

Inteligentna platforma monitoringu

autorzy:

Dominik Bober, Szymon Duda, Adam Klekowski, Przemysław Ziaja



AGH University of Science and Technology

Cracow

2021

Copyright © Dominik Bober, Szymon Duda, Adam Klekowski, Przemysław Ziaja, 2021. All rights reserved.

Spis treści

	Page
ROZDZIAŁ 1. Wprowadzenie	1
1.1 Zarys projektu	1
1.2 Funkcjonalności	1
1.3 Wymagania niefunkcjonalne	2
1.3.1 Wymagania niefunkcjonalne w obszarze funkcjonalności	2
1.3.2 Wymagania niefunkcjonalne w obszarze użyteczności i kompatybilności	2
1.3.3 Wymagania niefunkcjonalne w obszarze wydajności	2
ROZDZIAŁ 2. Projekt systemu	3
2.1 Architektura systemu	3
2.1.1 Opis modułów i interfejsów	4
2.1.2 Opis serwisu Kafka (komunikacja streaming-scrapera < – > car-recognizer)	4
2.1.3 Opis bazy danych MongoDB	5
2.1.4 Opis bazy danych mySQLDB	8
2.1.5 Opis serwisu Prometheus	8
2.2 Opis serwisu Streaming Scraper	12
2.3 Opis serwisu Car Recognizer	15
2.3.1 Model	15
2.3.2 Nasz classifier	15
2.4 Testy	17
2.4.1 Statyczna analiza kodu	17
2.4.2 Testy modułów	18
ROZDZIAŁ 3. Opis systemu	20
3.1 Cel	20
3.2 Granice systemu	20
3.3 Lista możliwości	20
3.3.1 Diagramy sekwencji	20
3.3.2 Scenariusze akcji	20
ROZDZIAŁ 4. Funkcjonalność	30
4.1 Spis stron	30
4.2 Logowanie i rejestracja	30
4.3 Strona główna, profilu i błędu 404	32
4.4 Strona przeglądania	36
ROZDZIAŁ 5. Lista narzędzi używanych przy realizacji projektu	40

ROZDZIAŁ 6. Licencje	41
ROZDZIAŁ 7. Bibliografia	43

ROZDZIAŁ 1. Wprowadzenie

1.1 Zarys projektu

Projekt "Wideo Analizer" to skalowalny system, który może pobierać oraz przechowywać klatki z filmów wideo z wielu różnych źródeł, analizować je w czasie rzeczywistym oraz przechowywać wyniki analizy oraz klatki przez dowolny okres czasu.

System podzielony jest na wiele mikroserwisów co wpływa na bezawaryjność systemu. W łatwy sposób można powiększyć funkcjonalność systemu przez dopisanie kolejnego serwisu. Ponadto w przypadku awarii nie lub rozbudowy nie trzeba resetować całego systemu, wystarczy zresetować wymagany komponent.

Program umożliwia założenie konta, powalajacego śledzenie streamów internetowych i analizowanie ich zawartości pod katem wystepujacych na nich osób i pojazdów. Użytkownik może oglądać przechwycone klatki streamu wideo i odczytywać zebrane informacje. System jest w stanie inteligentnie odnajdywać obiekty na obrazie, a także dzięki wytrenowanemu modelowi rozpoznawać i klasyfikować je. Potrafi rozpoznać na obrazie osoby i pojazdy a także określić ich pozycje na obrazie. W przypadku rozpoznania auta, użytkowi dostanie informacje o marce oraz modelu pojazdu.

1.2 Funkcjonalności

- Monitorowane obszaru po którym poruszaja sie pojazdy mechaniczne oraz osoby poprzez internetowy stream wideo.
- Zbieranie danych opisujacych pojazdy poruszajace sie po obserowanym obszarze.
- Wyszukiwanie obrazów na podstawie zawartości obrazu

1.3 Wymagania niefunkcjonalne

1.3.1 Wymagania niefunkcjonalne w obszarze funkcjonalności

- Aplikacja działa jako program na urządzeniach stacjonarnych lub na wielu klustrach serwerowych.
- Aplikacja musi być łatwo skalowalna i prosta w rozszerzaniu.
- Aplikacja jest dostępna w przeglądarkach internetowych

1.3.2 Wymagania niefunkcjonalne w obszarze użyteczności i kompatybilności

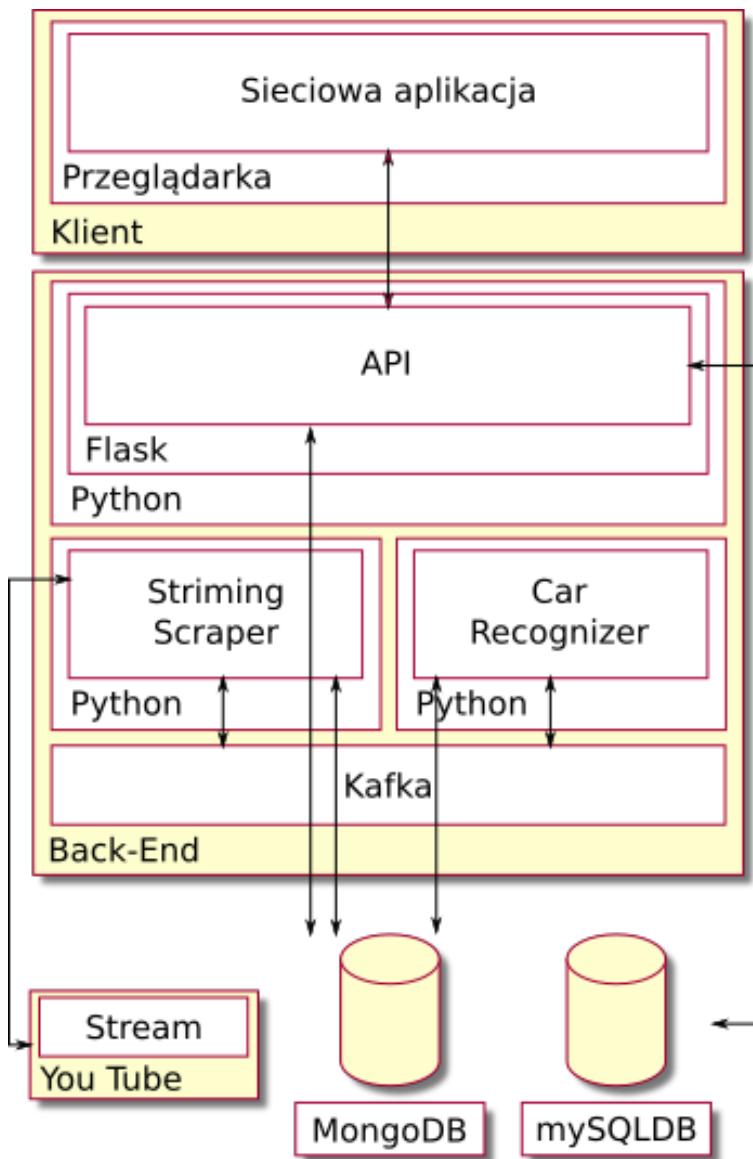
- Aplikacja potrafi obsłużyć różne formaty wideo (te najczęściej występujące)

1.3.3 Wymagania niefunkcjonalne w obszarze wydajności

- Aplikacja jest w stanie rejestrować przynajmniej jeden obraz danych monitorowanego obszaru w czasie 10 sekund
- Komunikacja pomiędzy mikroserwisami nie powinna zajmować wiecej czasu niż jedna sekunda

ROZDZIAŁ 2. Projekt systemu

2.1 Architektura systemu



2.1.1 Opis modułów i interfejsów

- Flask - mikro framework aplikacji webowych napisany w jezyku Python, wykorzystywany w celu implementacji API.
- Streaming Scraper - napisany przez nas w jezyku Python mikroserwis służacy do przechwytywania Klatek wideo i zapisywaniu obrazów, wykorzystuje bibliotekę OpenCV.
- Car Recognizer - napisany przez nas w jezyku Python mikroserwis, przy wykorzystaniu Tensorflow i YOLOv3 analizuje pliki obrazów, rozpoznane elementy zapisuje do bazy danych.
 - Tensorflow - kompleksowa platforma open source do uczenia maszynowego.
 - YOLOv3 - system pozwalajacy na użycie wytrenowanych modeli do analizy obrazów.
- Kafka - broker służacy do przesłania i odbioru rejestrowanych informacji. Pozwala na komunikacje pomiędzy serwisami i zmniejsza ruch w sieci.
- Stream - stream wideo deklarowany przez użytkownika.
- MongoDB - obiektowa baza danych. Przechowuje zdjecia i informacje przechwycone ze streamu wideo.
- mySQLDB - strukturalna baza danych. Jest baza użytkowników.
- prometheus - system do monitorowania api.

2.1.2 Opis serwisu Kafka (komunikacja streaming-scraper < – > car-recognizer)

- Wykorzystywany jest broker Apache Kafka, który zapewnia wysoka przepustowość przesyłania informacji
- Serwis przesyła dane w formacie binarnym
- Serwis umożliwia stworzenie w łatwy sposób kanałów, które odseparowywają od siebie wiadomości

W podstawowej wersji Apache Kafka odbiera wiadomości od serwisu Streaming Scraper, który to publikuje wiadomości na zadanym kanale. W tym przypadku serwis Streaming Scraper jest producentem. Natomiast serwis car-recognizer jest konsumentem, tzn. obserwuje on zadanego kanał, a w momencie pojawienia się nowej wiadomości, która dostarcza Apache Kafka, rozpakowywuje wiadomość oraz ją przetwarza.

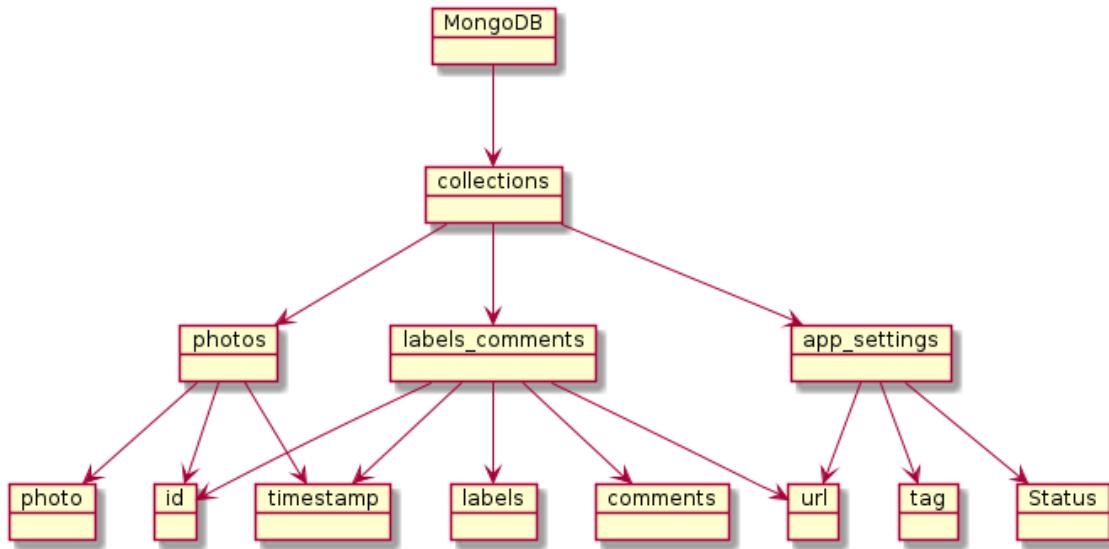
Zdecydowaliśmy się na wybór Kafki z powodu wysokiej przepustowości. Dzięki temu system komunikacji pomiędzy kontenerami nie będzie w przyszłości wąskim gardłem. Ponadto można w łatwy sposób utworzyć kanał notyfikujący o zadanym zdarzeniu.

