Graph API [17]. Jest to autorskie API twórców Facebooka, które opiera się na idei „social graph", która reprezentuje wszystkie informacje na Facebooku. Graph API opiera się na bibliotece GraphQL [18], która jest alternatywnym rozwiązaniem i odejściem od klasycznego REST API. Struktura grafowego API składa się z wierzchołków (z j. angl *nodes*), krawędzi (*edges*) i pól (*fields*). Wierzchołki reprezentują indywidualne obiekty takie jak użytkownik, strona czy post. Krawędzie tworzą połączenie między kolekcjami obiektów, a pojedynczym obiektem. Przykładem takiej krawędzi jest np. post, który znajduje się na stronie, albo komentarz, który znajduje się pod postem. Pola reprezentują

konkretne informacje na temat danego obiektu. Ilustrującym przykładem może być użytkownik - obiekt, który ma pola imię, data urodzenia czy miejsce zamieszkania.

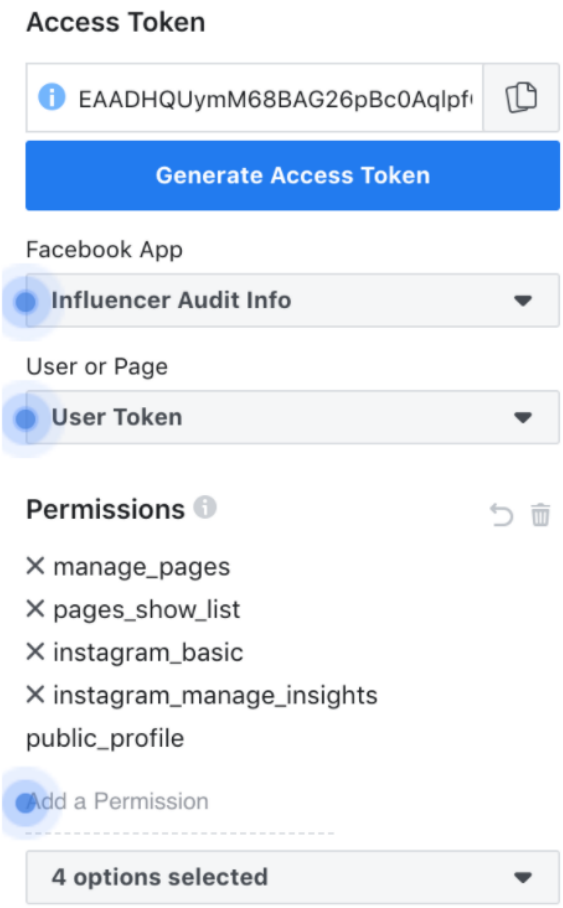
W praktyce wierzchołki stosuje się do pobierania informacji o pojedynczym, konkretnym obiekcie, krawędzi do pobierania informacji o kolekcji obiektów z pojedynczego obiektu, a pól do pobrania informacji z pojedynczego obiektu lub każdego obiektu z kolekcji.

Graph API oparte jest o protokół HTTP, co oznacza, że możemy dokonywać zapytań bezpośrednio z przeglądarki wpisując odpowiedni URL.

Instagram Graph API, jak każde API z rodziny Facebooka, opiera się na Graph API [19]. Oznacza to, że zasady działania i struktura są takie same jak w grafowym API. Ograniczeniem jakie wprowadził Instagram do swojego API jest dostępność - zapytania możemy prowadzić jedynie dla użytkowników o profilu biznesowym, nie jesteśmy w stanie pobrać danych o zwykłym użytkowniku. Aby pobrać informację o zwykłym użytkowniku należy używać Instagram Basic Display API [20], jednak ze względu na wymagania projektowe niniejszej pracy inżynierskiej, użyta zostanie biznesowa wersja. API podstawowe daje dostęp do odczytu jedynie ogólnych informacji np. nie zwraca informacji o liczbie polubień i komentarzy pod każdym postem, co w przypadku tego projektu jest niezbędne.

2.4.3. Pobieranie danych

Jak pobierać dane z Instagram API? Żeby móc do tego przystąpić najpierw należy założyć profil Facebook Developera, a następnie w panelu developerskim stworzyć aplikację. Większość udostępnianych API działa przy użyciu *access token* - specjalnego klucza wygenerowanego dla danego konta, który umożliwia dostęp i pobieranie danych. Podobnie działa np. Google Maps API [21]. Aby wygenerować token dostępu należy otworzyć narzędzie Graph API Explorer. Jest to narzędzie podobne do bardzo popularnego Postmana [22], w którym możemy symulować zapytania do API (Rys. 2.6, Rys. 2.7).



Rysunek 2.5. Narzędzie Graph API Explorer. Generowanie tokenu dostępu

Aby pobierać dane z Instagrama, należy dodać nagłówki *Permissions*, jak na załączonym powyżej Rys. 2.5. Wszystkie nagłówki spisane są w dokumentacji, a to których należy użyć, zależne jest od tego jakie informacje chce się pobrać. W niniejszej pracy, aby móc pobrać wszystkie niezbędne informacje, wykorzystane zostały nagłówki: *manage pages, pages show list, instagram basic, instagram manage insights* oraz *public profile.*

Rysunek 2.6. Narzędzie Graph API Explorer. Przykładowe zapytanie GET

Rysunek 2.7. Narzędzie Graph API Explorer. Wynik zapytania GET

2.4.4. API Service

Jak wcześniej zostało wspomniane, kluczowym i najważniejszym komponentem w całej aplikacji jest API Service. Jest to klasa z adnotacją @Service odpowiadająca za komunikację z Facebook Graph API. To z tego serwisu wysyłane są zapytania do interfejsu Facebooka, aby pobrać wszystkie potrzebne dane na temat wybranych profili.

Klasa IgApiService przechowuje niezbędne dane do wysyłania zapytań. Są to potrzebne punkty końcowe oraz token dostępu. Zawiera zestaw metod potrzebnych do otrzymania wszystkich niezbędnych informacji do stworzenia analizy i statystyk, to jest:

— Pobranie opisu bio, liczby obserwatorów, liczby obserwowanych, liczby dodanych postów, nazwę użytkownika, imię oraz zdjęcie profilowe
— Pobranie listy postów z opisem zdjęcia, liczbą komentarzy pod zdjęciem, liczbą polubień oraz datą dodatnia
— Stronicowanie po liście postów

Powyższe metody wywoływane są w serwisach stworzonych do rejestracji i skanowania profili. Mogą być wywołane także w każdej innej klasie, co pozwala na zachowanie poziomu abstrakcji. Warto zaznaczyć, że dalsza logika z przetwarzaniem danych została zaimplementowana w innych klasach tak aby nie mieszać funkcjonalności poszczególnych serwisów. Zadania takie jak np. obliczanie wskaźnika zaangażowania obserwatorów w profil realizowane są w specjalnie stworzonych do tego serwisach. Taki podział pozwala na zachowanie przejrzystości kodu, utrzymuje jego czytelność i zapewnia bezpieczeństwo dalszego rozwoju aplikacji.

2.4.5. Logika biznesowa

Przyglądając się wstępnym założeniom pracy, wśród wszystkich funkcjonalności można wyodrębnić następujące sekcje:

- Sekcja wyszukiwania profilu
- Sekcja podstawowych informacji o profilu
- Sekcja zaangażowania obserwatorów profilu

- Sekcja hasztagów
- Sekcja obserwatorów
- Sekcja obserwujących
- Sekcja postów

W związku z takim pogrupowaniem, postanowiono podzielić logikę biznesową według powyższej listy. Oprócz wyżej wyróżnionego IgApiService, stworzono jeszcze aż siedem innych klas z adnotacją @Service. Zawierają one pozostałe metody niezbędne do realizacji założeń pracy, to jest:

— Rejestracja profilu i zapisanie go do bazy
— Zapisanie podstawowych informacji o profilu
— Zapisanie informacji o postach i dodanie ich do bazy
— Obliczanie średniej liczby polubień
— Obliczanie średniej liczby komentarzy
— Obliczanie współczynnika zaangażowania obserwatorów
— Obliczanie sumy wszystkich unikatowych hasztagów użytych na profilu
— Stworzenie posortowanej listy użytych hasztagów wraz z liczbą użyć
— Stworzenie listy użytych niedozwolonych hasztagów wraz z liczbą użyć
— Obliczanie przyrostu obserwatorów z ostatnich czterech tygodni w wartości liczbowej i procentowej
— Stworzenie posortowanej listy według daty z liczbą obserwatorów
— Stworzenie posortowanej listy według daty z liczbą obserwowanych kont
— Obliczanie liczby dodanych postów z ostatnich czterech tygodni
— Stworzenie posortowanej listy dodanych postów według daty z liczbą polubień i komentarzy

Kluczowymi klasami, o których należy wspomnieć są ProfileRegisterService oraz ProfileScanService. Są to dwie klasy zajmujące się rejestracją i skanowaniem profilu.

