POLSKO-JAPOŃSKA AKADEMIA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania, Procesów Biznesowych i Baz Danych

Piotr Skomorowski
Nr albumu 25884

Elastyczny i inteligentny system pozyskiwania danych ze stron internetowych

Praca magisterska napisana pod kierunkiem:

dr inż. Mariusz Trzaska

Warszawa, wrzesień 2023

Streszczenie

Proces ekstrakcji danych ze stron internetowych (tzw. web scrapping) wiąże się z pewnymi znaczącymi trudnościami. Jednym z głównych wyzwań jest dynamiczna natura stron internetowych, które regularnie zmieniają swoją strukturę. Narzędzia do web scrapingu muszą być stale aktualizowane, aby uwzględnić te zmiany podczas pozyskiwania danych. Taka konieczność ciągłej aktualizacji pochłania czas i zasoby. Dodatkowo istniejące narzędzia do skrapowania dostępne na rynku często wymagają zaawansowanej wiedzy technicznej, w tym znajomości selektorów CSS, XPath czy wyrażeń regularnych, co skutkuje ograniczonym ich zastosowaniem przez osoby o mniejszych kompetencjach programistycznych.

W kontekście tych wyzwań, niniejsza praca proponuje koncepcję i implementację prototypu systemu pozyskiwania danych ze stron internetowych, wykorzystującego zaawansowane techniki przetwarzania danych, w tym uczenie maszynowe. Proponowany system charakteryzuje się intuicyjnym interfejsem użytkownika, upraszczającym proces skrapowania dla osób o różnym poziomie kompetencji technicznych. Doświadczonym użytkownikom system umożliwia również wdrożenie własnych sposobów powiadomień o wynikach skrapowania oraz własnych sposobów transformacji pozyskanych danych.

W kontekście rosnącego znaczenia skrapowania danych w różnych dziedzinach, takich jak e-commerce, badania rynku czy analiza danych, przedstawiony w pracy prototyp stanowi wkład w rozwijanie dostępności i funkcjonalności technologii skrapowania.

Słowa kluczowe:

Analiza danych nieustrukturyzowanych, Pozyskiwanie danych ze stron internetowych, Strony internetowe, Web scrapping

Spis treści

1. Wstęp

W erze cyfrowej, przetwarzanie ogromnych ilości danych stało się nieodłącznym elementem naszego codziennego życia. Strony internetowe są jednym z głównych źródeł tych danych, ale ze względu na ich nieustrukturyzowaną naturę, pozyskiwanie informacji jest wyzwaniem. W odpowiedzi na to wyzwanie powstała koncepcja skrapowania danych ze stron internetowych. Skrapery[1] są narzędziami służącymi do ekstrakcji danych z nieustrukturyzowanych źródeł.

Na rynku dostępne są różne narzędzia i rozwiązania umożliwiające pozyskiwanie danych ze stron internetowych, jednak większość z nich wymaga zaawansowanej wiedzy technicznej. Dlatego istnieje potrzeba stworzenia prostego i intuicyjnego narzędzia, które umożliwi tworzenie złożonych skraperów nawet osobom nieposiadającym specjalistycznej wiedzy. Tego rodzaju narzędzie powinno zawierać ulepszenia w stosunku do obecnych standardów, oferując szereg funkcji modyfikacji pozyskanych danych, a także umożliwiając różnorodne formy powiadamiania o spełnieniu warunków skrapowania zdefiniowanych przez użytkownika w skrapowaniu cyklicznym. W dobie rosnącego zainteresowania uczeniem maszynowym i sztuczną inteligencją, taki instrument powinien również umożliwiać inteligentne funkcje modyfikacji danych, takie jak te wykorzystujące uczenie maszynowe.

Dalsza część pracy skupia się na opisie aktualnego stanu istniejących narzędzi i problematyce z nimi związaną. Przedstawiona jest również koncepcja prototypu jako propozycji rozszerzającej funkcjonalności skraperów o zaawansowane technologie, takie jak uczenie maszynowe, automatyzacja skrapowania oraz funkcje powiadomień.

1.1. Motywacja

Motywacją do wyboru tego tematu jest chęć zrozumienia i pasja do rozwiązywania problemów związanych z przetwarzaniem i analizą danych. Ze względu na dynamiczną naturę stron internetowych oraz coraz częstsze wykorzystanie nieustrukturyzowanych danych w dzisiejszym cyfrowym świecie, obszar skrapowania danych wymaga ciągłej eksploracji i innowacji. Wzrost zainteresowania technikami uczenia maszynowego, które umożliwiają skuteczne przetwarzanie i analizę tych danych, dodatkowo potęguje potrzebę rozwijania nowych, bardziej zaawansowanych narzędzi skrapowania.

Doświadczenie i zainteresowania w obszarze przetwarzania danych oraz uczenia maszynowego wpłynęły na decyzję o wyborze tego tematu. Praca daje możliwość pogłębienia tych umiejętności i zrozumienia ich praktycznych zastosowań.

1.2. Kontekst pracy

Web skraper, zwany również skraperem, to oprogramowanie zaprojektowane do automatycznego pozyskiwania danych ze stron internetowych [1]. Realizuje ono swoje zadanie przez analizę struktury HTML strony, a następnie ekstrakcję kluczowych informacji, takich jak tekst, obrazy, linki czy dane tabelaryczne. Skraper pozwala na automatyzację procesu gromadzenia danych, co jest szczególnie cenne ze względu na oszczędność czasu i wysiłku, które byłoby potrzebne do ręcznego przeglądania i kopiowania treści. Skrapery znajdują zastosowanie w wielu dziedzinach, takich jak badania naukowe, analiza rynku, monitorowanie konkurencji czy tworzenie zbiorów danych do uczenia maszynowego.

[1] *„Skraper”* to polonizacja terminu *„scraper”* z języka angielskiego, pochodzącego od czasownika *„to scrape”*, co oznacza dosłownie *"skrobać"* lub *„zdrapywać”*. W informatyce, jest używany do opisania procesu *„zdrapywania”* danych ze stron internetowych [2]. W kontekście pracy użyto tego słowa, ponieważ brak jest odpowiedniego polskiego terminu.

1.3. Kwestie prawne związane z działalnością skraperów

Skraping danych ze stron internetowych stał się integralną częścią wielu branż. Działalność ta nie jest jednak pozbawiona swoich implikacji prawnych. Ten podrozdział zagłębi się w prawne aspekty skrapowania danych.

Jednym z kluczowych aspektów jest prawo autorskie. W myśl prawa *„Przedmiotem prawa autorskiego jest każdy przejaw działalności twórczej o indywidualnym charakterze, ustalony w jakiejkolwiek postaci, niezależnie od wartości, przeznaczenia i sposobu wyrażenia (utwór)”* [3]. Oznacza to, że każda unikalna i oryginalna treść, która jest tworzona i publikowana, jest chroniona prawem autorskim. W kontekście skrapingu danych, może to obejmować tekst, obrazy, filmy, muzykę, a nawet strukturę i układ strony internetowej. Dlatego twórcy skraperów muszą być świadomi, że nie mogą pobierać i wykorzystywać dowolnych danych, które znajdą w Internecie, bez ryzyka naruszenia praw autorskich.

Kolejnym aspektem są ograniczenia prawne w stosunku do baz danych. Zgodnie z ustawą z dnia 27 lipca 2001 r. o ochronie baz danych, baza danych – *„zbiór danych lub jakichkolwiek innych materiałów i elementów zgromadzonych według określonej systematyki lub metody, indywidualnie dostępnych w jakikolwiek sposób, w tym środkami elektronicznymi, wymagający istotnego, co do jakości lub ilości, nakładu inwestycyjnego w celu sporządzenia, weryfikacji lub prezentacji jego zawartości”* [4]. W praktyce wiele stron internetowych, z których są pobierane dane, może być uznane za bazy danych. W związku z tym skraping takich stron może prowadzić do naruszenia wymienionych praw. Dodatkowo zgodnie z art 7. rozdziału 3. Dyrektywy 96/9/WE Parlamentu Europejskiego i Rady *„Niedozwolone jest powtarzające się i systematyczne pobieranie danych i/lub wtórne ich wykorzystanie w nieistotnej części, jeśli czynności te są sprzeczne z normalnym korzystaniem z tej bazy danych lub w sposób nieuzasadniony naruszają słuszne interesy producenta bazy danych”* [5]. Co można interpretować jako ograniczenia prawne w stosunku do powtarzającego się skrapowania. Innymi słowy, chociaż dane mogą być publicznie dostępne, gromadzenie ich masowo i w sposób systematyczny może zostać uznane za nielegalne, jeśli przekracza to prawa właściciela danych i szkodzi jego interesom. Pomimo tego, w myśl art. 8. tejże ustawy, jest to dozwolone, jeśli wykorzystuje się taką bazę *„w celach dydaktycznych lub badawczych, ze wskazaniem źródła, jeżeli takie korzystanie jest uzasadnione niekomercyjnym celem, dla którego wykorzystano bazę”* [5]. W ramach tej pracy skrapery będą tworzone i wykorzystywane w celach dydaktycznych/badawczych.

Ważne jest także przestrzeganie prawa ochrony danych osobowych. W procesie pozyskiwania danych ze stron internetowych skrapery mogą również zbierać dane osobowe użytkowników. Niesie to za sobą konieczność przestrzegania przepisów takich jak Ogólne Rozporządzenie o Ochronie Danych (RODO) w Unii Europejskiej [6]. Powinno uzyskać się zgodę użytkowników na zbieranie i przetwarzanie ich danych osobowych oraz przestrzegać zasad bezpiecznego przechowywania tych informacji.

Twórca nowego skrapera powinien zapoznać się również z zasadami korzystania (*ToU – Terms of Usage*) z danej strony internetowej przed przystąpieniem do pozyskiwania danych. Konieczne jest także przestrzeganie technicznych zabezpieczeń stosowanych przez strony internetowe, takich jak pliki *robots.txt*, które określają zasady indeksowania i skrapowania treści. Nie bez znaczenia jest również przestrzeganie zasad uczciwej konkurencji i etyki internetowej. Niektóre działania skraperów mogą być uznawane za nieetyczne lub nieuczciwe, szczególnie jeśli naruszają prawa innych stron internetowych lub prowadzą do nadmiernego obciążenia zasobów serwerów.

1.4. Cel pracy

Praca ta ma na celu przeprowadzenie analizy problemów związanych z procesem pozyskiwania danych ze stron internetowych oraz zaprezentowanie i omówienie praktycznego rozwiązania, które przyczyni się do usprawnienia tego procesu.

W ramach pracy przeprowadzono szczegółową analizę istniejących narzędzi do skrapowania stron internetowych w celu zidentyfikowania funkcji, które mogą być wykorzystane w nowym

narzędziu. Zwrócono również uwagę na potencjalne słabości tych narzędzi, których należy unikać podczas procesu projektowania prototypu.

Ponadto koncepcja pracy zakłada wprowadzenie usprawnień podnoszących użyteczność pozyskanych danych. Jednym z takich ulepszeń jest zastosowanie technik uczenia maszynowego w celu przekształcenia danych do bardziej użytecznej formy. Dodatkowo powinna istnieć opcja automatycznego pozyskiwania danych ze stron internetowych zgodnie z harmonogramem zdefiniowanym przez użytkownika. Kolejnym celem jest zaprezentowanie wbudowanego systemu powiadomień, który będzie informować użytkownika o spełnieniu określonych warunków przez jego narzędzie do skrapowania.

1.5. Rezultat pracy

Efektem realizacji niniejszej pracy jest kompleksowy dokument, który opisuje wyzwania i trudności związane ze skrapowaniem stron internetowych, a także analizuje istniejące rozwiązania i podejścia do tego zagadnienia. W wyniku pracy stworzono także narzędzie, które umożliwia ekstrakcję danych zarówno ze stron dynamicznych, jak i statycznych. Rozwiązanie jest dostępne w postaci rozszerzenia przeglądarek opartych na silniku Chromium[2], a jego interfejs użytkownika jest osiągalny poprzez panel narzędzi deweloperskich. Dzięki takiemu umiejscowieniu głównego interfejsu graficznego zapewnione jest łatwiejsze zastosowanie narzędzia w stosunku do stron otwartych w przeglądarce. Narzędzie pozwala dostosować proces skrapowania, definiować warunki modyfikacji danych i powiadomień, a także ustawiać harmonogram skrapowania.