2.1.3 Opis bazy danych MongoDB

- Wykorzystywana baza danych to MongoDB, jest ona typu obiektowego
- Przechowuje dane w formacie JSON
- Baza danych jest podzielona na 3 kolekcje
 - photos - rekordy zawierają
 - * id - identyfikator rekordu
 - * timestamp - czas scrapowania zdjęcia
 - * photo - zdjęcie
 - labels_comments - rekordy zawierają
 - * id - identyfikator rekordu
 - * timestamp - czas scrapowania zdjęcia
 - * labels - lista wykrytych obiektów na zdjęciu, obiekty w liście posiadają atrybuty takie jak
 - współrzędna x obiektu na zdjęciu
 - współrzędna y obiektu na zdjęciu
 - szerokość obiektu

- wysokość obiektu
- nazwa obiektu
- app_settings - rekordy zawierają
 - * url - adres url streamu
 - * tag - opcjonalny komentarz
 - * user - użytkownik, który dodał stream
 - * Status - obecność rekordu świadczy o prawidłowym scrapowaniu

Figure 2.1 Relacje pomiędzy kolekcjami w MongoDB



Ze względu na specyfikę problemu MongoDB posiada ten atut, że rozmiar rekordu domyślnie jest ograniczony, ale to ograniczenie można zwiększyć. Domyślny limit to około 1 megabajta. Ponadto MongoDB jest w łatwy sposób skalowalny, można uruchomić wiele instancji jednocześnie przy wykorzystaniu kontenerów. Jest to o tyle ważne, że można uruchomić kontenery na wielu serwerach przez co pamięć jest nieograniczona.

Dzieki temu zwiększymy również bezpieczeństwo dostępu do danych. W przypadku kiedy jeden z node'ów zostanie odłączonych od sieci nie wpłynie to w znaczący sposób na działanie całej bazy danych. MongoDB wymaga, aby jeden node miał funkcje master. Nawet w przypadku

odłączenia master node'a pozostałe instancje negocjują pomiędzy sobą, który node przejmie ta funkcję. Przerwa w dostępie do danych nie wynosi wiecej niż minuta.

Dla zapewnienia bezpieczeństwa pod względem dostępu do danych MongoDB udostępnia mechanizm tworzenia użytkowników oraz przypisywania im ról. W serwisie wykorzystujemy ta opcje, każde połaczenie do bazy danych jest zabezpieczone.

2.1.4 Opis bazy danych mySQLDB

- Wykorzystywana baza danych to MongoDB, jest ona typu relacyjnego.
- Została zbudowana przy użyciu Flaska.
- Przechowuje informacje o wszystkich zarejestrowanych użytkownikach:
 - id (PRIMARY KEY)
 - username
 - password
 - email

2.1.5 Opis serwisu Prometheus

Prometheus to oprogramowanie służace do monitorowania serwerów udostepniajacy zasoby w internecie. Aby udostepnić serwisowi dane została wykorzystana biblioteka prometheus_flask_exporter. W sposób automatyczny udostepnia ona dane o zużyciu zasobów przez serwer Flask oraz umożliwia tworzenie własnych metryk. W sposób automatyczny zostały zaimplementowane metryki zliczajace odwiedzenie każdej strony na serwerze. Jednakże możliwości sa nieograniczone, można stworzyć dowolne metryki, które bedą zliczały dowolne dane. Serwis Prometheus zbiera te metryki poprzez wywołanie podstrony /metrics, gdzie składowane sa wszystkie metryki. Nastepnie na stronie serwisu Prometheus udostepniane sa informacje m.in. o tym, z jakich stron internetowych dane sa pobierane, czy serwis jest aktywny czy nie, oraz można tworzyć wykresy pobranych danych aby zobaczyć jaka jest dynamika serwisu.