Klasa ProfileScanService posiada metodę *scan*, która służy do przeskanowania danego profilu. To właśnie w tej klasie wykonują się metody z głównego serwisu IgApiService - *getProfileInfo* oraz *getAllMedia*, a następnie przekazywane są do klas stanowiących repozytoria.

Klasa ProfileRegisterService posiada metodę *register*, podczas której wywoływana jest metoda skanowania, należąca do klasy ProfileScanService. Ze względu na specyfikę działania wyszukiwarki, aby uprościć działanie interfejsu graficznego, postanowiono rejestrację profilu i wyszukiwanie go wykonywać za pomocą tego samego pola do wyszukiwania. Przekłada się to na wygodę użytkownika. Osoba wyszukująca profil nie musi zastanawiać się czy dane konto było już wcześniej skanowane i czy znajduje się w bazie. Metoda rejestracji przewiduje zarówno przypadek, w którym dany profil nie istnieje w bazie, jak i sytuacje gdzie profil jest już śledzony od jakiegoś czasu. Aby jednak nie wykonywać skanowania za każdym razem kiedy okazuje się, że profil w bazie jest już wpisany, postanowiono dodać

zabezpieczenie wykonując metodę sprawdzającą *profileRepository.exists()*. Jeśli jednak dany profil wpisany został po raz pierwszy, wykonywana jest metoda *getProfileInfo()* z klasy igApiService, która wywołuje zapytanie do Facebook Graph API, a następnie dzięki metodzie *save()* otrzymane dane zapisywane są do repozytorium ProfileRepository. Dopiero po wykonaniu wyżej wymienionych metod, dany profil ulega skanowaniu. Pełen kod klasy znajduje się na Rys. 2.8.

```
@Service
@RequiredArgsConstructor
public class ProfileRegisterService {

    private final IgApiService igApiService;
    private final ProfileRepository profileRepository;
    private final ProfileScanService profileScanService;

    Logger logger = LoggerFactory.getLogger(ProfileRegisterService.class);

    public boolean register(String profileId) {

        if(profileRepository.exists(profileId)) {
            return true;
        }
        try {
            // REGISTER PROFILE FOR FUTURE SCANS
            BusinessInfo businessInfo = igApiService.getProfileInfo(profileId);
            profileRepository.save(businessInfo.getUsername());
            // FIRST SCAN OF PROFILE
            profileScanService.scan(profileId);
        } catch (Exception exception) {
            logger.error("Unable to register profile {}", exception);
            exception.printStackTrace();
            return false;
        }

        return true;
    }
}
```

Rysunek 2.8. Fragment kodu przedstawiający implementację klasy ProfileRegisterService

2.4.6. Persystencja danych

Moduł persysencji danych został zaimplementowany z wykorzystaniem bazy danych MySQL wraz z biblioteką Liquibase. Do zarządzania danymi stworzone zostały cztery tabele, reprezentowane plikami XML z odpowiednimi changesetami:

- Registered Profile
- Basic Info History
- Media
- Banned Hashtags

Tabela Registered Profile zawiera wszystkie zarejestrowane w bazie profile do skanowania. Składa się z dwóch kolumn: *username* oraz *scan*. Kolumna *username* przechowuje nazwę profilu, natomiast kolumna *scan* przechowuje wartość logiczną zero lub jeden, w zależności od której dany profil będzie podlegał skanowaniu lub nie (Rys. 2.10). Fragment kodu przedstawiający changelog tabeli Registered Profile znajduje się na Rys. 2.9 poniżej.

```
<databaseChangeLog
        xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
    http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.4.xsd"
        logicalFilePath="changelog/003-create-registered-profile-table.xml">

    <changeSet id="003-create-registered-profile-table"  author="Zofia" >

        <preConditions>
            <not>
                <tableExists tableName="REGISTERED_PROFILE" schemaName="iginfodb"/>
            </not>
        </preConditions>

        <createTable tableName="REGISTERED_PROFILE">
            <column name="username" type="varchar(255)">
                <constraints primaryKey="true" nullable="false" unique="true"/>
            </column>
            <column name="scan" type="BIT(1)">
                <constraints nullable="false"/>
            </column>
        </createTable>

    </changeSet>

</databaseChangeLog>
```

Rysunek 2.9. Fragment kodu przedstawiający changelog tabeli Registered Profile

	username	scan
1	aleksandraastepien	☑
2	alexiiteresa	☑
3	calkiemzwyczajnie	☑
4	dagmarajarzynka	☑
5	fff	☑
6	gaaabrielyan	☑
7	jesuisaii	☑

Rysunek 2.10. Fragment tabeli Registered Profile

To czy dany profil będzie skanowany zależy od administratora aplikacji. W każdej chwili skanowanie można wyłączyć korzystając z pola wyboru umieszczonego w kolumnie *scan*. Takie rozwiązanie, choć nie znajduje się w głównych założeniach aplikacji, może być jednak bardzo przydatne, ponieważ pozostawia administratorowi możliwość zarządzania i ewentualnych zmian. Funkcja ta może być też przydatna do testowania.

Tabela Basic Info History przechowuje wszystkie podstawowe informacje z danego dnia (Rys. 2.11). Składa się z kolumn: *username*, *date*, *posts number*, *followers number* oraz *follows number*. Informacje te są niezbędne do tworzenia wykresów śledzących postęp rozwoju profili.

	username	date	posts_number	followers_number	follows_number
1	aleksandraastepien	2020-06-14	81	3336	287
2	aleksandraastepien	2020-06-15	81	3339	287
3	aleksandraastepien	2020-06-16	82	3340	287
4	aleksandraastepien	2020-06-17	82	3343	287
5	aleksandraastepien	2020-06-18	82	3346	287
6	aleksandraastepien	2020-06-19	82	3345	287
7	aleksandraastepien	2020-06-20	83	3351	287
8	aleksandraastepien	2020-06-21	84	3357	288
9	aleksandraastepien	2020-06-22	84	3358	288
10	aleksandraastepien	2020-06-23	84	3358	288

Rysunek 2.11. Fragment tabeli Basic Info History

Tabela Media zawiera kolumny: *id*, *username*, *like count*, *comments count*, *timestamp* oraz *hashtags*. Zebrane są tu wszystkie posty od wszystkich zarejestrowanych w bazie profili.

Tabela Banned Hashtags zawiera tylko jedną kolumnę *hashtags* i przechowuje oficjalnie zablokowane hasztagi, których listę można znaleźć w internecie [23].

Za bezpośrednią komunikację z bazą danych odpowiadają repozytoria. Są to klasy z adnotacją @Repository, które stanowią swoisty pomost pomiędzy bazą danych, a serwisami. Każda klasa z adnotacją @Repository zawiera mechanizm jdbcTemplate, który służy do komunikacji z bazą danych oraz do wywoływania zapytań SQL (Rys. 2.12). Mechanizm jdbcTemplate bazuje na JDBC API (The Java Database Connectivity), jednak eliminuje większość problemów jakie tworzy używanie samego JDBC API. Jednym z nich jest tworzenie nadmiernego kodu, bez którego nie można wywołać zapytania. Korzystanie z czystego JDBC API wymaga od użytkownika stworzenia całego kanału do komunikacji, włącznie z obsługą wyjątków dla każdego zapytania. JdbcTemplate pozwala użytkownikowi dokonywać bezpośrednich zapytań do bazy, omijając niepotrzebnie powtarzające się kroki.

```
public void save(BusinessInfo businessInfo) {

    Map<String, String> params = new HashMap<String, String>() {
        {
            put("username", businessInfo.getUsername());
            put("posts", String.valueOf(businessInfo.getMediaCount()));
            put("followers", String.valueOf(businessInfo.getFollowersCount()));
            put("follows", String.valueOf(businessInfo.getFollowsCount()));
        }
    };
    jdbcTemplate.update(IO.asString( path: "db/sql/insert-into-basic-info-history.sql"), params);
}
```

Rysunek 2.12. Fragment kodu przedstawiający metodę zapisywania podstawowych informacji o profilu do bazy danych.