Narzędzie oferuje możliwość modyfikowania pozyskanych danych przed ich filtrowaniem czy dalszym przetwarzaniem. W ramach zaawansowanych modyfikatorów danych system umożliwia wykorzystanie technik nauczania maszynowego. W aktualnej wersji są dostępne dwa modyfikatory danych tego typu. Pierwszy pozwala wykorzystać techniki przetwarzania języka naturalnego (*NLP*) w celu lematyzacji, a drugi identyfikacje obiektów na obrazach. Dzięki takim modyfikatorom użytkownicy mają możliwość poprawy jakości i użyteczności ekstrahowanych danych.

Kolejną zaletą przedstawionego narzędzia jest możliwość dodawania przez użytkowników niestandardowych modyfikatorów i sposobów powiadomień. Wprowadza to dodatkową elastyczność i rozszerzalność rozwiązania.

1.6. Organizacja pracy

Praca jest podzielona na kilka głównych sekcji, które mają na celu przedstawienie zagadnienia skrapowania stron internetowych.

Rozdział pierwszy stanowi wprowadzenie do tematyki pracy, omawiając motywację do podjęcia badania, aspekty prawne skrapowania oraz cel i rezultat pracy.

Rozdział drugi jest poświęcony analizie istniejących rozwiązań wykorzystywanych do skrapowania stron internetowych, ich ograniczeniom i zaletom.

Rozdział trzeci przedstawia propozycję prototypu, w tym jego wymagania i architekturę.

W rozdziale czwartym opisane są wykorzystane przy implementacji prototypu technologie, metodyki i języki.

W rozdziale piątym jest zaprezentowany sposób działania powstałego prototypu.

Na końcu, w rozdziale szóstym, znajduje się podsumowanie pracy i wnioski wynikające z pracy nad narzędziem oraz sugestie przyszłego kierunku jego rozwoju.

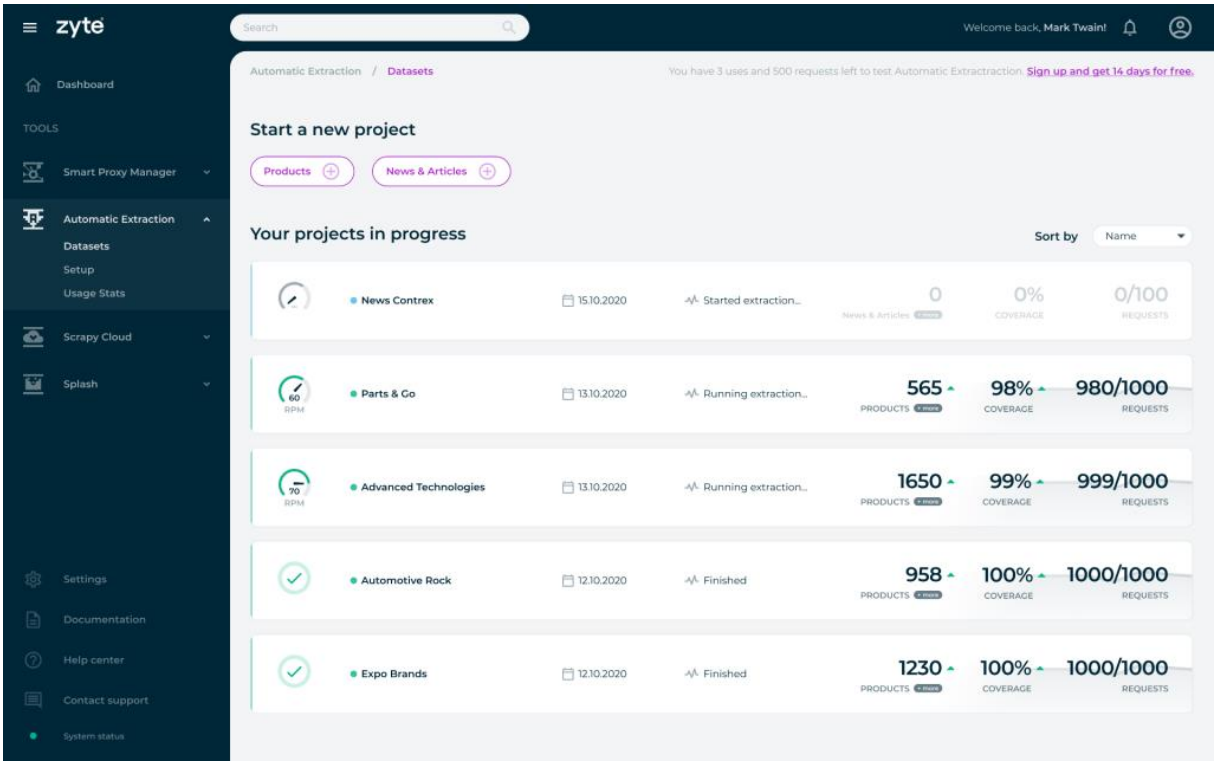
[2] Chromium to projekt otwarto źródłowej przeglądarki internetowej, które jest podstawą dla wielu popularnych przeglądarek, takich jak Google Chrome, Microsoft Edge i Opera [7].

2. Istniejące rozwiązania

W celu lepszego zrozumienia możliwości i zaprojektowania innowacyjnego systemu pozyskiwania danych ze stron internetowych, na początku warto przejrzeć i podsumować cechy istniejących rozwiązań. W tym rozdziale zostaną przedstawione istniejące narzędzia i produkty dostępne na rynku, a także zostaną wymienione ich wady i zalety.

2.1. Zyte.com

Zyte.com [8], dawniej znane jako Scrapinghub.com, to platforma i firma zajmująca się skrapowaniem stron internetowych oraz dostarczaniem narzędzi i usług związanych z pozyskiwaniem danych. Celem Zyte.com jest umożliwienie użytkownikom efektywnego i precyzyjnego pozyskiwania informacji z różnych źródeł online. Na rysunku 1. przedstawiony jest interfejs tego narzędzia.



Rysunek 1. Graficzny interfejs użytkownika strony internetowej zyte.com. Źródło: [8]

Zyte.com oferuje pięć głównych produktów, które wspierają proces skrapowania i ekstrakcji danych:

- Zyte Smart Proxy Manager (dawniej Crawlera): Jest to rozwiązanie pozwalające na zarządzanie i wykorzystywanie wielu serwerów proxy. Dzięki temu narzędziu użytkownicy mogą unikać blokad IP i utrzymywać ciągłość skrapowania nawet na stronach z ograniczeniami.
- Zyte AutoExtract: To usługa oparta na zaawansowanej technologii skrapowania, która umożliwia automatyczne pozyskiwanie strukturyzowanych danych z różnych stron internetowych. Użytkownicy mogą skonfigurować zapytania i reguły ekstrakcji, a AutoExtract wykona resztę potrzebnych operacji.

- Zyte Scrapy Cloud: Jest to platforma hostingu dla frameworku[3] Scrapy, która umożliwia użytkownikom budowanie i uruchamianie skraperów w chmurze. Scrapy Cloud umożliwia łatwe skalowanie i zarządzanie skraperami w zależności od potrzeb.
- Zyte Portia: To intuicyjne narzędzie do wizualnego skrapowania stron internetowych. Użytkownicy mogą identyfikować i opisywać elementy strony, które chcą pozyskać, bez konieczności programowania.

Wady narzędzi:

- Zyte.com jest płatnym narzędziem, co może być ograniczeniem dla osób o ograniczonym budżecie lub dla projektów niewymagających intensywnego skrapowania.
- Niektóre narzędzia, takie jak Zyte Smart Proxy Manager, mogą być bardziej skomplikowane dla użytkowników bez doświadczenia w skrapowaniu lub programowaniu.

Zalety narzędzi:

- Szeroki zakres funkcji i możliwości, umożliwiający elastyczne i precyzyjne pozyskiwanie danych z różnych źródeł online.
- Zyte.com oferuje intuicyjne interfejsy użytkownika, ułatwiające tworzenie zapytań i definiowanie reguł ekstrakcji danych.
- Narzędzia Zyte.com są zintegrowane z popularnym frameworkiem Scrapy, co umożliwia programistom wykorzystanie ich w swoich projektach.
- Zyte.com zapewnia wsparcie techniczne i regularne aktualizacje narzędzi, co przyczynia się do niezawodności i skuteczności procesu skrapowania.

[3] Framework – szkielet ułatwiający tworzenie aplikacji.

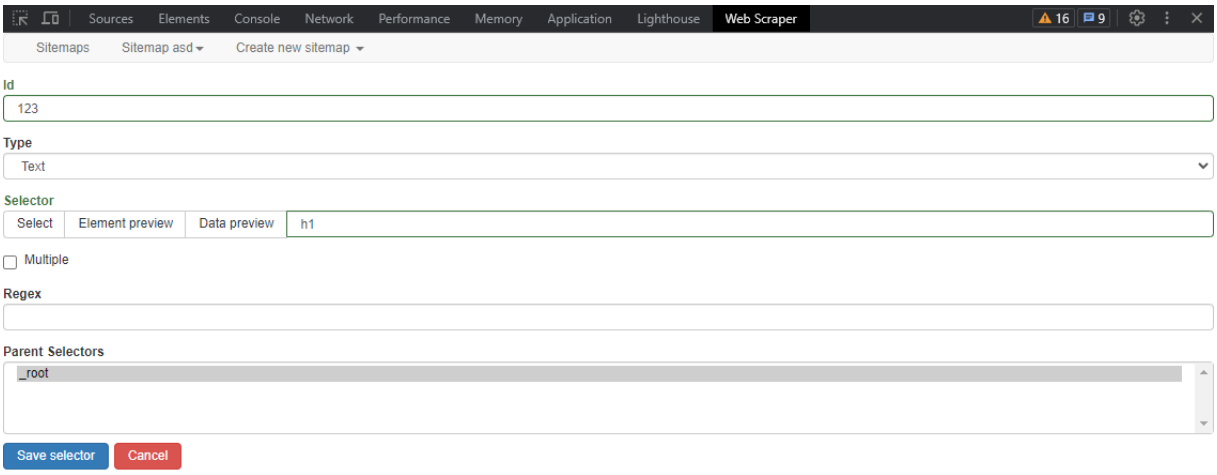
2.2. WebScrapper

WebScrapper [9] to darmowe rozszerzenie dla przeglądarek internetowych, które działa jako dodatek, dostępny w pasku narzędzi deweloperskich po zainstalowaniu. Ułatwia ono proces definiowania skraperów poprzez umożliwienie użytkownikom wskazywania elementów na stronie. Dzięki zawarciu interfejsu rozszerzenia w panelu narzędzi deweloperskich przeglądarki, użytkownik może intuicyjnie wybierać elementy na stronie, a jednocześnie na bieżąco obserwować interfejs użytkownika prototypu.

Podstawowe działanie rozszerzenia polega na analizie struktury strony internetowej i ekstrakcji danych zgodnie z określonymi przez użytkownika regułami. Użytkownicy mają możliwość wskazania elementów strony, takich jak tekst, obrazy czy linki oraz określenia, jakie informacje mają z nich pozyskać. Dzięki temu rozszerzenie umożliwia tworzenie precyzyjnych zapytań i selektorów.

WebScrapper oferuje również możliwość dostosowania skrapowanych danych. Użytkownicy mogą definiować własne porównania i filtry, co pozwala na precyzyjne przetwarzanie i selekcję.

Niestety, przez swój nieintuicyjny interface (przedstawiony na rysunku 2.), rozszerzenie jest trudne w obsłudze, a kolejnym minusem jest możliwość wykonywania skrapowania tylko po stronie przeglądarki.