Figure 2.2 Przykładowe metryki serwisu Prometheus

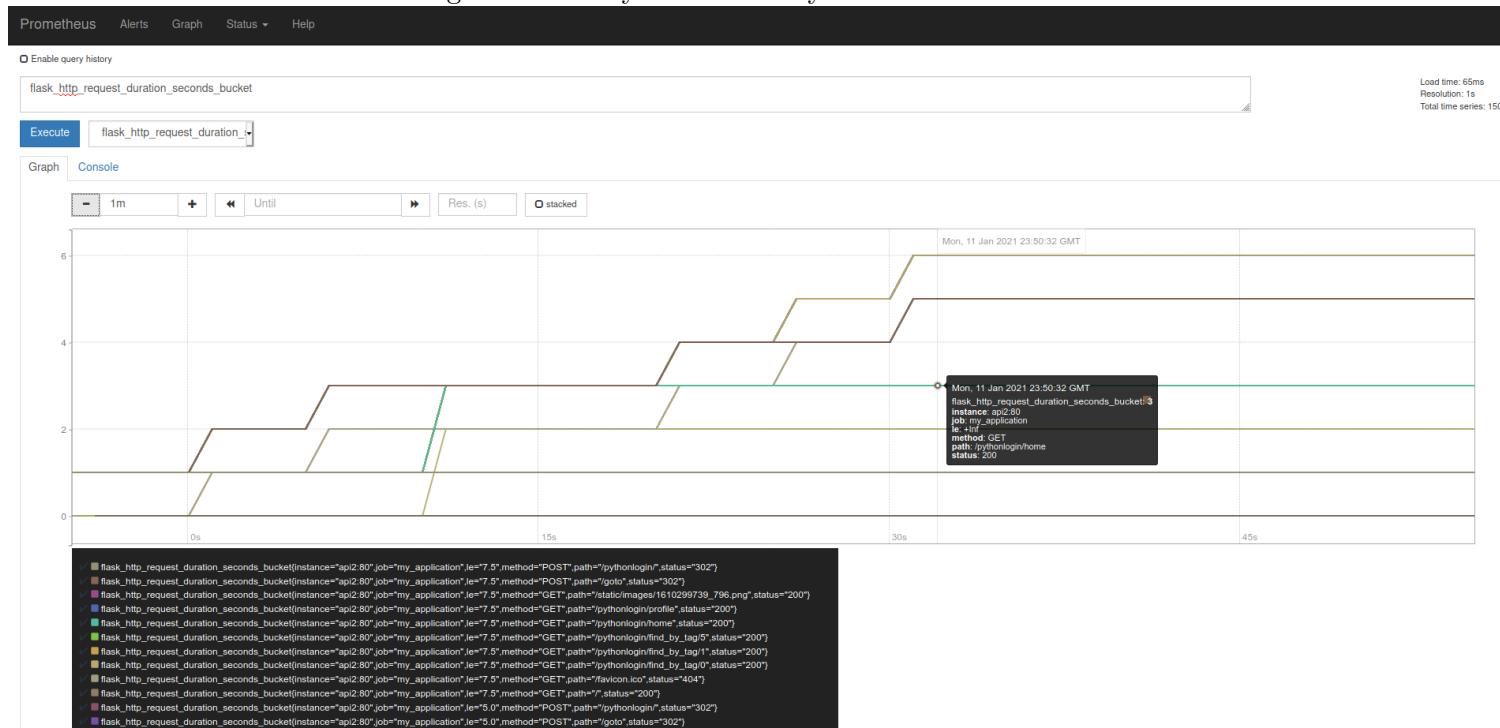


Figure 2.3 Przykładowe metryki serwisu Prometheus

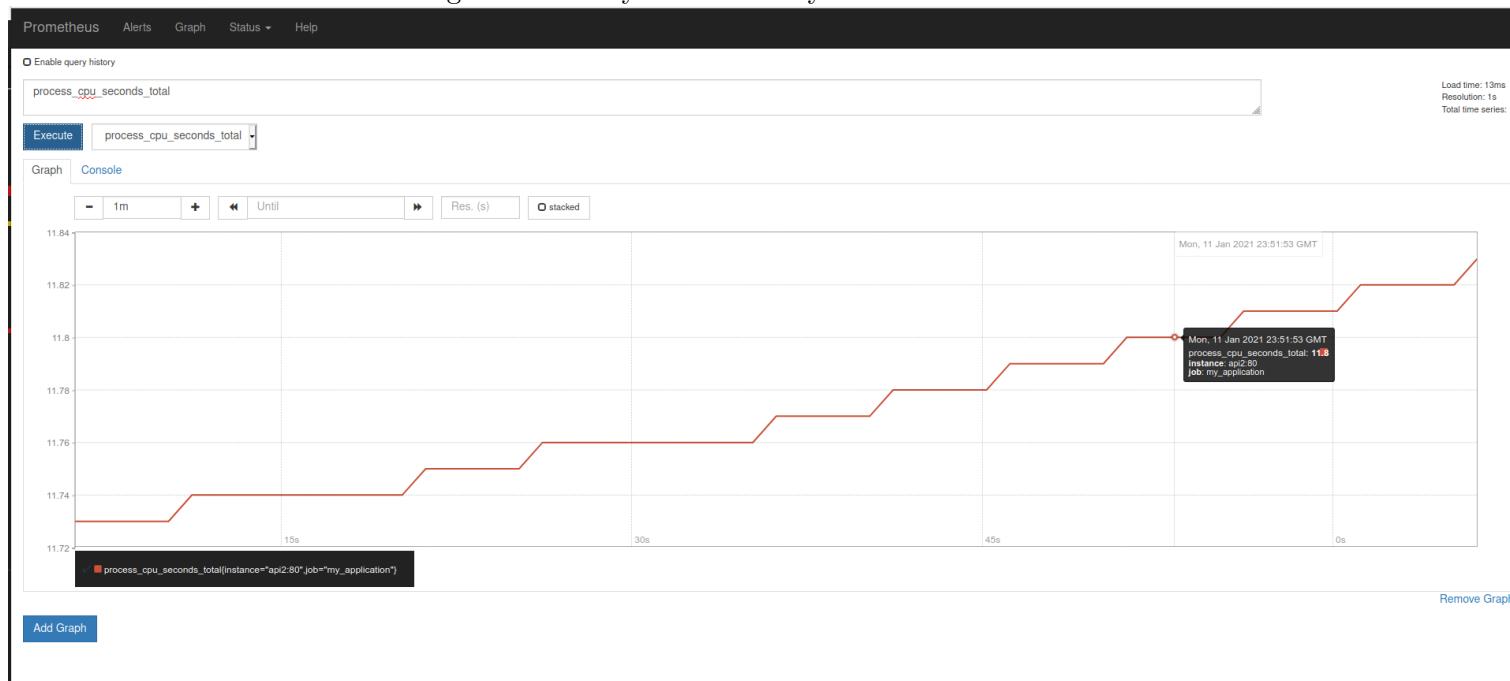
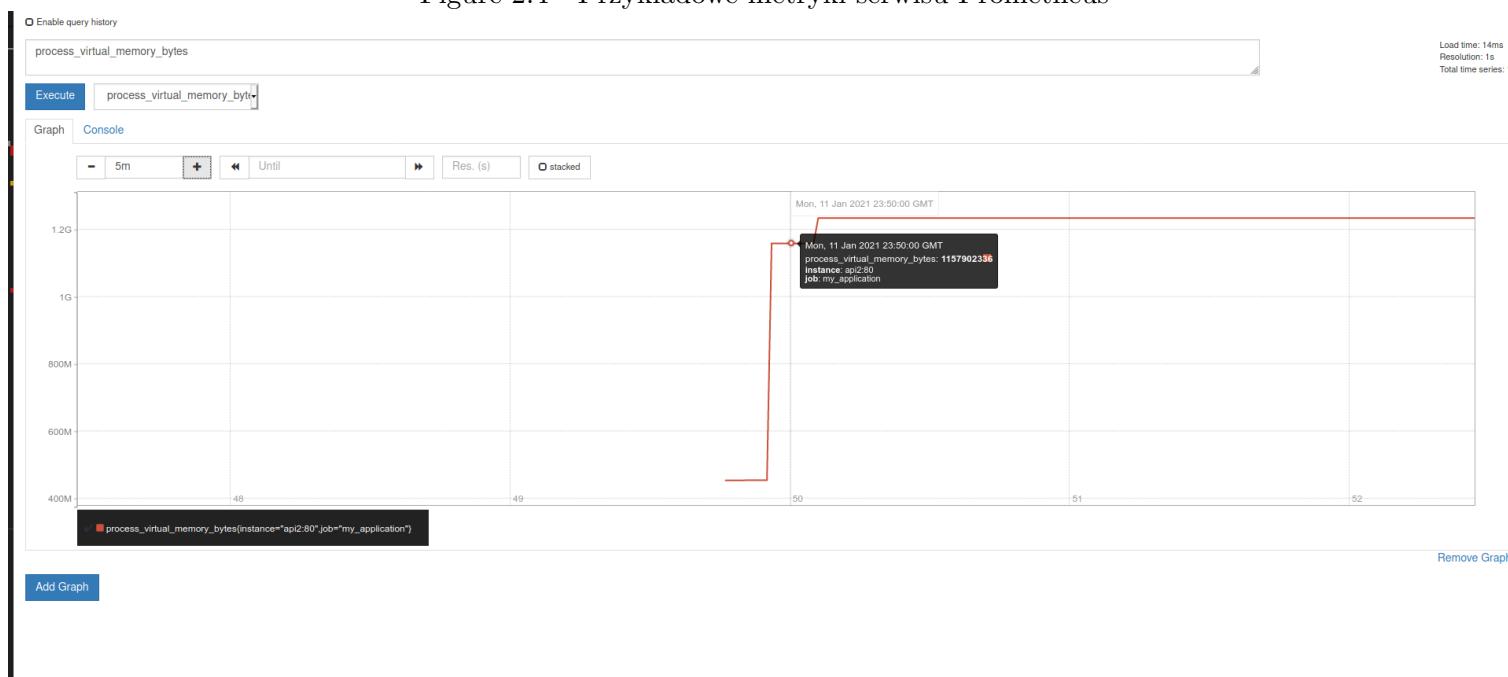


Figure 2.4 Przykładowe metryki serwisu Prometheus



2.2 Opis serwisu Streaming Scraper

Serwis StreamingScraper wykorzystuje w głównej mierze bibliotekę opencv-python. Przy jej pomocy są przechwytywane klatki z wideo.

Na początku serwis uruchamia się przy pomocy zapisanej konfiguracji. Nawiązuje połączenie do bazy danych (MongoDB) oraz notyfikatora (kafka). Połączenie do bazy danych wygląda w następujący sposób

- przy pomocy zapisanej lokalizacji nawiązywane jest połączenie do serwisu MongoDB
- definiowana jest baza danych
- następuje określenie mechanizmu autoryzacji
- następuje logowanie przy pomocy loginu i hasła

Wszystkie te operacje odbywają się przy inicjalizacji MongoClient().

Następnie inicjalizowany jest notyfikator. Notyfikator informuje inne serwisy o przechwyceniu klatki. W tym przypadku jest to zwyczajne podanie adresu serwisu, w którym uruchomiona jest kafka. Przy użyciu notyfikatora ważne jest aby podać temat, na który ma zostać wysłana informacja o przechwyceniu klatki. Wszystkie serwisy wykorzystujące kafka wykorzystują ten sam temat, który został zapisany w ich strukturach.

Ostatnim punktem inicjalizacji klasy jest pobranie konfiguracji samego scrapera, która jest zapisana w pliku obok kodu źródłowego. Plik zawiera jedynie 2 informacje:

- nazwa kolekcji do której mają być zapisywane zdjęcia
- odstęp czasu pomiędzy scrapowaniem klatek w zakresie [10 sek; inf)

Serwis wykorzystuje menager, który jest uruchamiany po zainicjalizowaniu obiektu. Serwis działa na wielu wątkach, co spowodowało konieczność wykorzystania menagera.

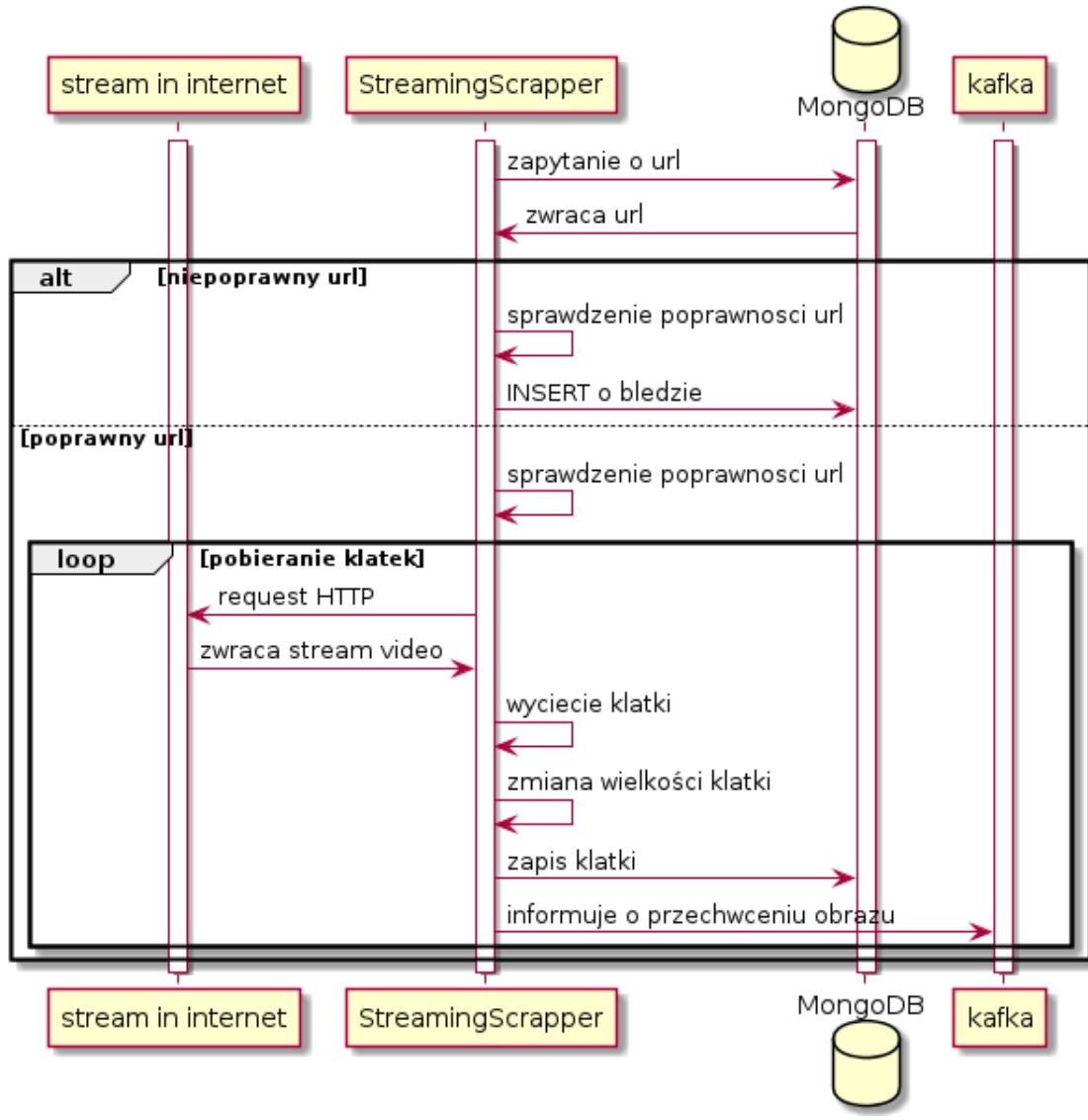
Na początku tworzony jest obiekt lock, który zapobiega wykonywaniu operacji krytycznych na wielu wątkach takich jak:

- zapis do bazy danych
- notyfikacja

Nastepnie w petli, która wykonuje się co 10 sekund wysyła zapytanie do bazy danych o informacje o obecnie scrapowanych streamach. Informacje te znajdują się w kolekcji app_settings. Każdy musi zawierać adres url streamu. Opcjonalnie może zawierać dodatkowe pola. W api dodawany jest także nazwa użytkownika, który dodał stream. Każdy otrzymany adres url jest sprawdzany, czy nie jest już scrapowany. Jeżeli jest stream, który jest scrapowany, a brakuje go w bazie danych, to scrapowanie jest zatrzymywane. Następnie, jeżeli występują adresy url, które nie są scrapowane, a pojawiają się w bazie danych to testowane jest połączenie do danego url. W przypadku gdy testy wypadają pomyślnie, uruchamiany jest nowy watek który bedzie scrapował stream z danego url. Dodatkowo pole, w którym był zapisany url otrzymuje nowy atrybut "Status" z wartością "up". W przypadku gdy testy nie zakończą się powodzeniem to adres url jest pomijany, a rekord usuwany z bazy danych.