W pracy zostały zaimplementowane cztery klasy o adnotacji @Repository zawierające metody spełniające poniższe operacje na bazie danych:

- Zapisywanie podstawowych informacji o profilu
- Pobieranie podstawowych informacji o profilu
- Zapisywanie postów zawartych na profilu
- Pobieranie wszystkich zliczonych polubień oraz komentarzy
- Pobieranie wszystkich postów zawartych na profilu
- Rejestrowanie profilu
- Sprawdzanie czy profil istnieje w bazie
- Pobieranie wszystkich profili zapisanych w bazie
- Pobieranie wszystkich zabronionych hasztagów

2.4.7. REST Controller

Do komunikacji modułu interfejsu graficznego z modułem serwerowym wykorzystane zostało REST API. Jest to specjalnie wystawiany interfejs na module serwerowym, który otrzymując odpowiednie zapytania od modułu graficznego na konkretny adres, zwraca dane przechowywane na tym adresie. REST API wykorzystuje metody protokółu HTTP (*Hypertext Transfer Protocol*), których łącznie jest dziewięć, jednak do stworzenia podstawowej aplikacji pozwalającej odczytywać, tworzyć, edytować i usuwać dane wystarczą cztery najbardziej znane - GET, POST, PUT oraz DELETE. Główną zaletą wykorzystania tego sposobu komunikacji jest bezstanowość, jasny podział na klienty i serwer oraz niezależność od technologii - REST API może być skonsumowane przez dowolnego klienta.

Aplikacja modułu interfejsu graficznego opiera się głównie na zapytaniach GET, ponieważ kluczową funkcjonalnością widoków jest wyświetlanie analizy konta, która odbywa się po stronie serwera. Wyjątkiem jest moment wyszukiwania konta - klikając przycisk *search* wysyłane jest zapytanie POST wraz z obiektem body zawierającym nazwę profilu.

Po stronie serwera, komponentem wystawiającym API jest kontroler (ang. *controller)*.

Jest to klasa o adnotacji @RestController, która zawiera metody zwracające dane na odpowiedni adres (Rys. 2.13). Aby dana metoda została przypisana do danego adresu url potrzebne jest dodanie adnotacji @RequestMapping, która przechowuje ścieżkę i informację jakie zapytanie obsługuje. Dzięki wykorzystaniu @RestController, a nie adnotacji @Controller, do metody wystawiającej dane nie trzeba dodawać adnotacji @ResponseBody. Obiekt zwracany przez metodę jest automatycznie serializowany jako treść odpowiedzi.

W pracy zaimplementowanych zostało siedem kontrolerów, każdy z nich odpowiada za osobną sekcję danych. Stworzone punkty końcowe:

- /profile/id/engagement/statistics
- /profile/id/followers/history
- /profile/id/followers/statistics
- /profile/id/following/history
- /profile/id/hashtags/statistics
- /profile/id/hashtags/statistics/top
- /profile/id/posts/history
- /profile/id/posts/statistics
- /profile/id/basic-info
- /profiles

```java
@CrossOrigin
@RestController
@RequiredArgsConstructor
public class EngagementController {

    private final EngagementService engagementService;

    @RequestMapping(value = "/profile/{id}/engagement/statistics", method = RequestMethod.GET)
    public EngagementInfo getEngagementInfo(@PathVariable("id") String id) {
        return engagementService.fetchData(id);
    }

}
```

Rysunek 2.13. Przykład kontrolera wystawiający informacje na temat zaangażowania obserwatorów danego konta.

2.4.8. Problem odświeżania danych

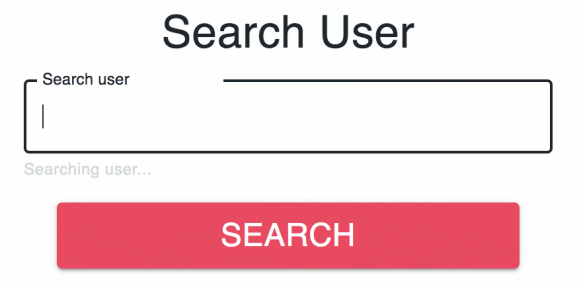
Jednym z problemów napotkanych podczas tworzenia aplikacji był sposób implementacji odświeżania informacji profili zapisanych w bazie danych. Rozwiązaniem było wykorzystanie adnotacji @Scheduled z platformy programistycznej Spring Boot, dzięki której można zaplanować wywoływanie danej funkcji w konkretnym czasie. Ostatecznie serwis planowania skanowania kont wywoływany jest co 24h i realizuje się dla każdego konta z flagą *isScan*. Fragment kodu przestawiający klasę ScheduledProfileScanner znajduje się na Rys. 2.14 poniżej.

```java
@Service
@RequiredArgsConstructor
public class ScheduledProfileScannerService {
    private final ProfileRepository profileRepository;
    private final ProfileScanService profileScanService;

    // every 24h
    @Scheduled(fixedRate = 86_400_000)
    public void scanAllRegisteredProfiles() {

        List<RegisteredProfileDbDto> profiles = profileRepository.getAllProfiles();

        profiles.stream()
                .filter(RegisteredProfileDbDto::isScan)
                .forEach(profile -> profileScanService.scan(profile.getUsername()));
    }
}
```

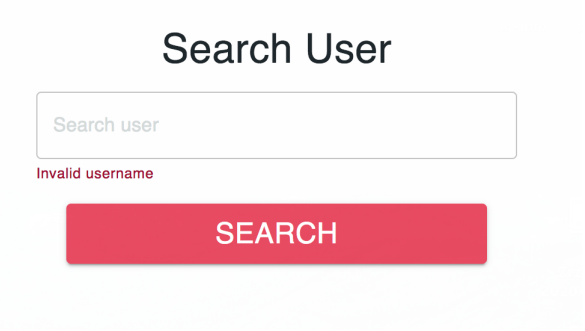
Rysunek 2.14. Fragment kodu przedstawiający ScheduledProfileScanner serwis.

2.5. Interfejs graficzny aplikacji

Poniższe zrzuty ekranu przestawiają stworzoną aplikację. Zrealizowany interfejs graficzny w swych założeniach miał być łatwy w obsłudze i intuicyjny. W związku z tym po wejściu na stronę użytkownik otrzymuje prosty widok z polem do wyszukiwania, które podczas szukania konta i pobierania danych informuje użytkownika o wyszukiwaniu (Rys. 2.15) lub ewentualnej porażce (Rys. 2.16) np. w przypadku kiedy użytkownik chce wyszukać profil, który nie ma konta biznesowego.



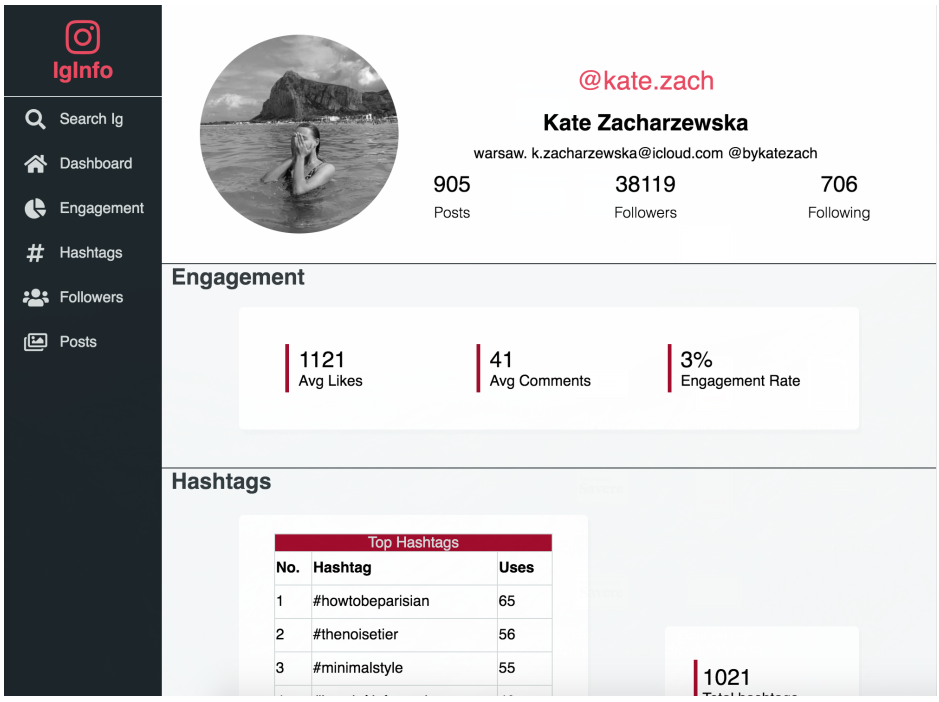
Rysunek 2.15. Informacja o stanie wyszukiwania