Rysunek 2. GUI rozszerzenia WebScrapper. Źródło: [9]

Wady:

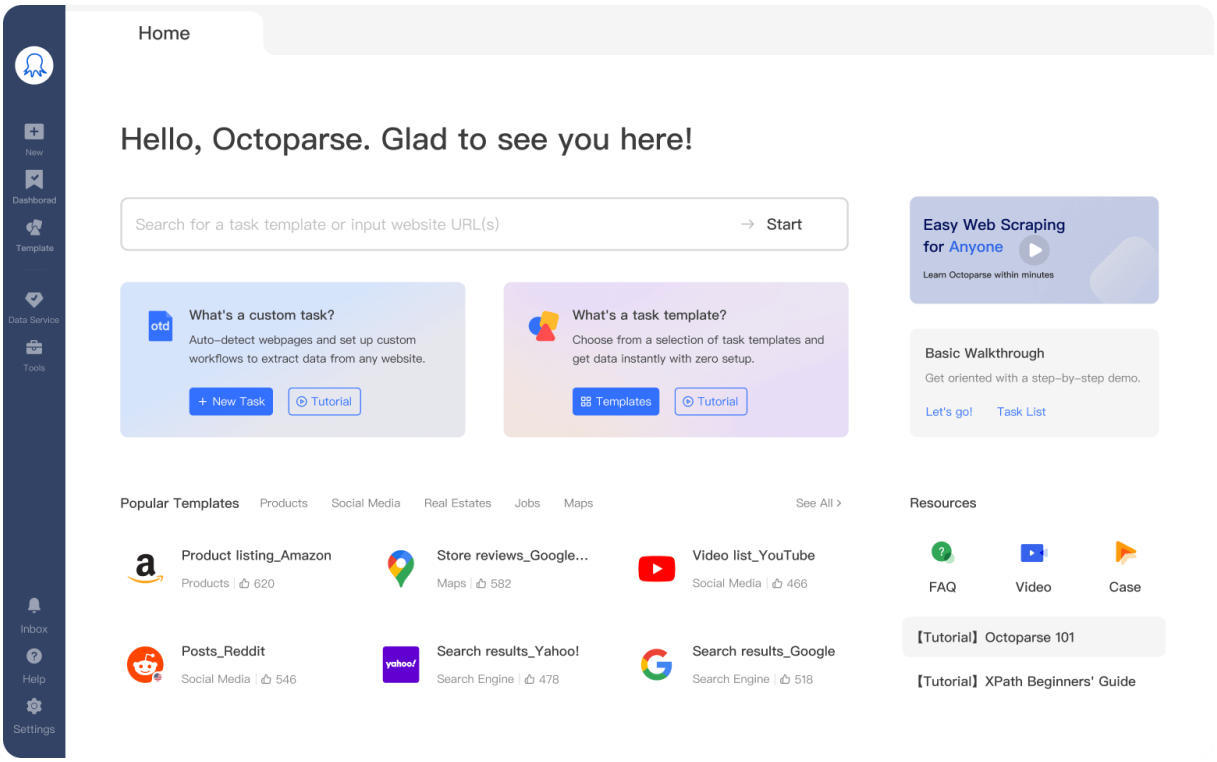
- Skomplikowane struktury stron mogą wymagać zaawansowanych umiejętności programistycznych do poprawnej definicji reguł ekstrakcji danych.
- Interfejs użytkownika może nie być intuicyjny, a brak dokumentacji może utrudnić przystosowanie się do narzędzia.
- Skrapownie może odbywać się jedynie po stronie przeglądarki.

Zalety:

- WebScrapper jest dostępny dla popularnych przeglądarek opartych na silniku Chromium, takich jak Chrome czy Edge.
- Umieszczenie rozszerzenia w pasku narzędzi w przeglądarce umożliwia wskazywanie elementów na stronie internetowej w prosty sposób, co usprawnia proces definiowania skraperów.
- Narzędzie oferuje elastyczność w definiowaniu reguł i filtrów, co pozwala na dostosowanie procesu skrapowania do indywidualnych potrzeb użytkownika.

2.3. Octoparse

Octoparse [10] to narzędzie umożliwiające pozyskiwanie danych ze stron internetowych bez konieczności pisania kodu. Narzędzie to wymaga instalacji jako aplikacja desktopowa i posiada pewne ograniczenia w wersji darmowej, która umożliwia skrapowanie danych wyłącznie na urządzeniu lokalnym. Ponadto dla bardziej zaawansowanych scenariuszy skrapowania może okazać się konieczne użycie narzędzi programistycznych. Interfejs graficzny został zaprezentowany na rysunku 3.



Rysunek 3. Graficzny interfejs użytkownika aplikacji Octoparse. Źródło: [10]

Zalety:

- Intuicyjny interfejs użytkownika z funkcją „point and click", umożliwiający łatwą interakcję z narzędziem.
- Zaawansowane funkcje filtrowania, sortowania i eliminowania duplikatów.
- Efektywne ekstrahowanie danych ze stron generujących dynamiczne treści.
- Automatyzacja procesów skrapowania, z opcją planowania zadań i regularnego monitorowania danych.
- Możliwość zapisu wyników skrapowania w wielu formatach danych.

Wady:

- Jest to aplikacja desktopowa, co wymaga instalacji na komputerze użytkownika.
- W wersji darmowej narzędzia występują ograniczenia, takie jak limit na liczbę stron do skrapowania i konieczność korzystania z własnego sprzętu.
- Dla niestandardowych skryptów skrapowania może być konieczne korzystanie z narzędzi programistycznych.

2.4. Datahen.com

Datahen [11] to usługa webowa umożliwiająca web skraping. Z jej pomocą użytkownicy mogą automatycznie skrapować dane ze stron internetowych na dużą skalę. Co więcej, usługa ta zapewnia wsparcie dla złożonych scenariuszy skrapowania, takich jak dynamicznie generowane treści czy dla stron, które wymagają logowania. Datahen oferuje również możliwość monitorowania zmian na stronach i gromadzenia tylko nowo dodanych danych. Niemniej jednak obsługa Datahen wymaga pewnego stopnia znajomości programowania i nie jest tak intuicyjna jak niektóre z narzędzi.

Zalety:

- Umożliwia skrapowanie danych na dużą skalę.
- Wsparcie dla skomplikowanych scenariuszy skrapowania, takich jak strony generujące dynamiczne treści.
- Możliwość monitorowania zmian na stronach i zbierania nowych danych.
- Chmura do przechowywania i zarządzania danymi.

Wady:

- Wymaga pewnej wiedzy z zakresu programowania.
- Wykorzystanie może nie być tak intuicyjne jak w przypadku innych narzędzi.

2.5. Podsumowanie istniejących rozwiązań

Podsumowując, nowoczesne technologie skrapowania stron internetowych dostarczają zróżnicowane narzędzia oferujące szerokie spektrum funkcji. Poniżej przedstawione są ich główne zalety i wady:

Zalety:

- Różnorodność funkcji - narzędzia te dostarczają bogaty zestaw funkcji ekstrakcji danych, obejmujących obsługę dynamicznych stron internetowych, zaawansowane techniki ekstrakcji oraz automatyzację procesów.
- Elastyczność - użytkownicy mają możliwość definiowania indywidualnych reguł i filtrów, co pozwala na precyzyjne ekstrahowanie pożądanych informacji ze stron internetowych.
- Intuicyjne interfejsy - wiele z tych narzędzi oferuje przyjazne dla użytkownika interfejsy, co ułatwia tworzenie zapytań i definiowanie reguł ekstrakcji.
- Wsparcie dla różnych formatów danych - narzędzia te obsługują zapisywanie danych w różnych formatach, co pozwala użytkownikom na pracę z danymi w preferowanym formacie.

Wady:

- Złożoność - niektóre narzędzia mogą okazać się skomplikowane dla niedoświadczonych użytkowników lub osób bez umiejętności programistycznych.
- Ograniczenia kosztowe - część z narzędzi oferuje swoje usługi w modelu płatnym, co może stanowić barierę dla osób lub projektów o ograniczonym budżecie.
- Wyzwania związane ze skomplikowanymi stronami - narzędzia do skrapowania mogą napotykać problemy przy obsłudze stron o złożonej strukturze lub posiadających zabezpieczenia przeciwko skrapowaniu.

- Potrzeba umiejętności programowania - w zależności od narzędzia i stopnia złożoności zadania, użytkownicy mogą potrzebować umiejętności programowania lub użycia dodatkowych narzędzi, aby osiągnąć zamierzone cele.
- Ograniczenia w modyfikacji danych - niektóre narzędzia, w ograniczonym stopniu umożliwiają modyfikowanie zeskrapowanych danych, pozostawiając tę część pracy użytkownikowi.
- Brak automatyzacji skrapowania - tylko niektóre z wymienionych narzędzi posiada tę funkcjonalność.
- Brak systemu powiadomień o osiągniętych wynikach/spełnionych warunkach – narzędzia, które posiadają automatyzację skrapowania często wymagają od użytkownika sprawdzania wyników samemu. Dodatkowo użytkownicy nie mogą sami decydować, w jakich warunkach mają dostać powiadomienie.
- Brak pełnej rozszerzalności - część z narzędzi, nie oferuje pełnej łatwości w rozszerzaniu swoich funkcji o własne implementacje zadań.
- Ograniczenia kompatybilności - niektóre narzędzia mogą być ograniczone do określonych przeglądarek lub silników, co wpływa na ich wsparcie dla różnych platform internetowych.

3. Propozycja rozwiązania

W tym rozdziale zostanie przedstawiona propozycja rozwiązania inteligentnego i elastycznego systemu pozyskiwania danych ze stron internetowych. Zaprezentowane zostaną również cechy i kryteria, które to rozwiązanie powinno spełniać, opierając się na aktualnym stanie sztuki oraz stawianych mu wymaganiach.

Propozycję systemu stanowić będzie dwukomponentowe rozwiązanie - wtyczka do przeglądarek opartych na Chromium, która pełni rolę części klienta, oraz usługę serwerową RESTful [12].

Część kliencka będzie umożliwiać użytkownikowi zalogowanie się, zdefiniowanie selektorów wskazujących na elementy HTML [13], ustawienie cyklicznego skrapowania, a także dodawania własnych modyfikatorów i sposobów powiadamiania o spełnionych warunkach skrapowania cyklicznego.

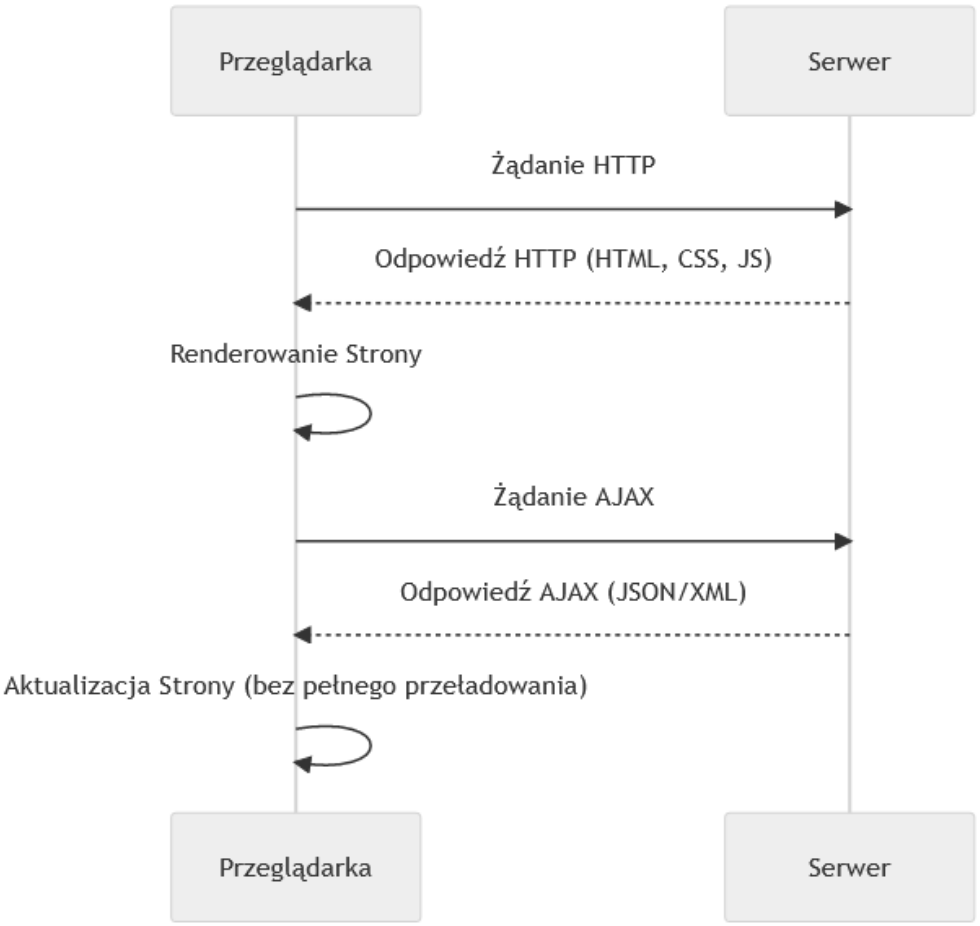
Natomiast serwer, działający w oparciu o REST API, odpowiadać będzie za realizację logiki rozwiązania. Jego zadaniem jest pozyskiwanie danych, cykliczne uruchamianie skraperów oraz udostępnianie danych, które obejmują między innymi: autoryzację, wyniki i stany zdefiniowanych przez użytkownika selektorów oraz ustawień cyklicznych.