Watek, który obsługuje scrapowanie streamu jest uruchamiany w następujący sposób. Na początku jest sprawdzany adres url. Jeżeli adres odnosi się do serwisu popularnego serwisu video youtube to przy pomocy dodatkowych bibliotek stream jest poddawany dodatkowej obróbce. Jest to specyficzny przypadek, który pomijany jest przy wszystkich innych adresach url. Następnie inicjowany jest obiekt z biblioteki opencv-python, który jest odpowiedzialny za przechwytywanie klatek. Następnie w nieskończonej petli ustawionej, aby wykonywała się co podany w pliku konfiguracyjnym okres czasu, następuje przechwycenie klatki. W przypadku niepowodzenia okrażenie petli jest pomijane. Natomiast przy poprawnym przechwyceniu następuje zmiana rozmiaru klatki (również przy pomocy opencv), nadanie unikalnego id zdjęcia. Następnie tworzona jest wiadomość zawierająca id oraz czas przechwycenia zdjęcia. Następuje zablokowanie ze względu na krytyczny obszar kodu, wysłanie wiadomości poprzez notyfikator oraz zapisanie zdjęcia wraz z id i czasem do bazy danych.

Figure 2.5 Diagram życia działania serwisu Streaming Scraper



2.3 Opis serwisu Car Recognizer

2.3.1 Model

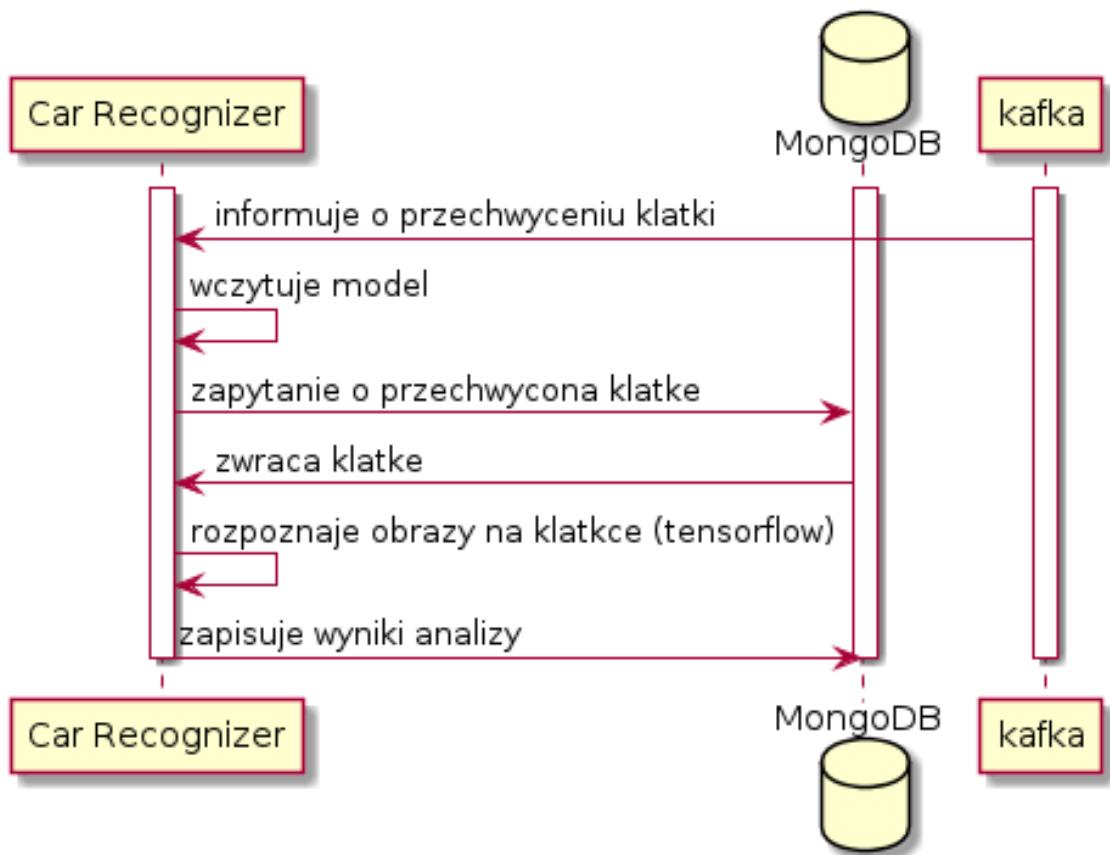
W naszym projekcie używamy modelu Spectrico. Bazując na szybkiej architekturze sieci neuronowej, model rozpoznawania marki i modelu samochodu może być łatwo integrowalny z aplikacjami, które wymagają dokładnego znakowania obrazów zawierających samochody. Posiada wysoką dokładność w różnych warunkach oświetleniowych i pod różnymi kątami.

W naszym programie używamy detektora obiektów YOLOv3 w celu znalezienia samochodu i wyznaczenia wokół niego ramki. Wykryte samochody muszą być przycięte i przeskalowane do rozmiaru 224x224 pikseli, który jest rozmiarem obrazu wejściowego klasyfikatora. Klasyfikator samochodów jest oparty na architekturze sieci neuronowej MobileNetV3. Jest on bardzo szybki i działa w czasie rzeczywistym na zwykłym komputerze klasy PC. W przypadku korzystania z wielordzeniowego CPU, możliwe jest uzyskanie bardzo dużej szybkości rozpoznawania. Ponieważ wykrywanie obiektów jest najbardziej intensywnym obliczeniowo zadaniem, możliwe jest uruchomienie detektora na GPU i dokonanie klasyfikacji na CPU. Nowoczesne karty graficzne posiadają kilkanaście razy więcej rdzeni niż CPU co w znaczny sposób przyspiesza obliczenia. Istnieje wiele sposobów na zintegrowanie klasyfikatora samochodowego z oprogramowaniem. Niektóre biblioteki uruchomieniowe, które można wykorzystać to Tensorflow, Microsoft ONNX Runtime, NVIDIA TensorRT, lekki szkielet do głębokiego uczenia się Alibaba MNN, biblioteka TFLite i zestaw narzędzi Intel OpenVINO. Możliwe jest uruchomienie klasyfikatora przy użyciu Pythona. Inna opcja jest użycie TensorFlow Serving, który jest wysokowydajnym systemem służącym do obsługi modeli uczenia maszynowego, zaprojektowanym dla środowisk produkcyjnych.

2.3.2 Nasz classifier

Wagi YOLOv3 zostały pobrane z witryny [YOLO](#). Nasz klasyfikator jest oparty na sieci MobileNet (TensorFlow backend). Zajmuje około 35 milisekund na procesorze Intel Core i5-7600 do wyliczenia pojedynczej klasyfikacji. Sam model może być akcelerowany na GPU oraz gdy

Figure 2.6 Diagram życia działania serwisu Car Recognizer



używamy procesora graficznego NVIDIA. W klasyfikatorze stosujemy jedna sieć neuronowa do pełnego obrazu. Sieć ta dzieli obraz na regiony i przewiduje pola ograniczające i prawdopodobieństwa dla każdego regionu. Te pola ograniczające są ważone przez przewidywane prawdopodobieństwa.

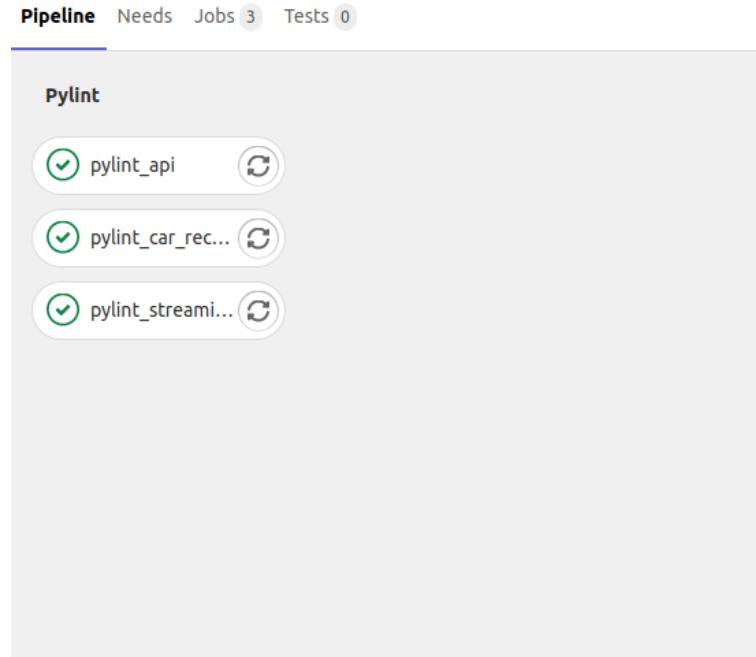
Nasz kod pobiera klatkę z wideo jako wejście, wykrywa samochody w każdej klatce za pomocą detektora obiektów YOLOv3, przycina obrazy samochodów, czyni je kwadratowymi przy zachowaniu proporcji, zmienia rozmiar wejściowy klasyfikatora i rozpoznaje markę i model każdego samochodu.

2.4 Testy

Testy i wyniki testów umieszczone w katalogu 'test/'.

2.4.1 Statyczna analiza kodu

Figure 2.7 Rezultat wykonania pipeline'ów po ostatnim commitcie



Wszystkie moduły projektu (API, streaming-scraper, car-recognizer) były sprawdzane narzędziem do statycznej analizy kody [Pylint](#). Sprawdza ono, takie rzeczy, jak zgodność kodu ze standardem PEP8, obecność docstringów, importy modułów, nieużywane zmienne, nieosiągalne fragmenty kodu, zbyt szerokie warunki w blokach try-catch, itp.

Pylint był przez nas używany w ramach pipelinów CI na gitlabie, dzięki czemu, każda zmiana była sprawdzana na bieżaco.

2.4.2 Testy modułów

Moduł	Rodzaj testów	Wykorzystywane biblioteki
car-recognizer	jednostkowe	unittest

Przykład testowanych funkcjonalności:

- rozpoznawanie obiektów na zdjciu z jednym samochodem z różnymi wartościami thresholdu oraz confidence
- rozpoznawanie wielu obiektów na zdjciu z wieloma samochodami z różnymi wartościami thresholdu oraz confidence
- rozpoznawanie na zadanym zdjciu wybranego typu obiektu

Moduł	Rodzaj testów	Wykorzystywane biblioteki
streaming-scraper	jednostkowe	unittest

Przykład testowanych funkcjonalności:

- zapis klatki do bazy danych
- połaczenie z kafka
- połaczenie do url ze streamingiem
- pobieranie klatek

Moduł	Rodzaj testów	Wykorzystywane biblioteki
API	funkcjonalne	selenium

Przykład testowanych funkcjonalności:

- logowanie z użyciem niepoprawnych danych
- rejestrowanie z użyciem istniejącej nazwy użytkownika
- rejestrowanie z użyciem niepoprawnego adresu email
- dodawanie url ze streamingiem
- szukanie zdjęć po tagu 'car'

ROZDZIAŁ 3. Opis systemu

3.1 Cel

Celem naszej aplikacji jest udostępnienie użytkownikowi możliwości monitorowania odcinka drogi lub innego obszaru pod względem pojawiających się tam pojazdów. Użytkownik jest w stanie gromadzić i przegądać dane zebrane ze streamu wideo przekazanego przez niego aplikacji. Może on wyszukiwać pojazdy, które pojawiały się w monitorowanym obszarze po modelu i marce samochodu, a także otrzymać informacje o ich pozycji na obrazie.