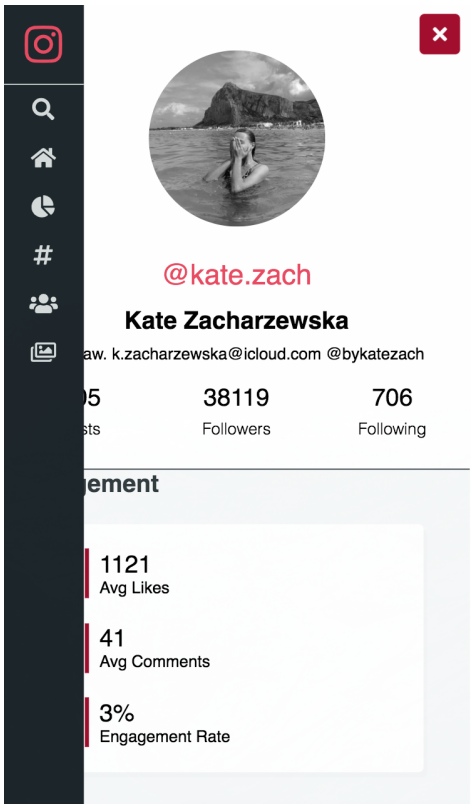
Rysunek 2.16. Informacja o niewłaściwym wyszukaniu

Z myślą o użytkowniku stworzone zostało także proste menu, które wyświetla się po wyszukaniu konta (Rys. 2.17). Klikając pierwszy element z menu - *Search* użytkownik zostanie cofnięty do głównego ekranu szukania, natomiast kolejne elementy z menu zawierają tzw. *kotwicę*, która przekierowuje do konkretnych podsekcji analizy konta. Dzięki temu użytkownik w łatwy sposób może przenieść się do dowolnie interesującej go kategorii analizy.

Ze względu na założenie o responsywności aplikacji, widok z analizą konta zrealizowany został tak aby można było obsługiwać aplikację również na telefonach czy tabletach. W tym celu stworzono kilka wersji menu dla różnych szerokości ekranu. W wersji mobilnej zrealizowano popularne rozwiązanie wysuwającego się menu z prawej strony. Menu obsługiwane jest ikoną znajdującą się w prawym górnym rogu ekranu (Rys. 2.18), która po kliknięciu otwiera lub chowa menu.



Rysunek 2.17. Widok z menu - wersja przeglądarkowa

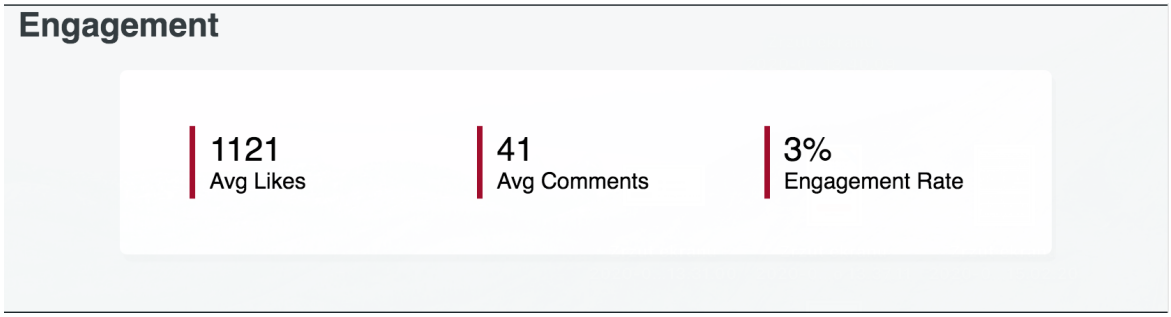


Rysunek 2.18. Widok z rozwiniętym menu - wersja mobilna

Aby analiza konta przebiegała przejrzyście, postanowiono wyświetlać poszczególne sekcje analizy zgodnie ze wcześniejszym podziałem. W pierwszej sekcji (Rys. 2.19) wyświetlane są podstawowe informacje o koncie takie jak nazwa użytkownika, imię, opis bio oraz liczby obserwatorów, obserwowanych i postów. Umieszczone jest też zdjęcie profilowe użytkownika.

Rysunek 2.19. Sekcja podstawowych informacji o koncie

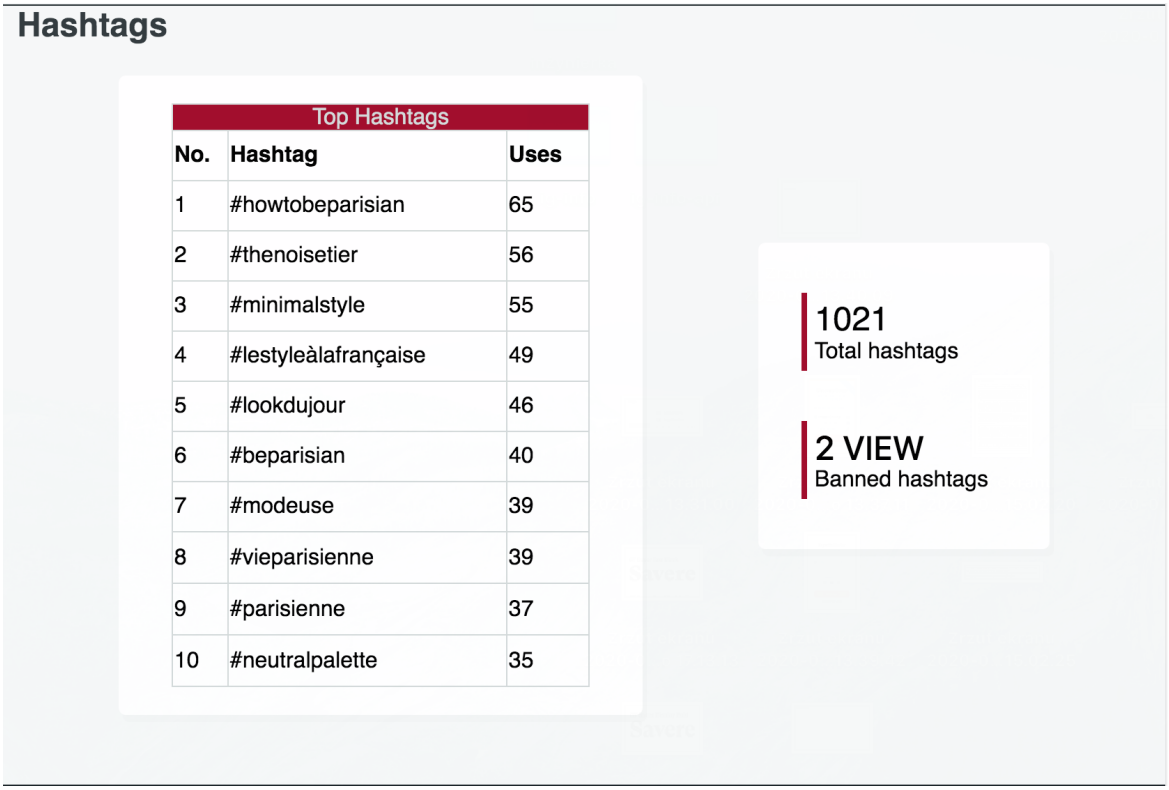
Kolejną sekcją jest sekcja *Engagement* (Zrzut ekranu 2.20) zawierająca statystyki aktywności obserwatorów konta. *Avg Likes* odpowiada za średnią liczbę polubień pod każdym postem, zaś *Avg Comments* za średnią liczbę komentarzy. Jednym z najważniejszych współczynników jest trzeci - *Engagement Rate* - współczynnik zaangażowania, który opisuje udział obserwatorów skanowanego konta. Obliczany jest poprzez podzielenie wszystkich reakcji obserwatorów (polubienia + komentarze) przez liczbę wszystkich obserwatorów. Wynik podawany jest w formie procentowej, dlatego na koniec mnożony jest przez 100. Współczynnik ten odzwierciedla zaangażowanie obserwatorów i opisuje jaki ich procent reaguje na aktywność profilu. W przypadku konta *kate.zach*, która posiada 38119 obserwatorów, a jej ER wynosi 3%, oznacza to, że w zaokrągleniu 1144 osoby dokonują aktywności na jej profilu. Wskaźnik ten jest szczególnie ważny dla reklamodawców, ponieważ wskazuje realną liczbę odbiorców.