3.1. Technika skrapowania

HTML [13] *(akronim od angielskiego Hyper Text Markup Language)*, jest językiem, który stanowi podstawę tworzenia stron internetowych. Każda strona internetowa, z którą się spotkamy, jest zbudowana z elementów HTML. Każdy z tych elementów jest opisany przez specyficzny znacznik HTML - `tag`. Aby skutecznie ekstrahować informacje ze stron internetowych, niezbędne jest zrozumienie struktury HTML. Bardziej szczegółowy opis tej technologii i ściśle z nią powiązanego języka CSS znajduje się w podrozdziale 4.9.

Struktura HTML jest hierarchiczna i tworzy tzw. „drzewo DOM" [14] *(Document Object Model)*. Elementy HTML są umieszczone w drzewie w taki sposób, że niektóre tagi są „rodzicami", a inne są „dziećmi". Dzięki temu techniki skrapowania stron internetowych często korzystają z tej hierarchicznej struktury, przeszukując drzewo od góry do dołu (w głąb) lub na szerokość.

Mimo że skrapowanie stron jest częste, spotkać można pewne utrudnienia. Przykładem są strony SPA (*Single Page Application*) oraz technologie dynamicznego ładowania danych, jak AJAX (*Asynchronous JavaScript and XML*) [15]. Te mechanizmy mogą sprawić, że pełna zawartość strony staje się dostępna dopiero po wykonaniu określonych interakcji lub po pewnym czasie, co komplikuje proces ekstrakcji informacji. Sekwencja ładowania stron w taki sposób przedstawiony jest na rysunku 4.



Rysunek 4. Diagram sekwencji przedstawiający sposób działania stron wykorzystujących AJAX. Źródło: opracowanie własne.

W związku z tym, dla stron zawierających elementy dynamiczne, bardziej zaawansowane podejście polega na symulacji przeglądania internetowego. Oznacza to, że oprócz pobrania i analizy statycznego HTML, niezbędne jest także uruchomienie zawartego na stronie kodu JavaScript. Narzędzia takie jak WebDriver, Puppeteer czy Selenium pozwalają na taką automatyzację przeglądania, co umożliwia pełne wykonanie skryptów JavaScript i uzyskanie zrenderowanej zawartości strony.

Warto jednak pamiętać, że korzystanie z przeglądarek internetowych do skrapowania stron dynamicznych może być bardziej zasobo- i czasochłonne niż prostsze pobieranie i parsowanie statycznego HTML [16]. Ponadto takie podejście jest bardziej podatne na zmiany w strukturze strony lub w zachowaniach dynamicznych elementów, co może wpłynąć na niezawodność procesu skrapowania.

Do pozyskiwania danych przykładowo, można wykorzystać wyrażenia regularne (*regex*). Umożliwiają on identyfikację i ekstrakcję danych poprzez dopasowanie wzorców w tekście. Przykład wykorzystania takiego wyrażenia w stosunku do HTML pokazany jest w listingu 1.

Ten kod pozwoli na ekstrakcję treści wszystkich tagów `<p>`, jednak nie uwzględnia on specyficznej struktury `div > div > div` i jest trudniejszy w automatycznym generowaniu. Przy bardziej złożonych strukturach HTML, wyrażenia regularne mogą nie działać prawidłowo.

Listing 1. Przykładowy listing w python prezentujący sczytywanie artykułów ze strony HTML przy użyciu wyrażeń regularnych. Źródło: opracowanie własne.

```
import re
html = """
<div>
    <div>
        <div>
            <p>Article 1</p>
            <p>Article 2</p>
            <p>Article 3</p>
        </div>
    </div>
</div>
"""

# Wzór do dopasowania wszystkich tagów <p>
pattern = r'<p>(.*?)<\/p>'

artykuly = re.findall(pattern, html, re.DOTALL)

for artykul in artykuly:
    print(artykul)
```

Alternatywnie, do identyfikowania i ekstrakcji danych, można korzystać z selektorów CSS [17], które zostały stworzone specjalnie do pracy z HTML. Są to wzorce, które pasują do elementów HTML na podstawie ich id, klasy, typu, atrybutów, wartości atrybutów, struktury drzewa i wielu innych kryteriów.

W przypadku wykorzystania selektorów CSS wcześniejszy kod wyglądałby jak w listingu 2.

Listing 2. Przykładowy listing prezentujący sczytywanie artykułów ze strony przy użyciu CSS selector. Źródło: opracowanie własne.

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(html, 'html.parser')