3.2 Granice systemu

- Użytkownik
- Stream

Jeden użytkownik jest w stanie monitorować wiele streamów.

3.3 Lista możliwości

Lista możliwości została przedstawiona w postaci diagramów sekwencji. Diagramy te reprezentują typowe działania systemu. W kolejnych sekcjach zostaną one zaprezentowane w postaci przypadków użycia i scenariuszy.

3.3.1 Diagramy sekwencji

3.3.2 Scenariusze akcji

Figure 3.1 Diagram sekwencji przykład akcji: rejestrowanie

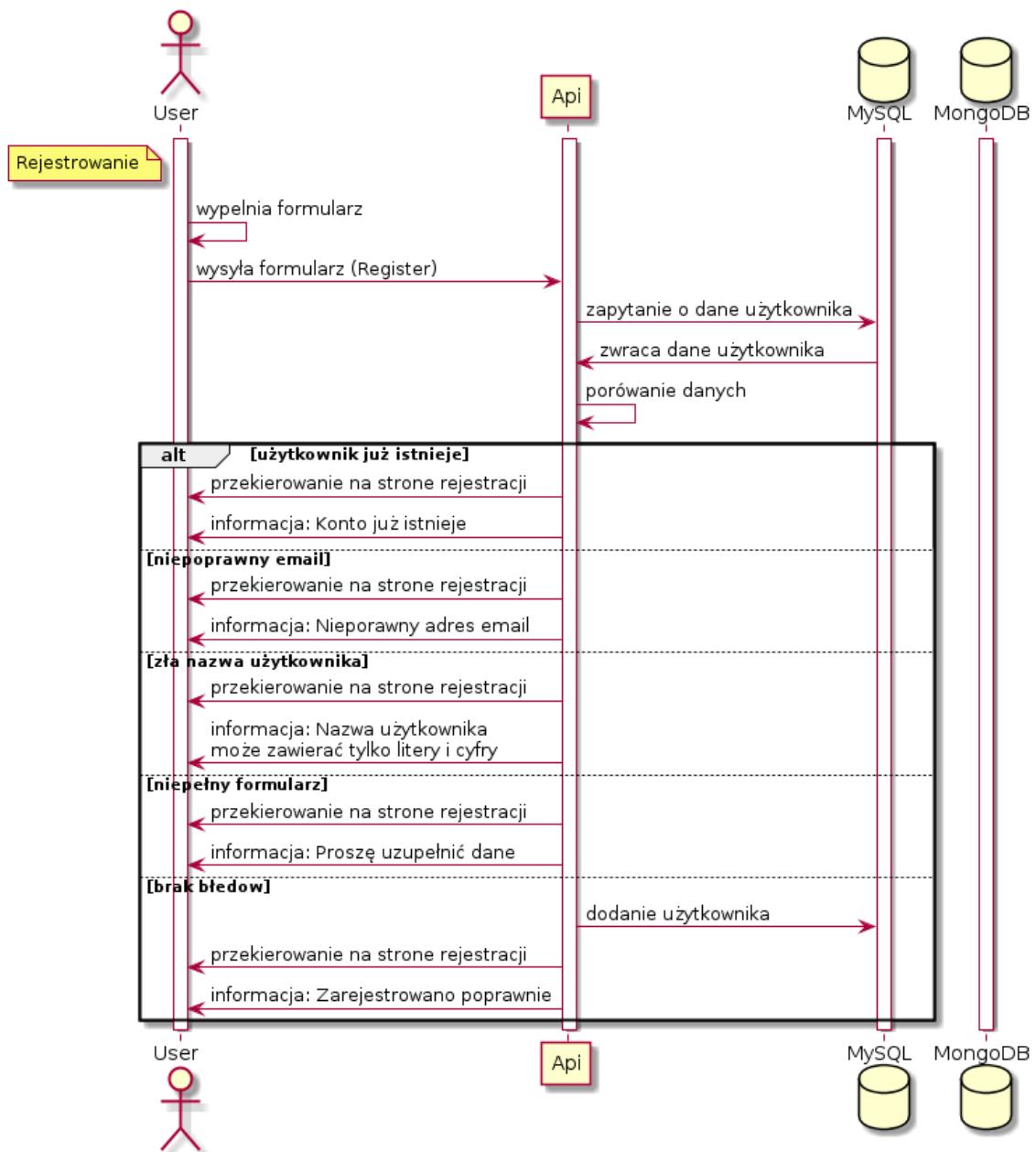


Figure 3.2 Diagram sekwencji przykład akcji: logowanie

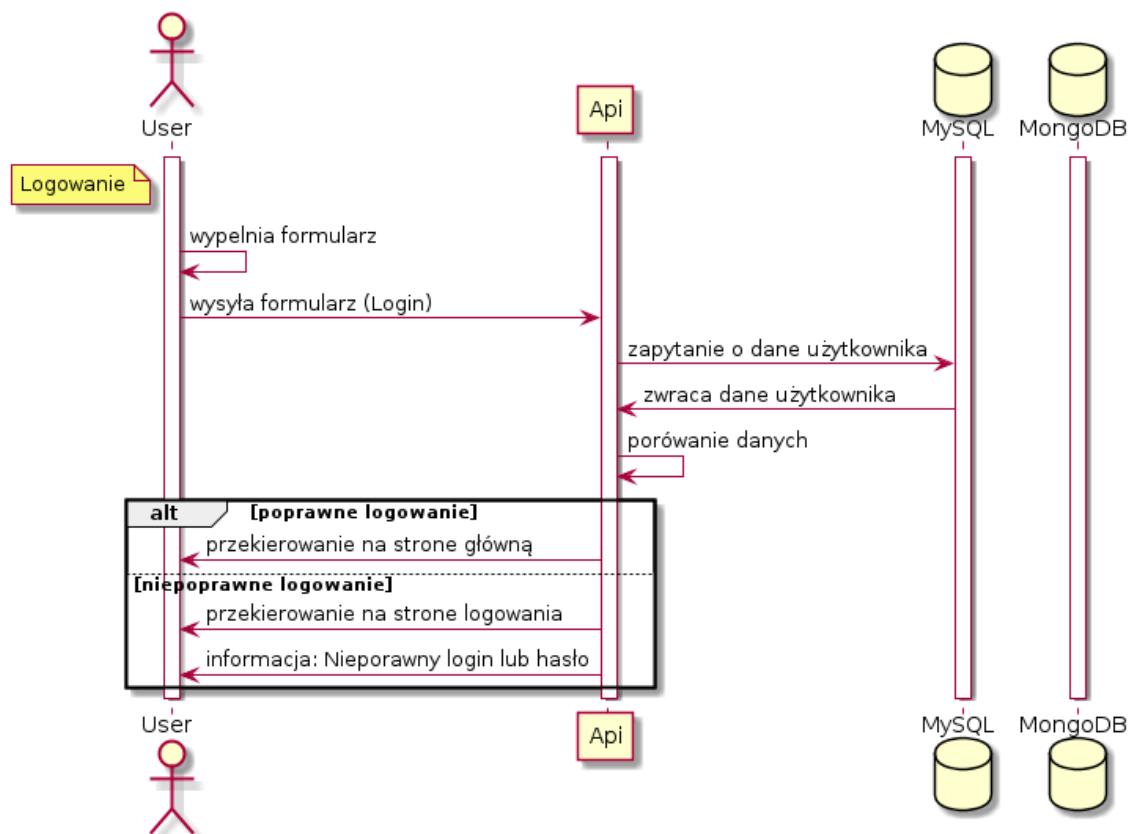


Figure 3.3 Diagram sekwencji przykład akcji: dodawanie adresu url

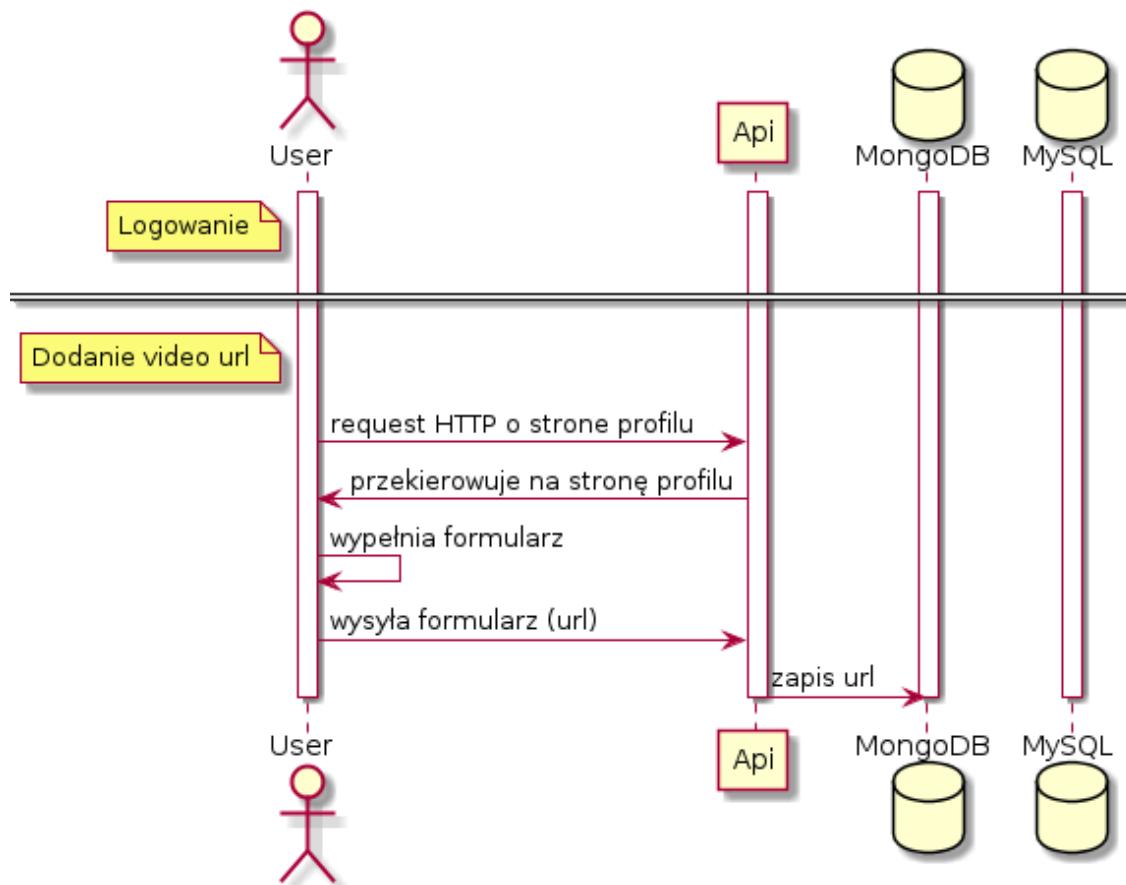


Figure 3.4 Diagram sekwencji przykład akcji: wyszukiwanie po tagach

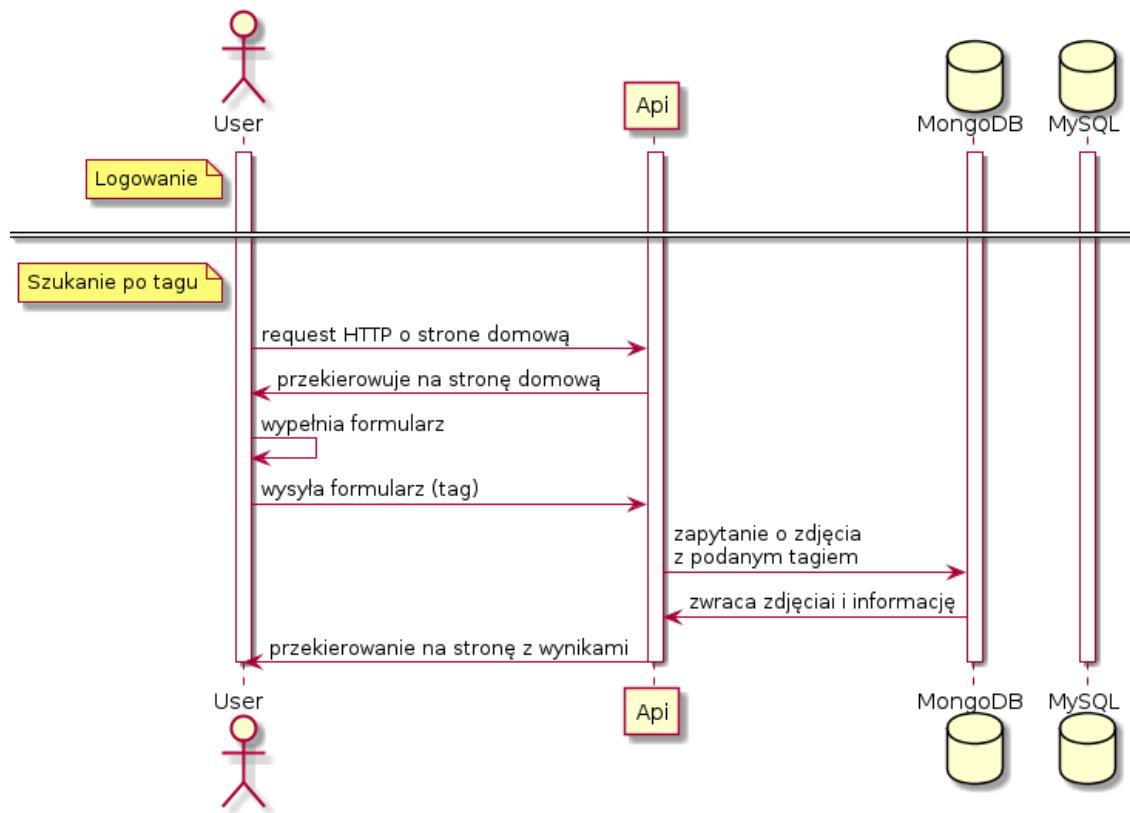


Figure 3.5 Scenariusz akcji: rejestrowanie

Przypadek użycia	Zaloguj	
Aktorzy	Klient	
Zakres	Serwis internetowy	
Kontekst użycia	Klient korzystający z systemu internetowego musi się zarejestrować	
Udziałowcy i cele	Udziałowiec	Cele
	Klient	Rejestracja konta
Zdarzenie wyzwalające	Klinięcie przycisku Register	
Warunki wstępne	Niezarejestrowany użytkownik, chcący założyć konto	
Warunki końcowe - Powodzenie	Został stworzony nowy profil użytkownika	
Warunki końcowe - Niepowodzenie	Nie został utworzony nowy profil użytkownika	
Scenariusz główny	Krok	Akcje
	1.	Pobierz dane użytkownika
	2.	Zweryfikuj dane użytkownika
	3.	Weryfikacja przebiegła pomyślnie
Scenariusz alternatywny	Krok	Akcje
	3a.	Weryfikacja nie powiodła się
	4a	Zwróć komunikat błędu

Figure 3.6 Scenariusz akcji: logowanie

Przypadek użycia	Zaloguj	
Aktorzy	Klient	
Zakres	Serwis internetowy	
Kontekst użycia	Klient korzystający z systemu internetowego musi się zalogować	
Udziałowcy i cele	Udziałowiec	Cele
	Klient	Zalogowanie do konta
Zdarzenie wyzwalające	Klinięcie przycisku Login	
Warunki wstępne	Niezalogowany użytkownik, chce się zalogować	
Warunki końcowe - Powodzenie	Użytkownik został zalogowany i może korzystać z serwisu	
Warunki końcowe - Niepowodzenie	Klient nie został zalogowany	
Scenariusz główny	Krok	Akcje
	1.	Pobierz dane użytkownika
	2.	Zweryfikuj dane użytkownika
	3.	Weryfikacja przebiegła pomyślnie
	4.	Otwórz stronę główną dla użytkownika
Scenariusz alternatywny	Krok	Akcje
	3a.	Weryfikacja nie powiodła się
	4a	Zwróć komunikat błędu

Figure 3.7 Scenariusz akcji: zmiana adresu url

Przypadek użycia	Zaloguj	
Aktorzy	Klient	
Zakres	Serwis internetowy - strona profilu	
Kontekst użycia	Klient korzystający z systemu dodaje stream do śledzenia	
Udziałowcy i cele	Udziałowiec	Cele
	Klient	Rozpoczęcie monitorowania streamu
Zdarzenie wyzwalające	Kliknięcie przycisku Add	