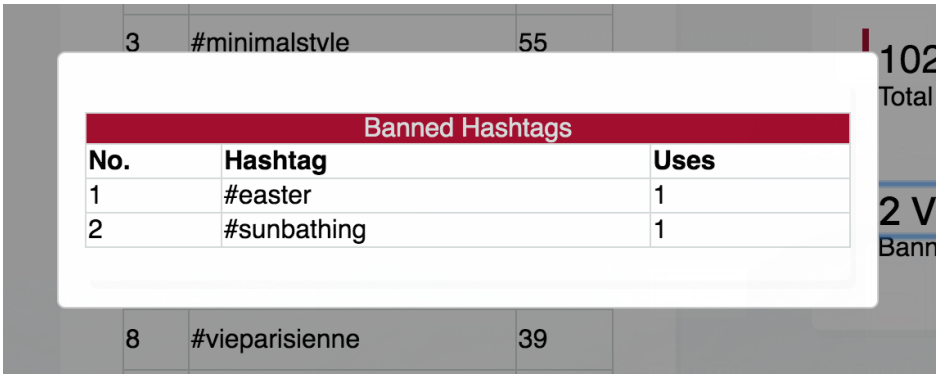
Rysunek 2.20. Sekcja statystyk aktywności obserwatorów konta

W trzeciej sekcji (Rys. 2.21) znajdują się informacje na temat najczęściej używanych hasztagów. Tabela z dziesięcioma najczęściej używanymi hasztagami może być przydatna zwłaszcza dla prywatnych użytkowników aplikacji, który skanują swoje konta. Najczęściej używane hasztagi pozwalają szybko ustalić, czy aby na pewno, wybierane są zgodnie z tematyką profilu.

W formie wyskakującego okienka zaimplementowano tabelę z użytymi hasztagami, które znajdują się na liście zakazanych (Rys. 2.22). Zarówno dla reklamodawców jak i użytkowników prywatnych tabela ma ogromne znaczenie. Używając zakazanych hasztagów konto na Instagramie może otrzymać tzw. *shadowban* [24]. Jest to nic innego jak blokada, która sprawia, że nowe posty publikowane z zablokowanego konta przestają się wyświetlać innym użytkownikom. Im więcej używanych jest zakazanych hasztagów, tym większe prawdopodobieństwo, że konto zostanie zablokowane.



Rysunek 2.21. Sekcja z analizą użytych hasztagów



Rysunek 2.22. Okienko wyskakujące z zakazanymi hasztagami wraz z liczbą użyć

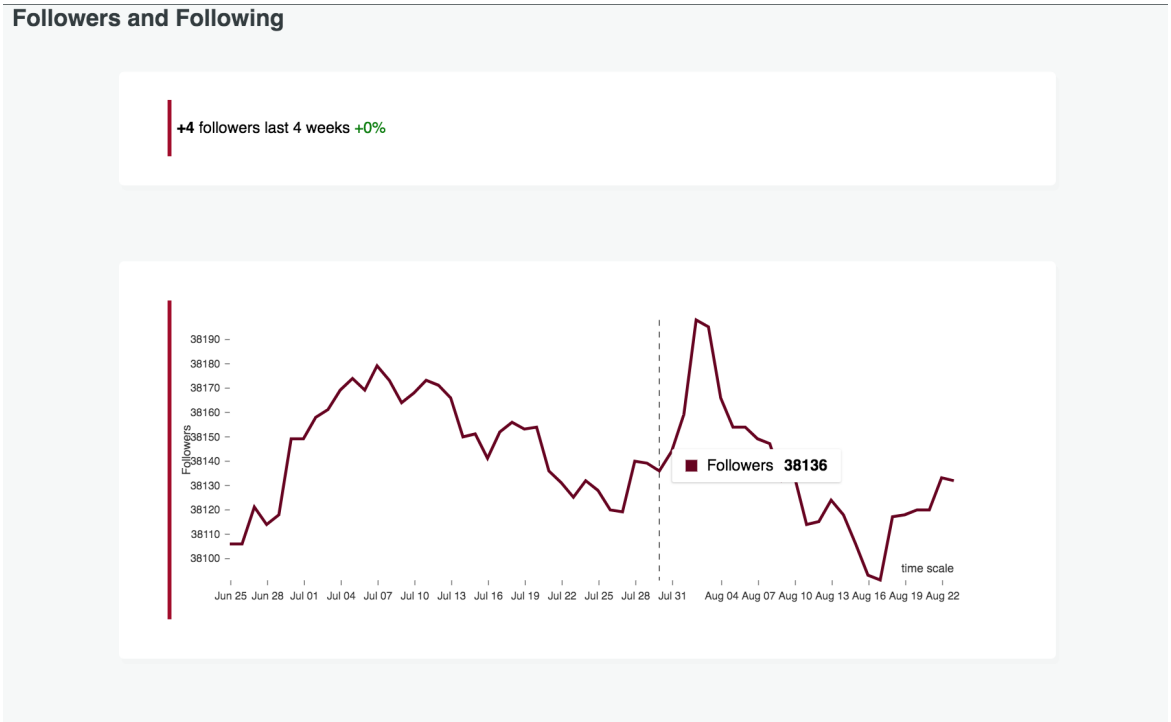
Kolejną sekcją jest analiza obserwatorów i obserwujących (Rys. 2.23, Rys. 2.24). Za pomocą wykresów liniowych przedstawione zostało zachowanie obserwatorów profilu oraz samego profilu. Dzięki wykresom od dnia rejestracji profilu, można śledzić jego rozwój. Oprócz wykresów w sekcji znajduje się przyrost obserwatorów w ciągu ostatnich czterech tygodni.

Wykresy stanowią bardzo ważną część analizy zachowań odbiorców profilu. Zazwyczaj jeśli przyrost jest organiczny, wykres obserwatorów powinien mieć łagodny wzrost. Naturalne są też chwilowe przestoje, gdzie liczba obserwatorów waha się o małe liczby, lecz średnio jest taka sama. Bardzo często niektóre posty cieszą się większą popularnością,

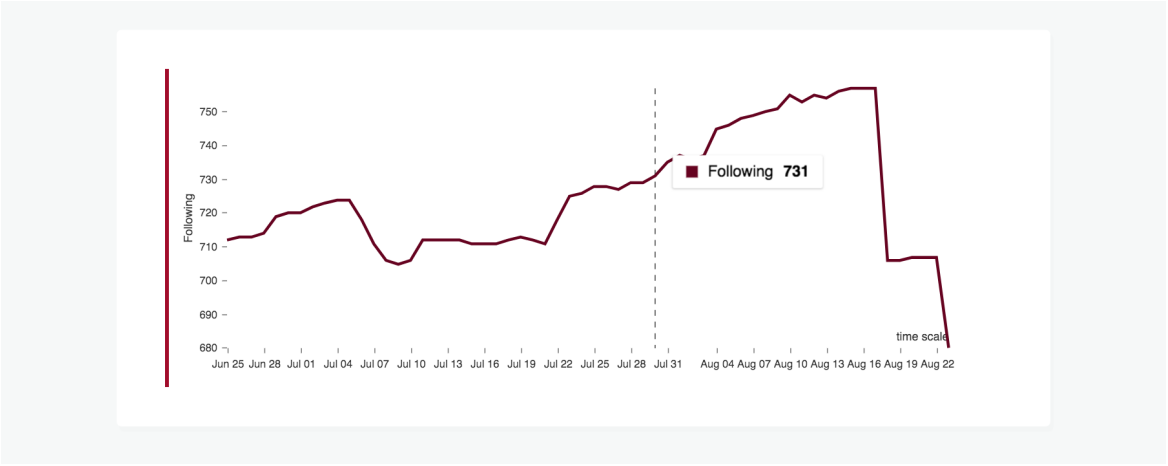
co sprawia, że zdarzają się jednorazowe skoki w liczbie obserwatorów, rzędu kilkuset osób. W przypadku kiedy wykres zawiera systematyczne wzrosty i spadki liczące kilkuset obserwatorów, zachowanie profilu można określić jako podejrzane. Metodą, która często wykorzystywana jest przez influencerów, do podnoszenia liczby obserwatorów, jest kupowanie botów. W ten sposób profil w bardzo krótkim czasie potrafi otrzymać nawet kilka tysięcy nowych obserwatorów. Takie oszustwo jest bardzo skrupulatnie śledzone przez Instagram, a fałszywie założone konta zostają usuwane. W związku z tym profil odnotowuje chwilowy wzrost obserwatorów, ale później bardzo szybki spadek.