# Użycie selektora CSS do wybrania wszystkich artykułów
articles = soup.select('div > div > div > p')

for article in articles:
    print(article.text)
```

W przeciwieństwie do wyrażeń regularnych, selektory CSS lepiej radzą sobie z zagnieżdżonymi strukturami HTML. Dzięki nim, można w precyzyjny sposób docierać do konkretnych elementów na stronie, korzystając z ich struktury i atrybutów.

Oczywiście, warto wspomnieć, że selektory CSS to nie jedyne narzędzie do parsowania struktury HTML. Istnieje wiele innych technik, takich jak na przykład XPath (*XML Path Language*). Język ten podobnie jak selektory CSS, jest wysoce efektywny w nawigacji po strukturze dokumentu HTML lub

XML[4]. Potrafi z łatwością poruszać się w głąb zagnieżdżonych elementów, a także zawiera znacznie szerszy zestaw funkcji i jest w stanie wykonywać bardziej skomplikowane zapytania.

Jednak w tym rozwiązaniu zdecydowano na wybór selektorów CSS z kilku powodów:

- Selektory CSS są powszechnie stosowane w projektowaniu stron internetowych, więc ich zrozumienie ma wiele dodatkowych korzyści poza tylko ekstrakcją danych.
- Składnia selektorów CSS jest prosta i zwięzła, co ułatwia szybkie tworzenie i modyfikowanie zapytań.
- Większość bibliotek do parsowania HTML, w tym BeautifulSoup, obsługuje selektory CSS natywnie, co sprawia, że są one łatwe do zastosowania w praktyce.

3.2. Wnioski na podstawie istniejących rozwiązań i definicja wymagań

Na podstawie przeprowadzonej w poprzednim rozdziale analizy konkurencyjnych rozwiązań, uwzględniając ich mocne i słabe strony, określono zestaw cech, które powinno posiadać narzędzie do pozyskiwania danych ze stron internetowych. Poniżej przedstawiono cechy charakterystyczne, które zapewnią wysoką funkcjonalność i użyteczność takiego systemu.

Cechy, które powinno posiadać narzędzie do pozyskiwania danych ze stron internetowych:

- Intuicyjny interfejs użytkownika - narzędzie powinno być przyjazne dla użytkowników, niezależnie od ich doświadczenia w web skrapingu. Dzięki temu osoby nieposiadające wiedzy technicznej również będą mogły skorzystać z rozwiązania.
- Rozszerzalność - dla użytkowników z umiejętnościami technicznymi, system powinien oferować możliwość rozszerzenia istniejących funkcji o własne modyfikatory i endpointy[5] powiadomień.
- Zaawansowane modyfikatory skrapowanych danych - narzędzie powinno zapewniać możliwość zaawansowanej modyfikacji i przetwarzania danych pozyskanych w procesie skrapowania.
- Część serwerowa z warstwą logiczną - system powinien umożliwiać dostęp do danych z poziomu chmury, co ułatwi użytkownikom dostęp do danych z poprzednich sesji.
- Dostępność i skalowalność - system powinien być zaprojektowany tak, aby obsłużyć wiele jednoczesnych wywołań od wielu użytkowników, umożliwiając asynchroniczne skrapowanie wielu stron jednocześnie.
- Łatwy dostęp do danych - użytkownik powinien mieć łatwy dostęp do wcześniej pozyskanych danych i możliwość ich eksportu np. do pliku CSV.
- Bezpieczeństwo - system powinien wykorzystywać najnowsze technologie i praktyki z zakresu bezpieczeństwa danych wrażliwych, zapewniając tym samym najwyższy poziom ochrony dla użytkowników.
- Rozszerzalność - system powinien być zaprojektowany w taki sposób, który umożliwia jego dalszy rozwój i adaptację do zmieniających się potrzeb użytkowników i dynamicznie rozwijającego się rynku.
- Wsparcie dla różnych formatów danych - narzędzie powinno umożliwiać skrapowanie danych w różnych formatach, włączając tekst, obrazy i więcej, dając użytkownikowi większą kontrolę nad typem danych, które chce pozyskać.

---

[4] XML – z ang. Extensible Markup Language – język znaczników, przeznaczony do reprezentowania danych w strukturalizowany sposób.

[5] Endpoint w aspekcie webowym to specyficzny adres URL, na który aplikacja lub serwis internetowy może wysyłać żądania w celu uzyskania lub modyfikowania danych.

- Integracja z innymi narzędziami - system powinien umożliwiać łatwą integrację z popularnymi narzędziami analitycznymi i bazami danych, umożliwiając tym samym efektywną analizę i wykorzystanie pozyskanych danych.

3.3. Wprowadzenie kluczowych pojęć

Wprowadzenie kluczowych pojęć jest istotne dla łatwiejszego zrozumienia działania systemu oraz procesu tworzenia i konfigurowania skraperów.

3.3.1. Selektor CSS

W kontekście narzędzia selektor odnosi się do mechanizmu, który umożliwia identyfikację i wybór konkretnych elementów na stronie internetowej. Selektor jest używany w celu precyzyjnego wskazania, z których komponentów HTML mają zostać pozyskane dane podczas procesu skrapowania. W praktyce reprezentuje on selektor CSS, który jest wykorzystywany do odnalezienia elementów na stronie internetowej na podstawie ich atrybutów, klas CSS, identyfikatorów lub innych cech.

3.3.2. Mapa witryny

Mapa witryny jest strukturą danych, która służy jako przewodnik dla narzędzia do skrapowania podczas procesu pozyskiwania informacji z dynamicznych stron internetowych. Mapa witryny zawiera definicje selektorów elementów, które mają zostać pozyskane oraz ich lokalizację na stronie docelowej. Jest to swojego rodzaju plan, który określa, gdzie znajdują się interesujące użytkownika dane na stronie.

Tworzenie mapy witryny polega na wyborze elementów na stronie za pomocą graficznego interfejsu użytkownika dostarczanego przez narzędzie. Użytkownik wskazuje znaczniki HTML, które reprezentują interesujące go elementy, a narzędzie tworzy odpowiednie selektory na podstawie tych wskazówek. Mapa witryny jest zapisywana i używana do automatycznego pozyskiwania danych z wybranych elementów na stronie internetowej.

3.3.3. Robotnik

Robotnik (*Worker*) to jednostka operacyjna w narzędziu do pozyskiwania danych, która wykonuje określone zadania skrapowania zgodnie z wybraną mapą witryny i według ustalonego harmonogramu. Robotnik jest powiązany z określoną mapą witryny i ma zdolność do regularnego i automatycznego skrapowania danych z wybranych elementów na docelowej stronie.

Użytkownik może tworzyć różnych robotników, z których każdy może być skonfigurowany do pozyskiwania danych z różnych stron internetowych na podstawie określonych map witryny. Może być zaplanowany do wykonywania skrapowania w określonym czasie i z określoną częstotliwością, co umożliwia automatyzację procesu pozyskiwania danych.

Tworzenie robotnika obejmuje przypisanie mapy witryny, wybór modyfikatorów danych i filtrów oraz określenie warunków powiadomień, jeśli użytkownik zdecyduje się na ich otrzymywanie. Obiekt ten stanowi centralną jednostkę, która wykonuje skrapowanie danych zgodnie z ustalonymi parametrami i umożliwia użytkownikowi efektywne zarządzanie procesem pozyskiwania danych.

W kontekście tej pracy używane są również określenie takie jak „pracownik”, „planer skrapowania” czy „cykliczne zlecenie skrapowania”.

3.4. Proces tworzenia skraperów

Rozszerzenie musi pozwalać na utworzenie nowej mapy witryn. Ta mapa witryny zawiera definicje selektorów do elementów, które mają zostać pozyskane oraz ich lokalizację na stronie docelowej. Proces tworzenia nowej mapy witryny powinien wyglądać następująco:

1. Otwórz docelową stronę internetową: użytkownik rozpoczyna od otwarcia strony internetowej, którą chce skrapować w przeglądarce internetowej. Będzie ona służyć jako źródło do pozyskiwania danych.
2. Określ szczegóły mapy witryny: w interfejsie użytkownika należy wprowadzić nazwę i adres URL strony docelowej. W razie potrzeby można również sparametryzować adres URL za pomocą symboli zastępczych (np. {subpart}), które można zastąpić określonymi wartościami podczas późniejszego definiowania procesu skrapowania. Ułatwi to użytkownikowi skrapowanie wielu podstron z taką samą strukturą HTML.
3. Wybieranie elementów za pomocą graficznego interfejsu użytkownika: należy wykorzystać graficzny interfejs użytkownika dostarczony przez narzędzie do interakcji z otwartą stroną internetową. Korzystając z GUI[6], użytkownik musi wizualnie znaleźć i wybrać określone elementy na stronie internetowej, z których dane w późniejszych etapach chce pozyskiwać. Wskazywane są znaczniki HTML, więc użytkownik może zaznaczyć każdy element na stronie, który wchodzi w skład komponentów HTML.
4. Zarządzanie wybranymi komponentami (selektorami CSS): po wybraniu żądanych elementów można zarządzać selektorami i dostosowywać je. Powinny być widoczne też w trakcie definiowania mapy witryny.
5. Testowanie selektorów mapy witryny (opcjonalnie): aby zapewnić dokładność konfiguracji mapy witryny, można uruchomić skrapowanie testowe. Pozwala to zweryfikować, czy zdefiniowane selektory CSS dokładnie wyodrębniają żądane dane. Uruchomienie testowe zapewnia informacje zwrotne w czasie rzeczywistym na temat wyodrębnionych danych, umożliwiając wprowadzenie niezbędnych zmian w selektorach.
6. Zapisanie mapy witryny: po sfinalizowaniu konfiguracji skrapingu należy zapisać mapę witryny. Obejmuje to przechowywanie wybranych elementów, odpowiadających im selektorów i wszelkich dodatkowych ustawień konfiguracyjnych takich jak nazwy i URL.

Po utworzeniu mapy witryny z selektorami użytkownik zobaczy ją na liście razem z uprzednio zdefiniowanymi mapami w panelu po zalogowaniu. Na podstawie mapy witryny użytkownik może utworzyć planer skrapowania.

3.5. Modyfikatory danych i filtry

Propozycja w przypadku definiowania cyklicznych zleceń skrapowania (robotników) obejmuje możliwość wyboru modyfikatorów i filtrów dla każdego selektora CSS w mapie witryny. Modyfikatory i filtry umożliwiają użytkownikom manipulację i przetwarzanie wyodrębnionych danych. To metody, które pozwalają na przetwarzanie danych zeskrapowanych z witryny. Na przykład, użytkownik może wykorzystać modyfikator, aby przekształcić tekst *„750 000 zł”* na liczbę *750000*. Dzięki temu osoba korzystająca z systemu może później łatwiej tworzyć warunki dla powiadomień i wykorzystywać te dane po wyeksportowaniu. Użytkownik powinien mieć możliwość wyboru wielu modyfikatorów dla jednego selektora, co pozwoli na przetwarzanie danych w sposób określony przez każdy modyfikator po kolei. Na przykład, użytkownik może zastosować modyfikator *„remove numbers”* (usuń cyfry), a następnie *„count words”* (policz słowa), aby policzyć tylko słowa w tekście wyodrębnionym przez skrapera, a wykluczyć liczby.

Modyfikatory powinny również pozwalać użytkownikom na przeprowadzanie bardziej zaawansowanych operacji na danych pozyskanych podczas skrapowania. Wykorzystanie technik uczenia maszynowego w ramach tych modyfikatorów dostarczy dodatkowego kontekstu. Dzięki temu użytkownicy będą mogli ekstrahować informacje nie tylko z tekstów i tabel, ale także np. z obrazów – pomoże to w zwiększeniu użyteczności zeskrapowanych danych i pozwoli na dokładniejsze definiowanie, kiedy właściciel robotnika ma zostać powiadomiony o pozyskaniu danych spełniających

[6] GUI – ang. *Graphical User Interface* - Graficzny interfejs użytkownika

określony przez niego warunek. Takie zaawansowane modyfikatory umożliwią np. automatyczną klasyfikację i ekstrakcję/przekształcanie danych, co ułatwi wydobywanie kluczowych informacji.

Rozwiązanie powinno domyślnie zawierać szereg modyfikatorów, takich jak:

- Uppercase - przekształca tekst na wielkie litery. Na przykład, jeśli tekst to *„hello world”*, po zastosowaniu modyfikatora otrzymamy „*HELLO WORLD*”.
- Lowercase - przekształca tekst na małe litery. Na przykład, jeśli tekst to *„HELLO WORLD”*, po zastosowaniu modyfikatora otrzymamy *„hello world”*.
- Get numbers - wyodrębnia liczby z tekstu. Na przykład, jeśli mamy tekst *„abc 123 xyz”*, po zastosowaniu modyfikatora otrzymamy *„123”*.
- Lemmatize PL - wykonuje lematyzację polskiego tekstu, czyli sprowadza wyrazy do ich podstawowej formy. Na przykład, słowo *„samochodów”* zostanie sprowadzone do formy *„samochód”*.
- Lemmatize ENG - wykonuje lematyzację angielskiego tekstu, czyli sprowadza wyrazy do ich podstawowej formy. Na przykład, słowo *„cars”* zostanie sprowadzone do formy *„car”*.