Warunki wstępne	Użytkownik chce dodać stream do bazy	
Warunki końcowe - Powodzenie	Stream został dodany do bazy i rozpoczęto się monitorowanie	
Warunki końcowe - Niepowodzenie	Stream nie został dodany do bazy danych	
Scenariusz główny	Krok	Akcje
	1.	Pobierz url wideo
	2.	Zweryfikuj url wideo
	3.	Weryfikacja przebiegła pomyślnie
Scenariusz alternatywny	Krok	Akcje
	3a.	Weryfikacja nie powiodła się
	4a	Zwróć komunikat błędu

Figure 3.8 Scenariusz akcji: wyszukiwanie po tagach

Przypadek użycia	Zaloguj	
Aktorzy	Klient	
Zakres	Serwis internetowy - strona domowa i przeglądania	
Kontekst użycia	Klient chce wyszukać zdjęcie po tagu	
Udziałowcy i cele	Udziałowiec	Cele
	Klient	Znalezienie zdjęcia z szukanym elementem
Zdarzenie wyzwalające	Kliknięcie przycisku Find	
Warunki wstępne	Użytkownik chce znaleźć zdjęcie zawierające wyszukiwany element	
Warunki końcowe - Powodzenie	Zdjęcie i informacje o szukanym elemencie pojawiają się na stronie	
Warunki końcowe - Niepowodzenie	Zdjęcie nie zostało znalezione, powrót na stronę główną	
Scenariusz główny	Krok	Akcje
	1.	Pobierz tag
	2.	Pobierz z bazy danych elementy zawierające tag
	3.	Przekierowanie użytkownika na stronę przeglądania
	4.	Wyświetl zdjęcie oraz informacje o szukanym elemencie
Scenariusz alternatywny	Krok	Akcje
	2a.	Brak elementów w bazie
	3a	Przekierowanie użytkownika ponownie na stronę główną

Figure 3.9 Scenariusz akcji: wyszukanie kolejnego zdjęcia

Przypadek użycia	Zaloguj	
Aktorzy	Klient	
Zakres	Serwis internetowy - strona przeglądania	
Kontekst użycia	Klient chce przejrzeć kolejne zdjęcie z wyszukiwanym elementem	
Udziałowcy i cele	Udziałowiec	Cele
	Klient	Znalezienie kolejnego zdjęcia z szukanym elementem
Zdarzenie wyzwalające	Kliknięcie przycisku Next	
Warunki wstępne	Użytkownik chce wyświetlić kolejne zdjęcie	
Warunki końcowe - Powodzenie	Zdjęcie i informacje o szukanym elemencie pojawiają się na stronie	
Warunki końcowe - Niepowodzenie	Zdjęcie nie zostało znalezione, powrót do tej samej strony ze zdjęciem	
Scenariusz główny	Krok	Akcje
	1.	Przekierowanie użytkownika na kolejną stronę przeglądania
Scenariusz alternatywny	Krok	Akcje
	2a.	Brak elementów w bazie
	3a	Przekierowanie użytkownika na tą samą stronę

ROZDZIAŁ 4. Funkcjonalność

4.1 Spis stron

Strona logowania:

`http://localhost/api/login`

Strona rejestracji:

`http://localhost/api/register`

Strona główna:

`http://localhost/api/home`

Strona profilowa:

`http://localhost/api/profile`

Strona przeglądania:

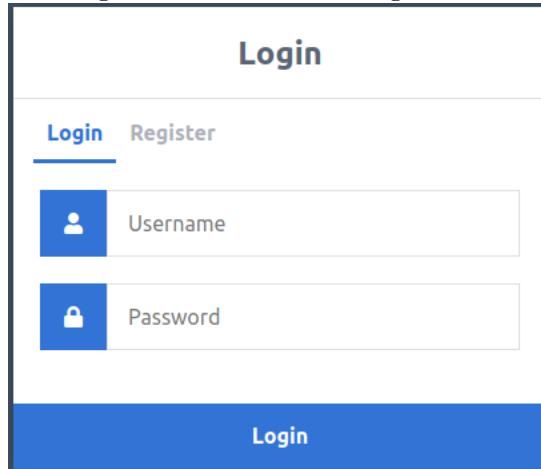
`http://localhost/api/find_by_tag/<index>`

index oznacza numer wyświetlanego zdjęcia, numery rosną w momencie przeglądania do przeodu i maleją gdy włącza się poprzednie zdjęcia.