Drugą metodą, którą stosują osoby prowadzące Instagram jest tzw. obserwowanie za obserwowanie. Metoda polega na regularnym obserwowaniu innych kont, by po określonym czasie np. dwóch dni, jeśli dane konto nie zaobserwowało z wzajemnością, przestać je obserwować. Zachowanie to łatwo zweryfikować, ponieważ wykres z liczbą osób, które dany profil obserwuje, również zawiera spadki i wzrosty. Może on przyjąć podobną formę jak w przypadku wykresu obserwatorów przy używaniu botów, jednak wzrosty i spadki liczone są w dziesiątkach, a nie w setkach.

Dokonując analizy konta *kate.zach*, którego wykresy obserwatorów i obserwujących znajdują się na Rys. 2.23 oraz Rys. 2.24, można przypuszczać, że osoba ta stosuje wyżej wymienioną metodę z botami. Wykres prezentuje wzrosty i spadki, jednak największa różnica wynosi około 60 obserwatorów, a średnie spadki wynoszą od 10 do 20 obserwatorów. Takie zachowanie można uznać za normalne, biorąc pod uwagę, że konto posiada ponad 38 tysięcy obserwatorów. Porównując liczbę obserwatorów na początku i na końcu miesiąca, jest ona przybliżona. Dodatkowo, wykres osób obserwowanych przez profil również utrzymywał swój stały poziom. Pod koniec ostatniego tygodnia widać gwałtowny spadek, jednak nie jest tu zachowana żadna systematyczność. W związku z tym spadek ten można uznać za naturalną czynność, wykonaną w celu selekcji osób, jakie profil kate.zach obserwował.

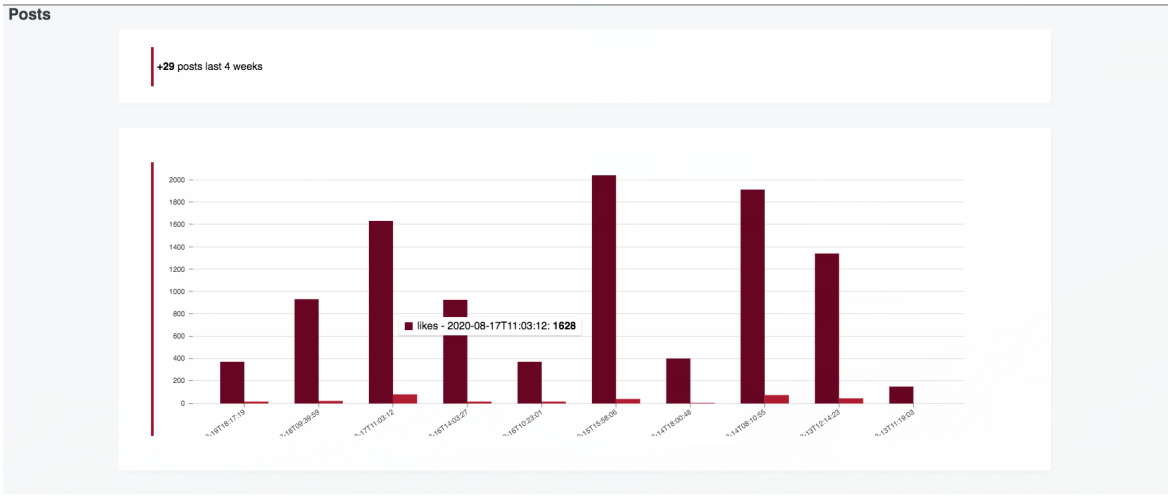


Rysunek 2.23. Sekcja z analizą obserwatorów i obserwujących cz. 1

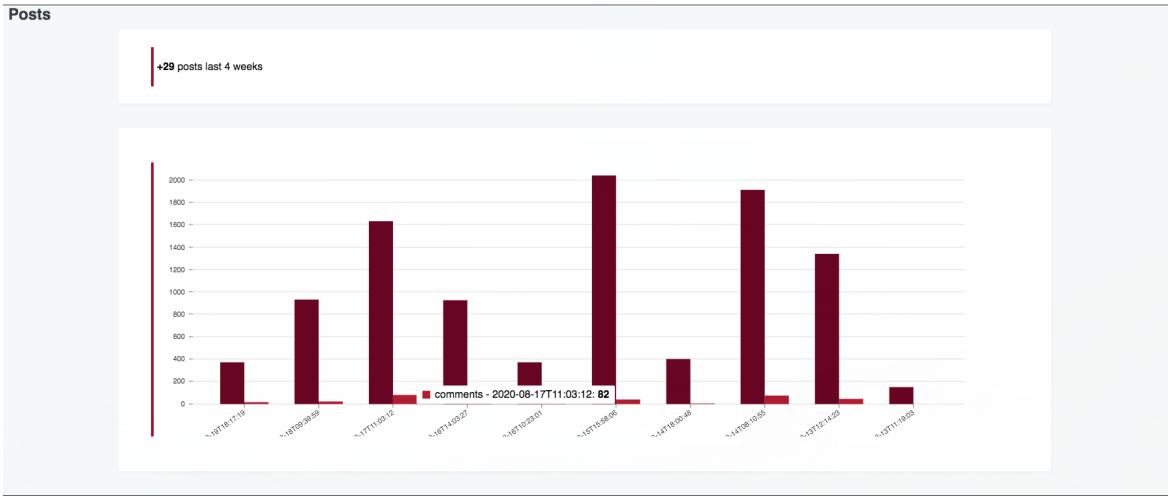


Rysunek 2.24. Sekcja z analizą obserwatorów i obserwujących cz. 2

Przedostatnią sekcją jest analiza postów. Za pomocą wykresu słupkowego przedstawionych zostało dziesięć ostatnich postów. Dla każdego dodanego zdjęcia wyświetlona jest liczba polubień (Rys. 2.25) oraz komentarzy (Rys. 2.26). Taka forma pozwoli w łatwiejszy sposób dostrzec, które posty cieszyły się największą popularnością.

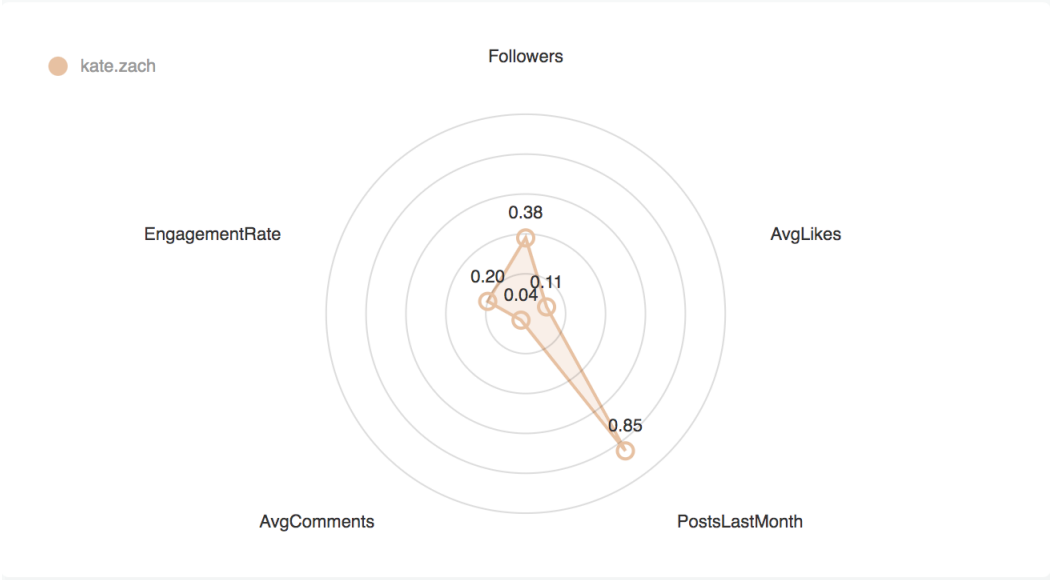


Rysunek 2.25. Sekcja z analizą postów. Liczba polubień



Rysunek 2.26. Sekcja z analizą postów. Liczba komentarzy

Ostatnia sekcja została stworzona specjalnie do analizy pod kątem wyboru profili do kampanii. Wybrane wskaźniki: średniej liczby polubień (*AvgLikes*), średniej liczby komentarzy (*Avg Comments*), zaangażowania obserwatorów (*EngagementRate*), liczby obserwujących (*Followers*) oraz liczby dodanych postów w ciągu ostatnich 4 tygodni (*PostsLastMonth*) zostały znormalizowane, a następnie przedstawione na wykresie z układem biegunowym (Rys. 2.27). W układzie maksymalna wartość jaką może przyjąć współczynnik wynosi 1.