- Remove interpunctions - usuwa znaki interpunkcyjne z tekstu. Na przykład, jeśli mamy tekst *„Hello, world!”*, po zastosowaniu modyfikatora otrzymamy *„Hello world”*.
- Remove white spaces - usuwa białe znaki (spacje, tabulatory itp.) z tekstu. Na przykład, jeśli mamy tekst *" hello world "*, po zastosowaniu modyfikatora otrzymamy „helloworld”.
- Extract hashtags - wyodrębnia hasztagi (słowa poprzedzone znakiem "#") z tekstu. Na przykład, jeśli mamy tekst *„Today's #weather is great!”*, po zastosowaniu modyfikatora otrzymamy *„#weather”*.
- Extract email - wyodrębnia adresy e-mail z tekstu. Na przykład, jeśli mamy tekst *„Mój e-mail to john@example.com”*, po zastosowaniu modyfikatora otrzymamy *„john@example.com”,*
- Extract date - wyodrębnia daty z tekstu. Na przykład, jeśli mamy tekst *„Dziś jest 2023-16-06”*, po zastosowaniu modyfikatora otrzymamy *„2023-16-06”*.
- Count words - zlicza liczbę słów w tekście. Na przykład, jeśli mamy tekst *„To jest przykładowe zdanie”*, po zastosowaniu modyfikatora otrzymamy liczbę 4.
- Remove numbers - usuwa cyfry z tekstu. Na przykład, jeśli mamy tekst „abc 123 xyz”, po zastosowaniu modyfikatora otrzymamy „abc xyz”.
- Extract phone number - wyodrębnia numery telefonów z tekstu. Na przykład, jeśli mamy tekst *„Mój numer telefonu to +48 5008310XX”*, po zastosowaniu modyfikatora otrzymamy *„+48 5008310XX"*.

Dodatkowo użytkownik może zdefiniować własne, niestandardowe modyfikatory, które pomogą mu osiągnąć własny, bardziej indywidualny cel. Przykładem może być dodanie modyfikatora *"Image content analyzer"*, który może analizować zawartość zdjęć, przyjmując ich adres URL (ze wskazanego przez selektor zdjęcia) i zwracając listę wykrytych na nim obiektów.

Filtry pozwalają użytkownikowi porównywać przetworzone dane (po zastosowaniu modyfikatorów) z określonymi przez niego argumentami. Jeśli wynik porównania w trakcie działania robotnika jest pozytywny i użytkownik zaznaczył, że chce zostać powiadomiony, to zostanie mu wysłana mu informacja o spełnionym warunku. Proponowane filtry to *„contains”* (zawiera), *„equals”* (równa się) oraz *„greater than”* (większa niż) i *„less than”* (mniejsza niż). Filtry te umożliwiają precyzyjne określanie warunków powiadomień - porównywanie wyodrębnionych i przekształconych przez modyfikatory danych z określoną wartością.

3.6. Proces tworzenia cyklicznych zleceń pozyskiwania danych

Definiując zlecenia cykliczne z obsługą modyfikatorów, filtrów, automatyzacji i powiadomień, użytkownik może dostosować i ulepszyć proces skrapowania do swoich konkretnych potrzeb. Elastyczne narzędzie do skrapowania stron internetowych powinno pozwalać na wydajną i ukierunkowaną ekstrakcję danych, wraz z możliwością automatyzacji i planowania zadań skrapingu. Dodatkowo funkcja powiadomień będzie informować użytkowników, jeśli zeskrapowane dane będą spełniać zdefiniowane przez niego warunki.

Proces definiowania takiego robotnika powinien przebiegać w następujący sposób:

1. Zdefiniowanie podstawowych parametrów: proces rozpoczyna się od zdefiniowania podstawowych parametrów dla robotnika. Użytkownik wybiera mapę witryny, z którą będzie powiązany, oraz nadaje mu nazwę.
2. Określenie modyfikatorów i filtrów: dla każdego selektora CSS zawartego w mapie witryny, użytkownik może zdefiniować listę modyfikatorów, które zostaną wykonane na danych pozyskanych przez dany selektor. Modyfikatory są stosowane w kolejności określonej przez użytkownika. Może on również określić warunki, które muszą zostać spełnione, aby otrzymać powiadomienie o napotkaniu przez robotnika oczekiwanych danych – posłużą do tego filtry.
3. Określenie listy dla sparametryzowanego URL: Jeśli użytkownik zdefiniował parametr *{subpart}* w URL mapy strony, to konieczne jest określenie listy wartości dla tej części URL. Osoba korzystająca z systemu musi zatem dostarczyć listę, która zostanie później użyta do zamiany części *{subpart}* na odpowiednie wartości podczas skrapowania.
4. Automatyzacja wywoływania robotnika: należy przeprowadzić konfiguracje częstotliwość i zakres czasowy, w którym robotnik automatycznie będzie wykonywać skrapowanie. Określa się także interwały czasowe, w jakich skrapowanie będzie miało miejsce.
5. Konfiguracja powiadomień: należy dokonać wyboru preferowanego sposób otrzymywania powiadomień o spełnieniu warunków zdefiniowanych w punkcie drugim, dla danych pozyskanych przez robotnika.

3.7. Architektura rozwiązania

Architektura rozwiązania będzie składała się z dwóch głównych komponentów, które współpracują ze sobą, aby zapewnić wymagane funkcjonalności. Głównym komponentem tej architektury będzie część serwerowa (*backend*), która obsługuje zadania ekstrakcji i przetwarzania danych. Drugim kluczowym komponentem będzie warstwa aplikacji klienta (*frontend*), która będzie udostępniać użytkownikowi funkcjonalności aplikacji poprzez graficzny interfejs w narzędziach deweloperskich przeglądarki opartej na Chromium.

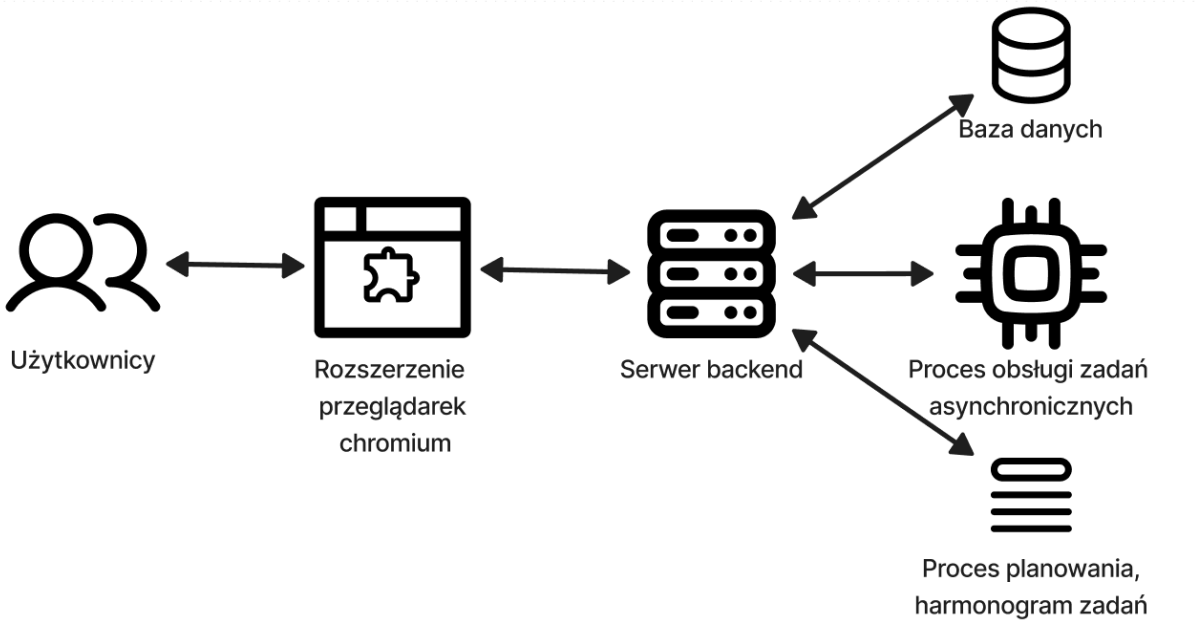
Komponent serwerowy będzie odpowiadać za zarządzanie procesem skrapowania, zarządzanie harmonogramami skrapowań, obsługę żądań HTTP oraz analizę treści HTML. Jego podstawowe funkcje obejmować będą interakcję z docelowymi stronami internetowymi, pobieranie pożądanych danych oraz wykonywanie operacji przekształcania i czyszczenia danych.

Za przyjazny dla użytkownika interfejs i płynną interakcję z systemem skrapującym odpowiadać będzie komponent interfejsu użytkownika. Umożliwiać on będzie zarówno użytkownikom technicznym, jak i nietechnicznym, konfigurowanie ustawień pozyskiwania danych, określanie docelowych stron internetowych oraz wizualizację wyodrębnionych danych w przejrzysty sposób.

Komunikacja między częścią kliencką, a serwerem odbywać się będzie za pośrednictwem interfejsów API RESTful, co zapewni płynny przepływ danych i synchronizację. Rozszerzenie wysyłać będzie żądania do serwera w celu zainicjowania zadań pozyskiwania danych, pobierania wyników skrapowań i otrzymywania aktualizacji statusu. W odpowiedzi serwer przetwarza te żądania, wykonuje operacje skrapowania i zwraca wyniki do GUI klienckiego.

Oprócz podstawowych komponentów serwerowych i aplikacji użytkownika architektura może również zawierać usługi pomocnicze, takie jak bazy danych, systemy buforowania i kolejki

komunikatów. Te usługi mogą zwiększyć wydajność, poprawić skalowalność i umożliwić zaawansowane funkcje, takie jak planowanie zadań, buforowanie wyników i przetwarzanie rozproszone. Rysunek 5. przedstawia zależności między komponentami aplikacji i użytkownikiem rozszerzenia.



Rysunek 5. Diagram opisujący propozycję architektury rozwiązania. Źródło: opracowanie własne.

3.8. Rozszerzenie do przeglądarki

Rozszerzenie będzie pełnić rolę warstwy klienckiej. Poszerza ono możliwości przeglądarki w łatwo dostępny dla użytkownika sposób – wystarczy ze użytkownik pobierze i zainstaluje je w przeglądarce, a następnie otworzy panel narzędzi deweloperskich i przejdzie do zakładki z rozszerzeniem, aby wyświetlić główny interfejs aplikacji.

Rozszerzenie będzie wykorzystywać możliwości przeglądarki internetowej i będzie wchodzić w interakcję z jej wbudowanymi narzędziami i interfejsami API. Ponadto będzie integrować się z narzędziami deweloperskimi, umożliwiając użytkownikom dostęp i wykorzystanie jego funkcji bezpośrednio z interfejsu przeglądarki. Takie umiejscowienie zapewnia jednoczesny dostęp do skrapowanej strony internetowej oraz do GUI systemu – znacząco ułatwi to proces dodawania i edytowania mapy strony.

3.9. Usługa serwerowa z bazą danych i z funkcją pozyskiwania danych ze stron

Serwer narzędzia do skrapowania danych pełnić będzie szereg kluczowych funkcji, które łącznie umożliwią efektywne i bezpieczne pozyskiwanie danych ze stron internetowych. Usługa ta odegra też kluczową rolę w przepływie informacji pomiędzy komponentami systemu i zapewni płynność operacji oraz synchronizację, umożliwiając skuteczną koordynację pracy różnych elementów systemu.

Pierwszą z tych ról jest zarządzanie cyklicznymi skanowaniami stron internetowych. Usługa kontroluje harmonogram i wywołuje skrypty skrapujące w odpowiednich momentach, zapewniając monitorowanie wybranych przez użytkowników stron. Dodatkowo system powinien zarządzać

zasobami, takimi jak pamięć czy moc obliczeniowa, aby skrapowanie odbywało się w sposób optymalny. Wybór technologii ułatwiającej cykliczność wykonywania operacji jest tu kluczowy.

Pod względem praktycznym, serwer będzie odpowiedzialny za implementację algorytmów skrapowania. Oznacza to, że system musi interpretować skrypty skrapujące, interagować je z witrynami internetowymi i ekstrahować oraz przetwarzać dane zgodnie z zadanymi ustawieniami robotników. Zastosowane algorytmy muszą być też zdolne do radzenia sobie z dynamiczną naturą stron internetowych.

Kolejną istotną funkcją serwera jest kontrola autoryzacji i dostępu do wyników skrapowania oraz do selektorów i ustawień cyklicznych zdefiniowanych przez użytkowników. Zapewnia to bezpieczne przechowywanie i dostęp do danych zarówno dla użytkowników, jak i dla samych procesów skrapowania.

W kontekście skalowalności całego systemu serwer będzie odgrywać największą rolę. Możliwość obsługi wielu zapytań jednocześnie oraz asynchroniczne skrapowanie stron zapewnią szybkie i efektywne działanie rozwiązania, niezależnie od liczby użytkowników korzystających z narzędzia jednocześnie. Struktura projektu i kod powinny umożliwiać łatwe rozbudowywanie i rozszerzanie funkcjonalności systemu w przyszłości.

Kolejnym istotnym elementem usługi serwerowej jest system zarządzania bazą danych (SZBD). To narzędzie jest niezbędne do skutecznego składowania, zarządzania i odzyskiwania wyników skrapowania danych. DBMS przechowuje szeroki zakres informacji, w tym dane o stronach internetowych do skrapowania, definicje skryptów skrapujących, logi operacji, wyniki skrapowania i informacje o użytkownikach.

Nie zapominając o bezpieczeństwie, system musi stosować aktualne technologie i praktyki z zakresu bezpieczeństwa danych, co zagwarantuje ochronę wrażliwych danych użytkowników.

4. Wykorzystane technologie, metodyki i języki

W tym rozdziale zostaną wymienione i opisane technologie i protokoły wykorzystane przy wytwarzaniu prototypu składającego się z rozszerzenia do przeglądarki i usługi serwerowej.

4.1. TypeScript i JavaScript

TypeScript [18] jest statycznie typowanym językiem programowania, będącym nadzbiorem języka JavaScript (JS). Został stworzony przez Microsoft i jest rozwijany jako oprogramowanie open source[7]. TypeScript wprowadza statyczną typizację, co oznacza, że programiści mogą określać typy zmiennych, funkcji i innych elementów kodu. Typy te są sprawdzane podczas kompilacji, co pomaga wykrywać błędy i zwiększa bezpieczeństwo kodu. Ponadto TypeScript wprowadza nowe funkcje, takie jak interfejsy, klasy, moduły i wyliczenia, które ułatwiają organizację i strukturę kodu.

Dzięki statycznemu typowaniu TypeScript zapewnia większą czytelność kodu i ułatwia jego zrozumienie przez innych programistów. Dodatkowo środowiska programistyczne (IDE) wspierające TypeScript mogą dostarczać bardziej zaawansowane funkcje, takie jak podpowiedzi dotyczące składni i autouzupełnianie, co dodatkowo przyspiesza proces programowania.

Używanie TypeScript zamiast JavaScript w projekcie systemu do pozyskiwania danych ze stron internetowych jest korzystne z kilku powodów:

- TS zapewnia statyczne typowanie, co zwiększa niezawodność kodu i zmniejsza liczbę potencjalnych błędów, a także umożliwia wychwytywanie błędów w trakcie rozwoju.
- TS obsługuje nowoczesne funkcje ECMAScript [19], zapewniając jednocześnie dodatkowe funkcje, takie jak interfejsy, klasy i moduły, które pomagają w strukturyzacji i organizacji bazy kodu.