4.2 Logowanie i rejestracja

Pierwszym ekranem który wyświetli się użytkownikowi będzie strona logowania. Użytkownik będzie mógł na niej wypełnić formularz logowania, tym samym zalogować się do serwisu na swoje konto (wcześniej utworzone). Weryfikowane są dane użytkownika, jeśli wprowadziona nazwa lub hasło nie będą poprawne, zostanie wyświetlony komunikat o ich błędności. Po podaniu poprawnych danych użytkownik przechodzi do ekranu głównego strony. Z tej strony użytkownik może również przejść na stronę rejestracji.

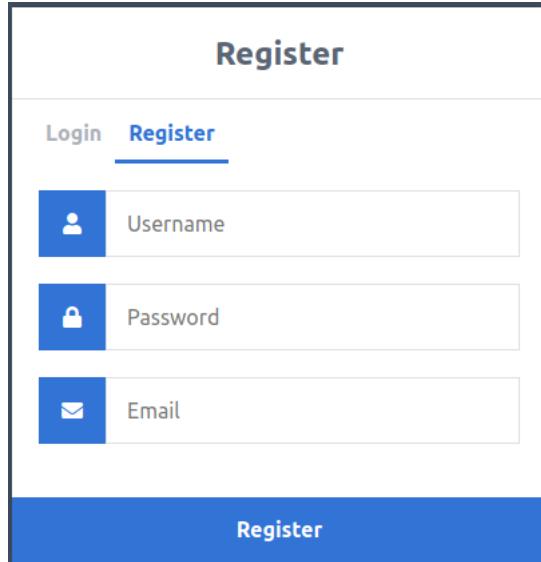
Figure 4.1 Formularz logowania



Formularz logowania z tytułem "Login". W górnym menu znajdują się linki "Login" i "Register". Poniżej znajdują się dwie pola do wpisania danych: "Username" z ikoną użytkownika i "Password" z ikoną zamka. W dolnej części znajduje się niebieski przycisk "Login".

W ekraie rejestracji użytkownik ma możliwość wypełnić formularz rejestracji i stworzyć dzięki temu nowe konto. Weryfikowana jest indywidualność nazwy użytkownika oraz format e-maila. Z tej strony użytkownik może przejść do strony logowania.

Figure 4.2 Formularz rejestracji

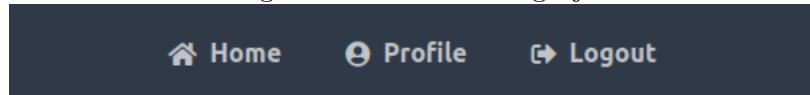


Formularz rejestracji z tytułem "Register". W górnym menu znajdują się linki "Login" i "Register", z którym jest skojarzona linia podkreślająca. Poniżej znajdują się trzy pola do wpisania danych: "Username" z ikoną użytkownika, "Password" z ikoną zamka i "Email" z ikoną koperty. W dolnej części znajduje się niebieski przycisk "Register".

4.3 Strona główna, profilu i błędu 404

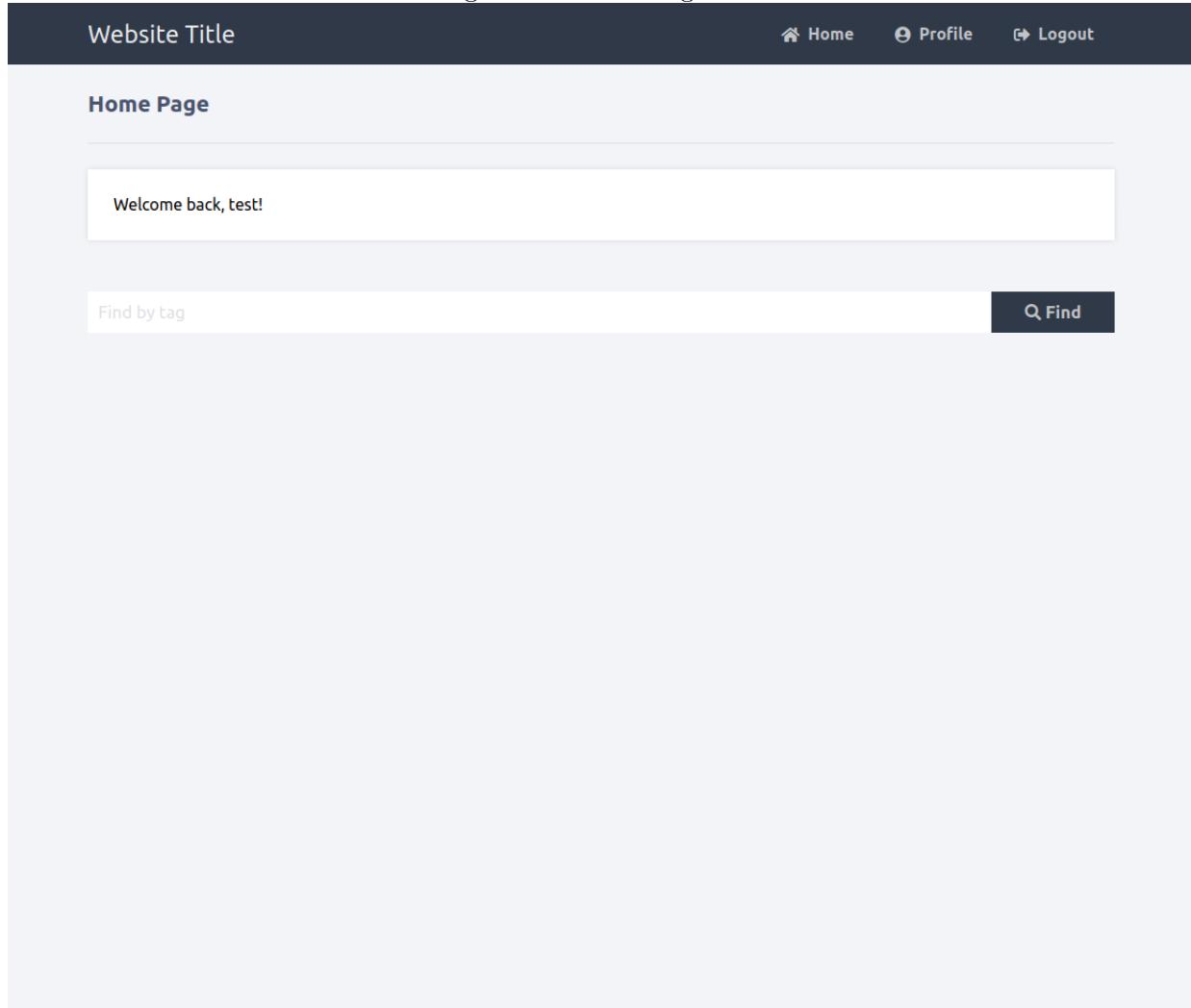
W serwisie na górze występuje pasek nawigacji, dzięki którego można przejść z dowolnej strony do strony głównej lub profilowej, albo wylogować się z konta.

Figure 4.3 Pasek nawigacji



Po zalogowaniu użytkownik odwiedza stronę główną serwisu, na ekranie wyświetla się pasek wyszukiwania, użytkownik może w nim wpisać tag szukanego obiektu (np. *person*, *car*, *Honda*, *CRV*) po kliknięciu przycisku szukania zostanie on przeniesiony na stronę przeglądania.

Figure 4.4 Strona główna



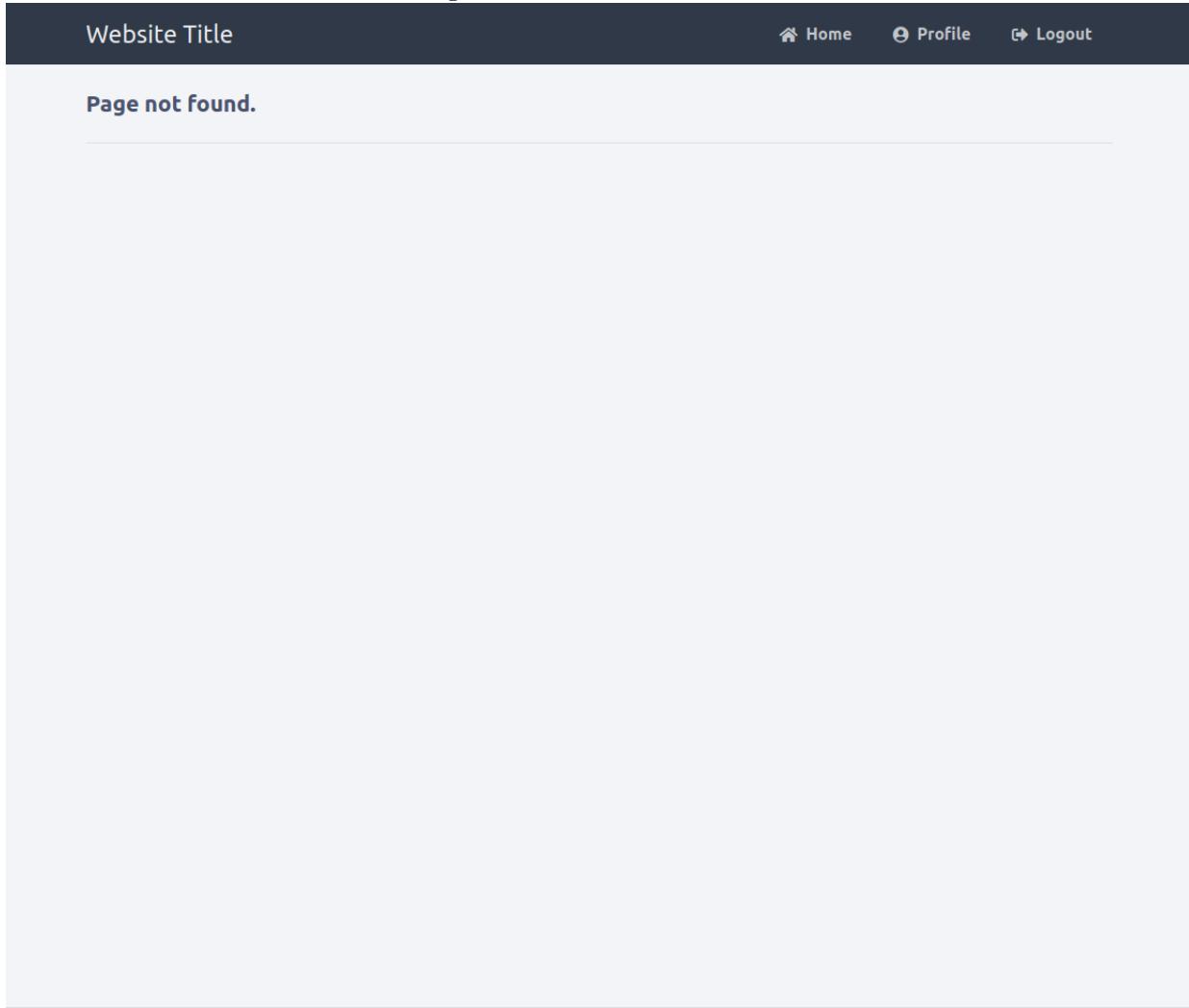
Strona profilowa posiada informacje o koncie użytkownika. Wyświetlone zostana 2 obszary, pierwszy bedzie posiadać informacje o nazwie, hasło i mail. Drugi obszar posiada natomiast liste używanych urli, które wykorzystywane sa do śledzenia.