Rysunek 2.27. Prezentacja wskaźników: AvgLikes, AvgComments, EngagementRate, Followers, PostsLastMonth w układzie biegunowym.

3. Testy

3.1. Testy funkcjonalne

Testowanie aplikacji podzielone było na kilka faz. Pierwszą zaimplementowaną częścią aplikacji był interfejs graficzny. W związku z tym testowanie działania komponentów przebiegało na stałych danych zapisanych w plikach JavaScript. Testowane były główne funkcjonalności takie jak wyświetlanie danych, prawidłowe rysowanie wykresów, a także sprawność routingu na stronie. Dodatkowo ze względu na założenie o responsywności, aplikacja była testowana również pod tym kątem. W tym celu wykorzystany został przeglądarkowy Responsive Tool, który umożliwia twórcy dowolne dopasowanie szerokości i wysokości ekranu. Ponadto interfejs testowany był również na telefonie oraz na bardzo dużym monitorze.

Kolejną zaimplementowaną częścią aplikacji była część serwerowa. W tym wypadku testy były wykonywane sukcesywnie wraz z postępem prac. Do sprawdzenia działania wystawionego API bardzo przydatnym urządzeniem okazał się Swagger UI, który dzięki metodzie sprawdzania zapytań na żywo, pozwalał na szybką weryfikację ewentualnych błędów.

Ostatnim etapem implementacji była integracja interfejsu graficznego z częścią serwerową. Tu również testy przebiegały stopniowo wraz z postępem prac. Przy testach brane były pod uwagę różne warunki działania takie jak prędkość łącza, które można dowolnie zmieniać używając Network Tool. Zauważono, że przy wolnej pracy łącza dane bardzo wolno się aktualizują. W związku z tym po stronie interfejsu dla każdej sekcji zaimplementowano kółka ładowania, które kręcą się do momentu załadowania danych.

Testowano również obsługę błędów w przypadku kiedy dane konto, które użytkownik chce wyszukać nie istnieje. Postanowiono dodać informacje o błędzie i wyświetlać ją w interfejsie. W momencie uzyskania odpowiedzi z serwera z kodem błędu użytkownik otrzyma informację pod polem do wyszukiwania.

3.2. Symulacja wyboru profili do kampanii reklamowej

Kluczowym etapem testowania aplikacji była symulacja wyboru profili do kampanii reklamowej. W tym celu wybrano dziesięć profili o podobnej tematyce i zarejestrowano je w bazie. Aktywność kont śledzono przez kolejne 4 tygodnie tak aby powstały całkowite wykresy dla obserwatorów, obserwujących i postów. Po 4 tygodniach zbierania danych przystąpiono do wyboru profili. W tym celu szczególnie ważną rolę odegrało narzędzie przestawiające wskaźniki profilu w układzie biegunowym.

Ze względu na dużą rozpiętość wyników dla takich wartości jak np. liczba obserwatorów, postanowiono dokonać normalizacji wskaźników, tak aby przedstawione wyniki były miarodajne i porównywalne. Przed dokonaniem normalizacji stworzone zostały tzw. najbardziej pożądane wartości współczynników, które w układzie biegunowym przyjmować będą wartość 1. Tabela 3.1 prezentuje ustalone wartości.

Tabela 3.1. Tabela najbardziej pożądanych wartości współczynników

Rodzaj współczynnika	Pożądana wartość
Followers	100000
AvgLikes	10000
PostsLastMonth	20
AvgComments	1000
EngagementRate	15

W celu normalizacji współczynników, każdy z nich został podzielony przez ich najbardziej pożądaną wartość. W przypadku, w którym jeden ze współczynników wynosiłby więcej niż pożądana wartość, przyjmuje on wartość 1.

Do symulacji wybrane zostały profile prowadzone przez kobiety w wieku 20-30 lat, o tematyce związanej z modą. Założono, że najważniejszymi wskaźnikami będą: średnia liczba polubień pod postem - *AvgLikes*, zaangażowanie obserwatorów - *EngagementRate* oraz częstotliwość dodawania postów - *PostsLastMonth*. Pozostałe wskaźniki będą również brane pod uwagę lecz z mniejszym priorytetem. Podsumowanie otrzymanych wyników prezentuje tabela 3.2. Wybrane profile uszeregowane są w kolejności alfabetycznej, zaś wskaźniki według wyżej ustalonej hierarchii. Cztery najwyższe wyniki zostały zaznaczone pogrubioną czcionką.

Tabela 3.2. Wybrane profile wraz z wartościami wskaźników.

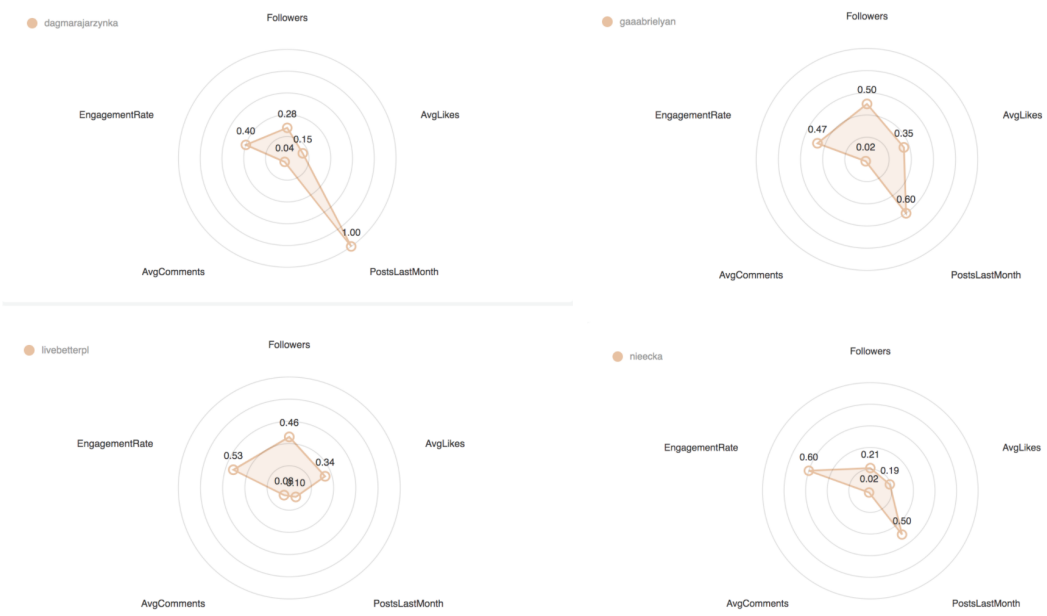
Profil	AvgLikes	ER	PostsLastMonth	Followers	AvgComments
alexiiteresa	0.07	0.13	**1**	0.32	0.01
dagmarajarzynka	**0.15**	0.4	**1**	0.28	**0.04**
gaaabrielyan	**0.35**	**0.47**	0.6	**0.5**	0.02
kasspalmas	0.13	**0.47**	0.3	0.2	**0.1**
kate.zach	0.11	0.2	**0.85**	**0.38**	**0.04**
livebetterpl	**0.34**	**0.53**	0.1	**0.46**	**0.08**
nieecka	**0.19**	**0.6**	0.5	0.21	0.02
oliviakijo	0.05	0.13	0.55	0.27	0.02
oliwiapakosz	0.09	0.13	**0.85**	**0.44**	0.03
rapujacepierogi	0.06	**0.47**	0.3	0.10	0.02

Do kampanii postanowiono wybrać cztery z pośród dziesięciu profili. Profile: dagmarajarzynka, gaaabrielyan, nieecka oraz livebetterpl miały najwyższe wyniki w dwóch z trzech najważniejszych współczynników i to one zostały zakwalifikowane do wzięcia udziału w kampanii reklamowej.