- TS oferuje lepszą obsługę IDE, w tym autouzupełnianie, sprawdzanie typów i narzędzia do refaktoryzacji, co prowadzi do poprawy produktywności.
- Ponadto TS kompiluje się do zwykłego JavaScript, umożliwiając płynną integrację z istniejącymi bibliotekami JS i zapewnia to możliwość wykorzystania kodu w rozszerzeniu Chromium.

4.2. React

React [20] to biblioteka JavaScript wykorzystywana do tworzenia interfejsów użytkownika w aplikacjach internetowych. Przez środowisko programistów jest ceniona za swoją deklaratywną naturę, która umożliwia programistom opisywanie, jak interfejs powinien wyglądać w zależności od stanu aplikacji. React wykorzystuje komponenty, które są samodzielnymi blokami kodu, odpowiedzialnymi za konkretne części interfejsu. Te komponenty mogą być ponownie wykorzystywane i łączone w hierarchię, co przyczynia się do modularności i łatwości w utrzymaniu kodu.

React obsługuje również jednokierunkowy przepływ danych, co ułatwia śledzenie i zarządzanie stanem aplikacji.

Kolejną z kluczowych zalet Reacta jest rozbudowana społeczność oraz ekosystem narzędzi i bibliotek. Dzięki temu programiści mają dostęp do wielu zasobów, narzędzi deweloperskich oraz gotowych komponentów, które można wykorzystać w projektach.

[7] Open Source, czyli otwarte oprogramowanie, odnosi się do modelu licencjonowania oprogramowania, który umożliwia ogólny dostęp do kodu źródłowego. Ten model promuje filozofię, która zakłada, że współdzielenie i kolaboracja mogą prowadzić do lepszego i szybszego rozwoju oprogramowania.

W kontekście skrapowania stron internetowych, użycie Reacta może przynieść korzyści, takie jak łatwa strukturyzacja kodu, ponowna możliwość wykorzystywania komponentów interfejsu użytkownika oraz możliwość efektywnego odświeżania danych w czasie rzeczywistym.

4.3. React-bootstrap

React-Bootstrap [21] to szeroko stosowana biblioteka komponentów i styli UI do tworzenia responsywnych i interaktywnych interfejsów internetowych w React. Dzięki React-Bootstrap programiści mogą korzystać z kolekcji wstępnie zaprojektowanych i konfigurowalnych komponentów, takich jak przyciski, formularze, moduły i elementy nawigacyjne, aby tworzyć atrakcyjne wizualnie i funkcjonalne interfejsy użytkownika. Integracja biblioteki z React pozwala na płynne zarządzanie stanem, obsługę zdarzeń i dynamiczne aktualizacje. Jej popularność wynika z prostoty, elastyczności i obszernej dokumentacji, co czyni ją preferowanym wyborem dla programistów chcących tworzyć nowoczesne i responsywne aplikacje internetowe.

4.4. Django

Django [22] jest otwartoźródłowym frameworkiem do tworzenia aplikacji webowych w języku Python. Zaprojektowany został w celu zapewnienia szybkiego i wydajnego tworzenia skalowalnych, bezpiecznych i elastycznych aplikacji. Django opiera się na wzorcu projektowym Model-Widok-Kontroler (MVC) lub Model-Szablon-Widok (MTV) [23], który pomaga w organizacji kodu i separacji logiki biznesowej od warstwy prezentacji.

Jedną z głównych cech Django jest jego wbudowany ORM (*Object-Relational Mapping*), który pozwala na interakcję z bazami danych za pomocą obiektów Pythona. ORM Django umożliwia programistom tworzenie modeli danych, które reprezentują tabele w bazie danych. Modele te można definiować przy użyciu języka Python, a Django automatycznie generuje odpowiednie zapytania SQL dla konkretnej bazy danych. Dzięki temu programiści mogą pracować z danymi w sposób obiektowy, co ułatwia zarządzanie danymi i unika konieczności bezpośredniego manipulowania zapytaniami SQL.

Django oferuje także zestaw narzędzi do zarządzania sesjami, uwierzytelnianiem, autoryzacją, walidacją formularzy i obsługą plików statycznych. Ponadto Django zapewnia elastyczność w obszarze konfiguracji, dzięki czemu programiści mogą dostosować zachowanie aplikacji, takie jak ustawienia bazy danych, cache czy logi, do swoich indywidualnych potrzeb.

Użycie Django w projekcie systemu do pozyskiwania danych ze stron internetowych zapewni możliwość łatwego definiowania modyfikatorów pozyskanych danych. Większość modeli ML[8] dostępnych w Internecie jest dostępna dla języka python.

4.5. Selenium

Selenium [16] to zestaw narzędzi i bibliotek programistycznych, które umożliwiają programistom tworzenie i wykonanie testów automatycznych na stronach internetowych i aplikacjach webowych. Selenium umożliwia programistom symulowanie interakcji użytkownika z aplikacją webową, takich jak kliknięcia, wprowadzanie danych, nawigacja po stronach, a także odczytywanie treści strony i weryfikację jej poprawności. Jest często używany do testowania funkcjonalności, wydajności, kompatybilności i użyteczności aplikacji webowych [24]. Selenium działa w połączeniu z silnikami przeglądarek internetowych, takich jak Chrome, Firefox, Safari, Edge, umożliwiając automatyczne sterowanie tymi przeglądarkami i interakcję z nimi.

[8] ML – ang. Machine Learning – nauczanie maszynowe

Podstawowym elementem Selenium jest WebDriver, który jest odpowiedzialny za komunikację z instancją przeglądarki otwartą przez Selenium i kontrolę nad nią. WebDriver umożliwia programistom otwieranie okna przeglądarki, przechodzenie do konkretnych adresów URL, znajdowanie elementów na stronie (np. przycisków, pól tekstowych) za pomocą selektorów CSS lub XPath, oraz wykonanie różnych akcji na tych elementach.

Niektóre strony internetowe, zwłaszcza te bardziej złożone, wykorzystują techniki, takie jak leniwe ładowanie (*lazy loading*) lub asynchroniczne wczytywanie danych za pomocą technologii AJAX [15]. Te metody pozwalają na optymalizację wydajności strony, ale mogą stanowić problem dla tradycyjnych narzędzi do skrapowania danych, które nie są w stanie interpretować i wykonać takich skryptów.

Dzięki Selenium system jest w stanie interaktywnie ładować stronę, wykonując skrypty JavaScript. Pozwala to na wczytanie i ekstrakcję danych, które mogą nie być dostępne przy użyciu prostszych technik skrapowania. Narzędzie, działając jako pełnoprawna przeglądarka, jest w stanie wczytać całą stronę, włącznie z danymi ładowanymi dynamicznie. Dzięki temu istnieje możliwość pozyskania pełnego obraz strony i jej zawartości.

4.6. Celery i Celery Beat

Celery [25] to popularne narzędzie do zarządzania zadaniami asynchronicznymi w aplikacjach opartych na Django. Działa jako efektywne rozwiązanie do przetwarzania zadań w tle, takich jak przetwarzanie dużej ilości danych, wysyłanie powiadomień e-mail, generowanie raportów i inne.

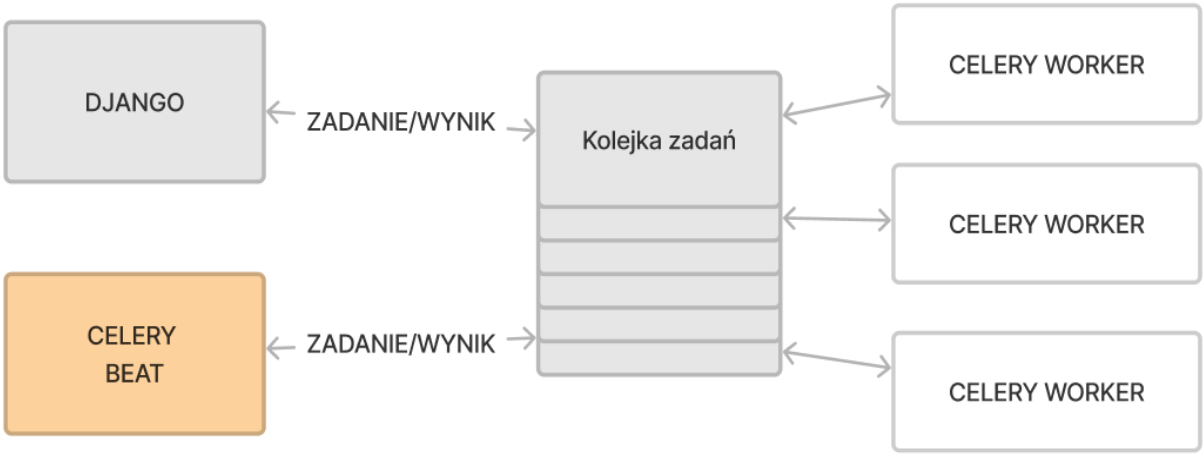
Aplikacje Django definiują zadania jako funkcje, które mogą być uruchamiane synchronicznie lub asynchronicznie. Kiedy zadanie jest uruchamiane asynchronicznie, Celery przekazuje je do kolejki wiadomości. Od tego momentu, zadanie jest niezależne od głównego wątku żądania HTTP, co przekłada się na lepszą wydajność i responsywność aplikacji.

Pracownicy Celery (*workers*) pobierają zadania z kolejki i wykonują je. Wymiana informacji pomiędzy aplikacją Django a pracownikami odbywa się za pośrednictwem brokera wiadomości, który jest kolejnym kluczowym elementem Celery.

W ramach projektu wykorzystano również dodatek do Celery - Celery Beat. To narzędzie umożliwia harmonogramowanie i planowanie zadań. Dzięki niemu można określać interwały czasowe, w których zadania powinny być automatycznie uruchamiane. Jest to szczególnie przydatne w przypadku zadań cyklicznych, dziennych, tygodniowych itp.

Celery oferuje również wiele elastycznych opcji konfiguracyjnych, takich jak ustawienia brokera wiadomości, liczby pracowników, zarządzanie błędami, monitorowanie zadań i inne. Ponadto dostępna jest obszerna dokumentacja i wsparcie społeczności, co ułatwia rozwiązywanie problemów i rozwój aplikacji opartych na Celery.

Rysunek 6 przedstawia uproszczony sposób zlecania i wykonywania zadań przy pomocy Celery i Celery Beat.



Rysunek 6. Diagram przedstawiający sposób zarządzania zadaniami asynchronicznymi Celery i Celery Beat. Źródło: opracowanie własne.

4.7. BeautifulSoup

BeautifulSoup [26] to biblioteka Python służąca do parsowania i przetwarzania danych zawartych w dokumentach HTML i XML. Wykorzystywana jest w ekstrakcji informacji z witryn internetowych, web scrapingu oraz analizie struktury dokumentów. Pozwala na proste poruszanie się po strukturze dokumentu HTML/XML, umożliwiając wyszukiwanie i ekstrakcję konkretnych elementów na podstawie selektorów CSS i wyrażeń XPath.

BeautifulSoup daje możliwość selekcji komponentów HTML według tagów, atrybutów, klas CSS, identyfikatorów, co dostarcza użytkownikom dużą elastyczność w procesie ekstrakcji danych. Oferuje też szereg metod manipulacji i modyfikacji struktury dokumentu, umożliwiając dodawanie, usuwanie i modyfikowanie elementów, a także zmianę atrybutów i ich zawartości. Wszystko to pozwala dostosować strukturę dokumentu do specyficznych potrzeb lub przekształcać dane do bardziej czytelnego formatu.

BeautifulSoup charakteryzuje się łatwością w użyciu, a także dostępną i czytelną dokumentacją. Liczne przykłady i poradniki ułatwiają szybkie zrozumienie działania biblioteki i efektywne rozpoczęcie pracy z nią. Biblioteka ta jest również wysoce rozszerzalna, umożliwiając integrację z innymi bibliotekami i narzędziami Pythona, takimi jak pandas do analizy danych lub requests/Selenium do pobierania stron internetowych.

W kontekście tego projektu, do załadowania danych do BeautifulSoup wykorzystano narzędzie Selenium. Jest to istotne, ponieważ BeautifulSoup nie obsługuje stron dynamicznych, które ładują fragmenty strony poprzez kolejne wywołania HTTP. Zatem, w ramach niniejszego projektu, BeautifulSoup pełni głównie funkcję wyboru elementów HTML na stronach wczytanych przy użyciu Selenium.

4.8. HTTP

HTTP (*Hypertext Transfer Protocol*) to protokół klient-serwer używany do komunikacji w Internecie. Wykorzystuje model żądanie-odpowiedź, gdzie klient (np. przeglądarka internetowa) wysyła żądania do serwera, który odpowiada przesyłając żądaną treść. HTTP jest protokołem bezstanowym, co oznacza, że serwer nie przechowuje żadnych informacji między dwoma żądaniami. Protokół ten obsługuje różne metody żądań, takie jak:

- GET: służy do pobierania danych z serwera.
- POST: umożliwia wysłanie danych do serwera.

- HEAD: podobne do GET, ale pobiera tylko nagłówki odpowiedzi.
- PUT: pozwala na aktualizację istniejącego zasobu.
- DELETE: służy do usunięcia zasobu.

HTTP jest także rozszerzalne, co oznacza, że pozwala na dodawanie niestandardowych nagłówków, metod i kodów statusu, zgodnie z potrzebami konkretnych aplikacji. Jest to istotne w kontekście interakcji z dynamicznymi stronami internetowymi, które często ładują dodatkowe fragmenty strony (np. poprzez AJAX) po pierwotnym żądaniu HTTP.

W takich przypadkach, używanie narzędzi takich jak Selenium staje się kluczowe, tak jak wspomniano w opisie tego narzędzia.

4.9. HTML i CSS

HTML [13] (*HyperText Markup Language*) definiuje strukturę treści na stronie internetowej, opierając się na zestawie elementów, które wyznaczają sposób prezentacji i funkcjonowania różnych sekcji strony. Elementy HTML są znacznikami, które określają, jak zawartość powinna być wyświetlana na stronie internetowej.

Atrybuty HTML dostarczają dodatkowych informacji o zachowaniu i prezentacji elementów HTML. Są one zdefiniowane w znaczniku początkowym i zwykle składają się z par name/value, na przykład name="value". Powszechnie stosowane atrybuty obejmują:

- id - unikalny identyfikator elementu HTML.
- class - klasy do zastosowania do elementu.
- style - dodaje właściwości stylu do elementu.
- href - adres URL do którego prowadzi link.
- src - adres URL obrazu do wyświetlenia.
- alt - alternatywny tekst dla obrazu, gdy nie można go wyświetlić.
- value - wartość elementu wejściowego.
- name - nazwa elementu wejściowego.
- type - typ elementu wejściowego.