Figure 4.5 Strona profilowa

The screenshot shows a web application interface for a 'Profile Page'. At the top, there is a dark header bar with the text 'Website Title' on the left and navigation links for 'Home', 'Profile', and 'Logout' on the right. Below the header, the main content area has a title 'Profile Page'. A large central box contains account details: 'Your account details are below:' followed by 'Username: test', 'Password: test', and 'Email: test@test.com'. Below this box is a search bar with the placeholder 'Add stream url' and a dark button with the text 'Add'. To the right of the search bar is a small icon. At the bottom of the page, another section titled 'Used urls:' displays the same account details and two URLs: <https://youtu.be/1EiC9bvVGnk> and <https://www.youtube.com/watch?v=CbcXRT8o6NY>.

W przypadku błędu serwisu lub niepoprawnego wpisania adresu url użytkownik zostanie przeniesiony do strony błędu 404

Figure 4.6 Strona błędu 404



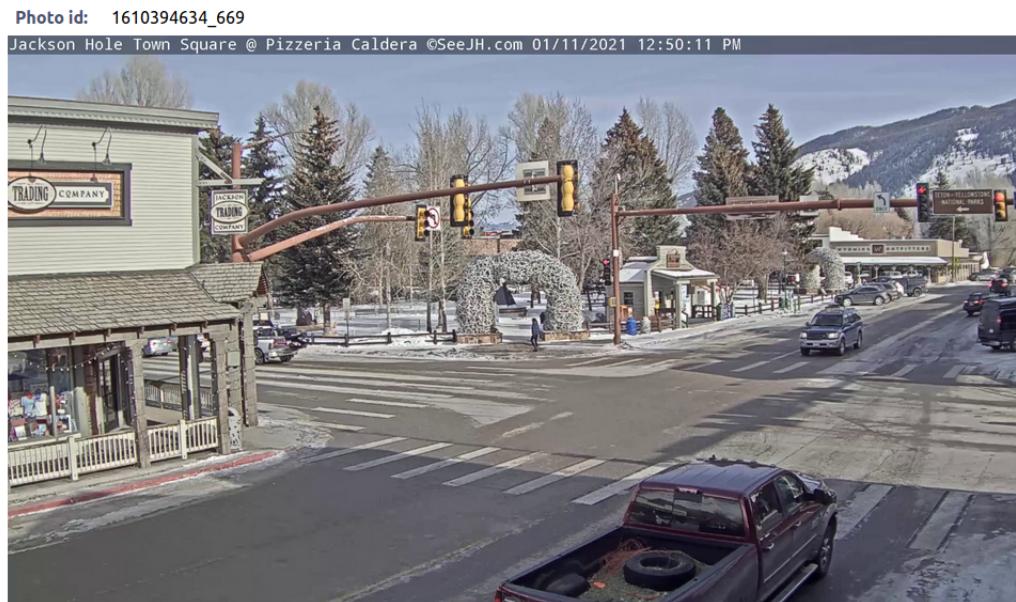
4.4 Strona przeglądania

Po wyszukaniu tagu zostaje wyświetlona strona przeglądania. Tutaj użytkownik może przeglądać obrazy które zawierają elementy pasujące do wyszukiwanego tagu. Wyszukane zdjecie wraz z dokładnymi informacjami zostają wyświetlane na ekranie

Na początku wyświetli się znalezione zdjecie z jego id w bazie danych.

Figure 4.7 Przykład znalezionej zdjęcia dla tagu *car*.

Photo founded



Poniżej wyświetli się to samo zdjecie jednak tym razem z zaznaczonymi wszystkimi odnalezionymi obiektami, które pasują do szukanego tagu oraz ich liczba.

Figure 4.8 Przykład zaznaczonych obiektów na zdjeciu odpowiadajacych tagowi *car*.

6 objects founded on image:



Na końcu zostają wyświetlane dokładne informacje o każdym obiekcie, pogrupowane są w trzy kategorie. Pierwsza zawsze zawiera nazwę przedmiotu, jeśli znalezionym obiektem jest samochód to wypisane zostają również jego marka i model. Następnie podawane są informacje położeniu i wielkości obiektu na zdjeciu, odpowiednio:

- pixel w osi x, w którym zaczyna się obiekt,
- pixel w osi y, w którym zaczyna się obiekt,
- długość obiektu,
- szerokość obiektu.

Wyświetlany zostaje również zdjecie obiektu.

Figure 4.9 Przykład wyświetlanych informacji dla obiektu, który może zostać wyszukany po tagach: *car*, *Toyota*, *Land Cruiser*.

Details

Object: Car
Brand: Toyota
Model: Land Cruiser

Szczegóły położenia obiektu na obrazie.

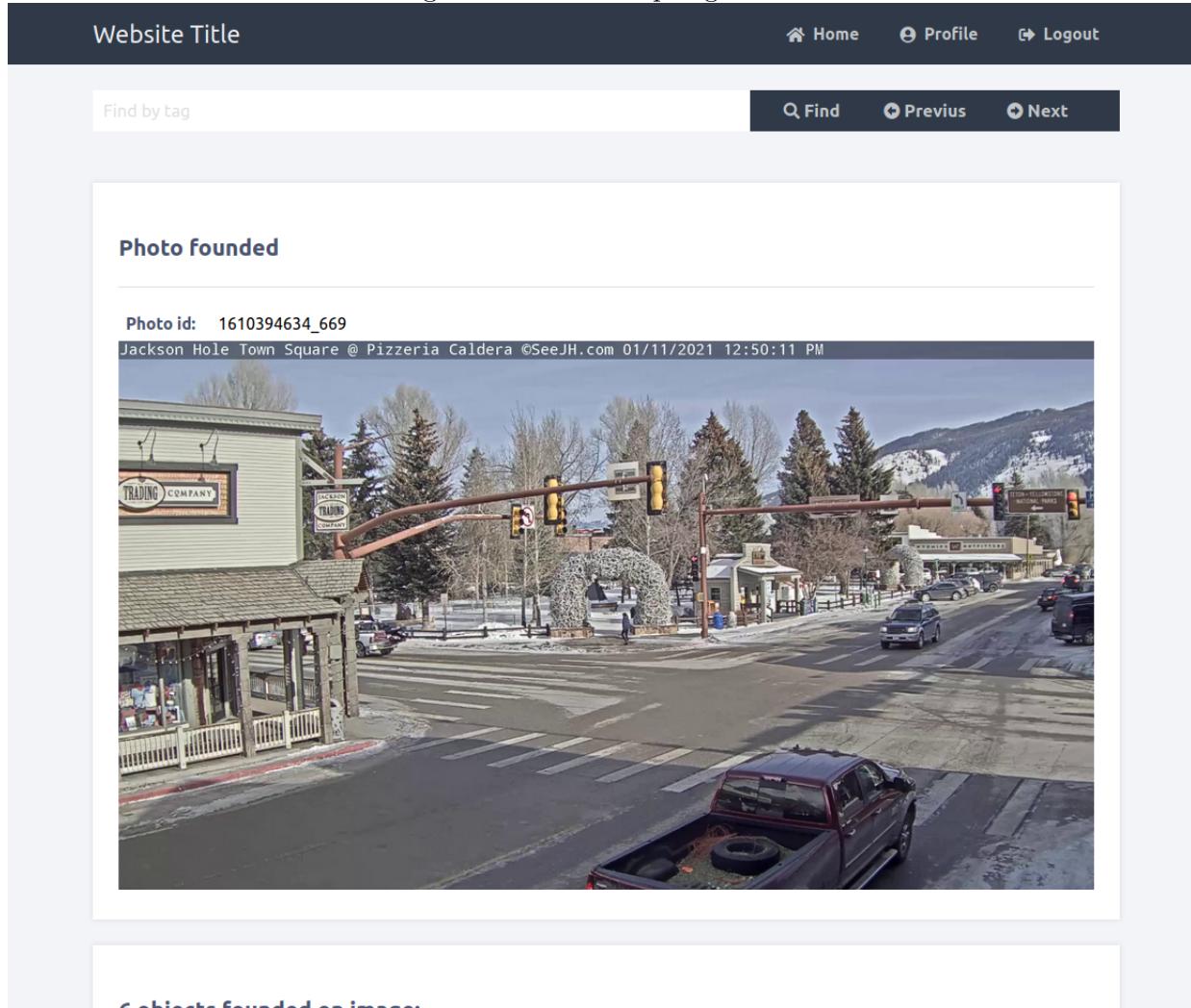
X coordinate: 1000 pixel
Y coordinate: 349 pixel
Width: 79 pixeli
Height: 55 pixeli

Photo of object



Z tego ekranu użytkownik może wyszukać nowy zestaw zdjęć pasujący do nowego tagu, bądź przechodzić do poprzedniego lub następnego znalezionej zdjęcia w bazie danych po kliknięciu odpowiedniego przycisku zostaje wyświetlona nowa strona przeglądania z nowym zdjęciem.

Figure 4.10 Strona przeglądania



ROZDZIAŁ 5. Lista narzędzi używanych przy realizacji projektu

- GitHub
- GitLab / Gitlab-CI
- Discord
- Latex - Overleaf
- Excel
- Inkscape
- PlantUMLEditor
- Docker / Docker-compose

ROZDZIAŁ 6. Licencje

- Obrazy z docker hub
 - tiangolo/uwsgi-nginx:python3.7 - Apache license
 - jjanzic/docker-python3-opencv:opencv-4.0.1 - 3-clause BSD license
 - wurstmeister/zookeeper - Apache license
 - wurstmeister/kafka - Apache license
 - prom/prometheus - Apache license
 - mongo - Apache license
 - mysql - GPLv2 license

- Python oraz używane biblioteki
 - python - Python Software Foundation License - kompatybilna z GPL
 - opencv-python - BSD-3-Clause license
 - flask - BSD-3-Clause license
 - flask_mysqldb - MIT license
 - prometheus_flask_exporter - MIT license
 - pager - Public Domain
 - pymongo - Apache license
 - pillow - HPND license
 - numpy - BSD (OSI Approved) license
 - kafka - Apache license
 - tensorflow - Apache license

- pafy - LGPLv3 license
- youtube-dl - Public Domain

ROZDZIAŁ 7. Bibliografia

- Notatki z wykładów z przedmiotu Inżynieria Oprogramowania
- Writing effective use cases, Alistair Cockburn. Boston, Addison-Wesley, 2001