Profil kate.zach posiada najwyższe wyniki dla trzech współczynników, jednak o mniejszej hierarchii i pomimo wysokiej liczby obserwatorów - wynik *Followers* równy 0.38, to współczynnik zaangażowania obserwatorów *EngagementRate* wynosi jedynie 0.2. W związku z tym profil nie został wybrany do udziału w kampanii.

Warto też zwrócić uwagę na profil rapujacepierogi, który pomimo niskich wyników dla *AvgLikes* i AvgComments posiada jeden z najwyższych wyników *EngagementRate*. Dzieje się tak, ponieważ profil ten ma stosunkowo mało obserwatorów - wynik *Followers* równy 0.1. Ostatecznie profil ten nie został wybrany do kampanii, jednak gdyby częstotliwość dodawania postów była większa, mógłby być wzięty pod uwagę.

Prezentacja wyników wybranych profili w układzie biegunowym znajduje się na Rys. 3.1.



Rysunek 3.1. Prezentacja wyników wybranych profili w układzie biegunowym.

4. Podsumowanie

W ramach niniejszej pracy powstała aplikacja webowa umożliwiająca użytkownikowi przeskanowanie profili biznesowych w sieci społecznościowej Instagram. Aplikacja może być wykorzystywana zarówno przez osoby prywatne, posiadające konta o profilu biznesowym jak i przez agencje reklamowe planujące nadchodzące kampanie i wybór osób biorących w nich udział. Narzędzie dostępne jest do powszechnego użytku. Jest w pełni bezpłatne i nie wymaga logowania z użyciem danych do konta na Instagramie. Ponadto aplikacja spełnia założenie o responsywności, w związku z tym może być używana zarówno w przeglądarkach jak i na urządzeniach mobilnych takich jak smartfony czy tablety.

Minusem aplikacji jest ograniczenie liczby zapytań na godzinę. Obecnie Facebook Graph API pozwala na wykonanie 200 zapytań w ciągu godziny. Problem ten mógłby być bardzo istotny, jeżeli w przyszłości aplikacja miałaby być wykorzystywana komercyjnie. Obecnie przy każdym wyszukaniu profilu, wysyłane jest zapytanie do Facebook Graph API w celu pobrania najnowszych informacji o koncie. Być może w przyszłości, lepszym rozwiązaniem byłoby ograniczenie wysyłanych zapytań i odświeżanie danych raz dziennie. Wykorzystując takie rozwiązanie, problem ograniczenia liczby zapytań zostałby rozwiązany.

Atrakcyjną funkcjonalnością dla reklamodawców mogłoby być filtrowanie kont z bazy po użytych hasztagach. Dzięki temu, tak rozwinięta wyszukiwarka mogłaby znajdować interesujące agencję konta o podobnej tematyce. Ułatwiłoby to i przyspieszyło proces znajdywania profili. Wyszukiwarka mogłaby również zweryfikować czy konta, które były brane pod uwagę faktycznie są adekwatne do kampanii.

Kolejnym przydatnym rozszerzeniem byłoby rozwinięcie istniejącego już wykresu z układem biegunowym. Obecnie układ pokazuje jedynie wartości wskaźników dla jednego konta. Ciekawym i bardzo pomocnym rozwiązaniem byłaby możliwość nakładania współczynników kilku profili, tak aby porównywać ich mocne i słabe strony. Dodatkowym atutem byłaby możliwość edycji najbardziej pożądanych wartości wskaźników. Dzięki takiemu rozwiązaniu agencje i domy mediowe mogłyby dowolnie regulować swoje zapotrzebowania.

Wykresy przedstawiające obserwatorów również mogłyby być rozwinięte o mechanizm wykrywający boty, najlepiej oparty o samouczenie. Znając i analizując zachowanie obserwatorów z przeszłości, mechanizm mógłby wykrywać podejrzany ruch na profilu np. kiedy właściciel konta korzysta z wykupionych botów do obserwowania profilu. Ponadto na podstawie danych z przeszłości mógłby szacować przyrost obserwatorów w kolejnych miesiącach.

Bibliografia

[1] Starcom, Dostęp zdalny (11.08.2020): https://www.wirtualnemedia.pl/artykul/reklama-wydatki-reklamowe-w-2019-roku-w-telewizji-internecie-prasie-radiu-i-kinach.

[2] M. PBI/Gemius, *Najpopularniejsze serwisy internetowe i aplikacje mobilne w Polsce wyniki Gemius*, Dostęp zdalny (11.08.2020): https://www.wirtualnemedia.pl/artykul/najpopularniejsze-serwisy-internetowe-i-aplikacje-mobilne-w-polsce-wyniki-gemius-pbi-z-sierpnia-2019-roku-google-i-facebook-w-dol-olx-przed-interia/page:1.

[3] Instagram, *Profil Roberta Lewandowskiego*, Dostęp zdalny (12.08.2020): https://www.instagram.com/_rl9/?hl=pl.

[4] G. U. Statystyczny, *Ludność. Stan i struktura w przekroju terytorialnym (stan w dniu 30.06.2019)*, Dostęp zdalny (12.08.2020): https://stat.gov.pl/obszary-tematyczne/ludnosc/ludnosc/ludnosc-stan-i-struktura-w-przekroju-terytorialnym-stan-w-dniu-30-06-2019,6,26.html.

[5] D. Blystone, *The Story of Instagram: The Rise of the 1 Photo-Sharing Application*, Dostęp zdalny (11.08.2020): https://www.investopedia.com/articles/investing/102615/story-instagram-rise-1-photo0sharing-app.asp.

[6] S. Aslam, *Instagram by the Numbers: Stats, Demographics Fun Facts*, Dostęp zdalny (11.08.2020): https://www.omnicoreagency.com/instagram-statistics/.

[7] *Popularna aplikacja dla Instagramu wykradała hasła użytkowników*, Dostęp zdalny (11.08.2020): https://www.komputerswiat.pl/aktualnosci/aplikacje/popularna-aplikacja-dla-instagramu-wykradala-hasla-uzytkownikow/d30w4d3.

[8] HypeAudtior, Dostęp zdalny (12.08.2020): https://hypeauditor.com/.

[9] ——, *Pricing*, Dostęp zdalny (12.08.2020): https://hypeauditor.com/pricing/.

[10] G. W. Sylwia Czubkowska, *Brand24 stracił 8 proc. klientów w wyniku blokady Facebooka*, Dostęp zdalny (13.08.2020): https://wyborcza.pl/7,156282,25603398,brand24-stracil-8-proc-klientow-w-wyniku-blokady-facebooka.html.

[11] *JavaScript*, Dostęp zdalny (13.08.2020): https://developer.mozilla.org/pl/docs/Web/JavaScript.

[12] StackOverflow, *Developer Survey Results 2019*, Dostęp zdalny (13.08.2020): https://insights.stackoverflow.com/survey/2019#technology.

[13] *Nivo Rocks Library*, Dostęp zdalny (14.08.2020): https://nivo.rocks/.

[14] *Axios Library*, Dostęp zdalny (14.08.2020): https://github.com/axios/axios.

[15] *Fetch*, Dostęp zdalny (14.08.2020): https://javascript.info/fetch.

[16] *Swagger UI*, Dostęp zdalny (14.08.2020): https://swagger.io/.

[17] *Facebook Graph API*, Dostęp zdalny (15.08.2020): https://developers.facebook.com/docs/graph-api/.

[18] *Graph QL*, Dostęp zdalny (15.08.2020): https://graphql.org/.

[19] *Instagram Graph API*, Dostęp zdalny (15.08.2020): https://developers.facebook.com/docs/instagram-api/.

[20] *Instagram Basic Display API*, Dostęp zdalny (15.08.2020): https://developers.facebook.com/docs/instagram-basic-display-api/.

[21] *Google Maps Access Token*, Dostęp zdalny (16.08.2020): https://developers.google.com/maps/documentation/javascript/get-api-key.

[22] *Postman*, Dostęp zdalny (16.08.2020): https://www.postman.com/.

[23] *List of Instagram Banned Hashtags [Updated 2020]*, Dostęp zdalny (17.08.2020): https://ideadeco.co/2019/05/14/list-of-instagram-banned-hashtags-updated-2020/.

[24] ALABASTERFOX, *Instagram: co to jest shadowban?*, Dostęp zdalny (20.08.2020): https://www.alabasterfox.pl/instagram-co-to-jest-shadowban/.

Spis rysunków

Spis tabel