Atrybuty HTML modyfikują domyślne funkcje elementu lub zapewniają funkcje dla typów elementów, które nie mogą działać bez nich. Na przykład, atrybut "charset" określa kodowanie znaków dokumentu, a "disabled" wyłącza element interaktywny.

CSS (*Cascading Style Sheets*) [27] - język arkuszy stylów, który pozwala na kontrolowanie prezentacji i wyglądu dokumentów HTML. CSS pozwala na stylizację elementów, umożliwiając tworzenie atrakcyjnych i angażujących interfejsów użytkownika.

Kluczowe elementy CSS obejmują:

- Stylizacja: CSS pozwala na kontrolę nad układem, kolorami, czcionkami i innymi aspektami wizualnymi strony.
- Selektory: Selektory CSS są używane do kierowania stylów do określonych elementów HTML. Mogą być oparte na typach elementów, klasach, identyfikatorach, atrybutach czy relacjach między elementami.
- Kaskadowość: CSS wprowadza koncepcję kaskadowości, która pozwala na nakładanie wielu stylów na ten sam element, z zastosowaniem reguł dotyczących specyficzności i kolejności.
- Model pudełkowy: CSS wprowadza model pudełkowy, który określa, jak elementy są wyświetlane i rozmieszczone na stronie.

Selektory CSS są kluczowym elementem w procesie ekstrakcji danych ze stron internetowych. W tym kontekście, selektory CSS pozwalają na wybieranie elementów według ich atrybutów, umożliwiając precyzyjne lokalizowanie danych do ekstrakcji. Na przykład, selektor CSS może identyfikować wszystkie linki na stronie, które mają określoną klasę lub id, co umożliwia precyzyjne ekstrahowanie tych konkretnych danych.

4.10. REST

REST [28] (*Representational State Transfer*) to styl architektury sieciowej, który określa zestaw reguł i konwencji do tworzenia usług internetowych. Te usługi, znane jako usługi RESTful, są oparte na protokole HTTP i korzystają z jego metod (GET, POST, PUT, DELETE) do przesyłania i odbierania danych.

Kluczowe koncepcje w REST obejmują:

- Zasoby: W REST, zasoby to podstawowe jednostki, z którymi pracujemy. Każdy zasób ma unikalny identyfikator (URI), który jest używany do lokalizacji i manipulowania danym zasobem.
- Reprezentacje: Zasoby mogą być reprezentowane na różne sposoby, najczęściej w formatach JSON lub XML. Reprezentacja zasobu jest tym, co jest zwracane do klienta po zapytaniu o zasób.
- Bezstanowość: REST jest bezstanowy, co oznacza, że serwer nie przechowuje żadnych informacji o stanie klienta między poszczególnymi zapytaniami. Każde zapytanie jest samowystarczalne i zawiera wszystkie niezbędne informacje do przetworzenia zapytania.
- Komunikaty: Klient komunikuje się z serwerem poprzez wysyłanie komunikatów HTTP, które zawierają metody (takie jak GET, POST, itp.), które mówią serwerowi, co ma zrobić z zasobem.
- Cache'owanie: W celu poprawy wydajności, odpowiedzi z serwera mogą być cache'owane przez klienta. To oznacza, że klient może przechowywać kopię pewnych odpowiedzi, aby uniknąć konieczności ponownego zapytania o te same dane.

Usługi sieciowe RESTful są często wybierane ze względu na swoją prostotę i elastyczność w porównaniu do usług opartych na SOAP [13]. Dzięki użyciu standardowych protokołów HTTP są one łatwe do zrozumienia i mogą być używane przez różne technologie i platformy. Przykłady popularnych usług, które korzystają z RESTful API, obejmują Twitter, Google Maps, eBay i Amazon [29].

4.11. Postman

Postman [30] to uznane narzędzie do tworzenia i testowania interfejsów API, które usprawnia proces projektowania, dokumentowania i testowania interfejsów API. Zapewnia przyjazny dla użytkownika interfejs do wysyłania żądań HTTP, ułatwiając interakcję z usługami internetowymi i interfejsami API. Dzięki Postman programiści mogą tworzyć żądania, ustawiać nagłówki, przekazywać parametry i sprawdzać odpowiedzi, umożliwiając im wydajne testowanie i debugowanie interfejsów API. Postman oferuje również funkcje takie jak kolekcje do organizowania żądań, zautomatyzowane przepływy pracy testowania i możliwości współpracy. Jest to cenne narzędzie dla programistów i konsumentów API, które upraszcza zadania związane z tworzeniem, testowaniem i integracją API.

Zalety:

- Przyjazny dla użytkownika interfejs, umożliwiający łatwe tworzenie żądań HTTP i interakcję z usługami API.
- Organizacja żądań za pomocą kolekcji, ułatwiająca zarządzanie testowaniem i rozwojem API.
- Automatyzacja przepływów pracy testowania, zapewniająca powtarzalność i spójność wyników.
- Możliwość współpracy, umożliwiająca łatwe udostępnianie i współpracę nad testami API.

W kontekście tworzenia prototypu Postman jest szczególnie przydatny dla testowania punktów końcowych serwera Django, umożliwiając łatwe sprawdzanie powtarzalności skrapingu i debugowanie usługi serwera. Oprócz tradycyjnych zastosowań do testowania i rozwoju API może być również wykorzystywany w celu ułatwienia testowania i odtwarzalności danych uzyskanych za pomocą modeli uczenia maszynowego. Tworząc określone żądania i definiując pożądane dane wejściowe, można zasymulować i oceniać trafność modelu ML.

4.12. Webstorm

WebStorm [31] jest zaawansowanym środowiskiem programistycznym (IDE[9]) stworzonym przez firmę JetBrains. Jest szeroko stosowany przez programistów do tworzenia aplikacji webowych. WebStorm oferuje wiele funkcji, które ułatwiają rozwój, testowanie i debugowanie kodu. Środowisko udostępnia rozbudowane narzędzia do debugowania, które pozwalają programistom na śledzenie i analizowanie kodu podczas jego wykonania. Możliwe jest dzięki temu wykrywanie błędów oraz obserwacja wartości zmiennych. Szczególną cechą, która wyróżnia WebStorm, jest zdolność monitorowania interakcji z rozszerzeniem Chrome. Jedynie w tym środowisku programistycznym udało się skutecznie połączyć narzędzie z panelem w narzędziach deweloperskich przeglądarki. Debugowanie rozszerzeń przeglądarek jest niezwykle cenne podczas rozwijania aplikacji wykorzystujących API przeglądarek, takie jak manipulacja DOM, zarządzanie zdarzeniami czy komunikacja sieciowa.

WebStorm oferuje również funkcje, które ułatwiają ogólny proces tworzenia aplikacji webowych. IDE dostarcza narzędzia do automatycznego uzupełniania kodu, refaktoryzacji, wyszukiwania i analizowania kodu, co przyspiesza pracę programisty. Ponadto WebStorm zapewnia integrację z systemami kontroli wersji, takimi jak Git, co ułatwia zarządzanie kodem źródłowym i współpracę w zespole. IDE wspiera wiele popularnych języków programowania webowego, takich jak JavaScript, TypeScript, HTML, CSS i wiele innych. IDE oferuje inteligentne podpowiedzi składniowe, analizę statyczną kodu, narzędzia do testowania jednostkowego i wiele innych, co pomaga w tworzeniu bardziej wydajnego i niezawodnego kodu.

Dodatkowo WebStorm ma rozbudowany ekosystem w postaci pluginów i rozszerzeń, które można dostosowywać do indywidualnych potrzeb programistów. Dostępne są pluginy dla różnych frameworków i bibliotek, takich jak Angular, React, Vue.js, co ułatwia pracę z tymi technologiami.

4.13. Modele NLP z biblioteki spaCy

Rozwiązanie, przedstawione w ramach pracy, obejmuje wykorzystanie bibliotek lematyzacji z biblioteki spaCy [32]. Biblioteka ta zapewnia wstępnie wytrenowane modele dla różnych języków, które można wykorzystać do lematyzacji. W przypadku tej pracy został wykorzystany en_core_web_md i pl_core_web_md, które są odpowiednio średniej wielkości modelami języka angielskiego i polskiego.

Głównym celem lematyzacji jest przekształcenie słów w ich formę podstawową, czyli lematy. Dzięki temu procesowi, różne formy słów (takie jak odmienione formy czasowników czy rzeczowników) są sprowadzane do jednej reprezentacji, co ułatwia dalsze analizowanie i przetwarzanie tekstu. Ponadto ułatwia to porównywanie, grupowanie i analizowanie danych tekstowych. Jest szczególnie przydatna w zadaniach takich jak klasyfikacja tekstu, analiza sentymentu czy ekstrakcja informacji.

Modele en_core_web_md i pl_core_web_md, które są wykorzystywane w tym rozwiązaniu, posiadają zaawansowane rozumienie struktury morfologicznej słów. Dzięki temu są w stanie dokładnie przekształcić słowa do ich formy podstawowej, uwzględniając kontekst i znaczenie.

[9] IDE (ang. *Integrated Development Environment*) to zintegrowane środowisko programistyczne, które wspomaga pisanie kodu źródłowego i inne aspekty procesu tworzenia oprogramowania, takie jak testowanie i projektowanie graficznego interfejsu użytkownika.

Standardowy potok spaCy (w tym przypadku dla języka polskiego [33]) przedstawia się w następujący sposób:

- Tok2vec: Jest to podpotok, który stosuje serię transformacji, aby przekształcić tokeny (słowa) na wektory (numeryczne reprezentacje). Te wektory przechwytują semantyczne znaczenie słów i mogą być używane jako dane wejściowe dla innych komponentów w potoku.
- Morphologizer: Ten komponent jest odpowiedzialny za przewidywanie cech morfologicznych słów, takich jak czas dla czasowników, liczba dla rzeczowników i stopień dla przymiotników. Używa modelu statystycznego do przewidywania tych cech na podstawie kontekstu słowa w zdaniu.
- Parser: Komponent parser wykonuje analizę zależnościową, która polega na identyfikacji gramatycznych relacji między słowami w zdaniu. Może to pomóc zrozumieć strukturę zdania i jak słowa odnoszą się do siebie. Na przykład, może zidentyfikować, które słowa są podmiotami w zdaniu.
- Lemmatizer: Redukuje słowa do ich podstawowej lub słownikowej formy, znaną jako lemat. Może to pomóc uprościć język i zredukować złożoność danych tekstowych, grupując razem różne odmienione formy słowa.
- Tagger: Tagger przypisuje oznaczenia części mowy do każdego słowa, takie jak rzeczownik, czasownik, przymiotnik itp. Jest to realizowane za pomocą modelu statystycznego, który został wytrenowany na dużym zbiorze już oznaczonych danych tekstowych.
- Attribute_ruler: Ten komponent pozwala definiować reguły do ustawiania atrybutów tokenów na podstawie ich tekstu i innych atrybutów. Może to być przydatne do obsługi specyficznych przypadków lub wyjątków, które nie są objęte modelami statystycznymi.
- NER (*Named Entity Recognizer*) [34]: Identyfikuje nazwane jednostki w tekście, takie jak ludzie, organizacje, lokalizacje, daty itp. Używa modelu statystycznego do przewidywania jednostek na podstawie kontekstu słów w zdaniu.

Każdy z tych komponentów odgrywa kluczową rolę w przetwarzaniu i zrozumieniu danych tekstowych. Pracują razem w potoku, przy czym każdy komponent przyjmuje wynik poprzedniego komponentu jako swoje dane wejściowe. Pozwala to na skomplikowaną serię transformacji i analiz wykonywanych na danych tekstowych.

4.14. Model detekcji obiektów hustvl/yolos-tiny

Wykrywanie obiektów to zadanie z dziedziny wizji komputerowej, które polega na identyfikacji i lokalizacji obiektów na obrazach. W systemie wykorzystano model hustvl/yolos-tiny, odmianę algorytmu YOLO [35] (*You Only Look Once*), dostępnego w bibliotece Hugging Face's Transformers. Model ten ma za zadanie wykrywać obiekty na dostarczonych mu obrazach w trakcie zastosowywania modyfikatorów. Wybrano ten model ze względu na jego szybkość i efektywność, które są istotne dla systemu.

YOLO różni się od tradycyjnych metod detekcji obiektów. Zamiast skanować obraz wielokrotnie w poszukiwaniu obiektów, YOLO robi to tylko raz, stąd jego nazwa. YOLO działa poprzez podział obrazu na siatkę. Każda komórka siatki jest odpowiedzialna za przewidywanie obiektu, jeśli środek obiektu wypada na niej.

Podczas przewidywania, każda komórka siatki tworzy wiele prostokątów, które są potencjalnymi lokalizacjami obiektów. Każdy z tych prostokątów ma przypisany wynik pewności, który reprezentuje prawdopodobieństwo, że obiekt znajduje się w tym prostokącie i jak dobrze prostokąt pasuje do obiektu. Wraz z przewidywaniem prostokątów, każda komórka siatki przewiduje również, jaki typ obiektu znajduje się w prostokącie.

Na koniec, model YOLO generuje listę prostokątów z wynikami pewności i przewidywaniami klas. Prostokąty z niskimi wynikami pewności są odrzucane, a ostatecznym wynikiem są końcowe przewidywania modelu posortowane według pewności malejąco.

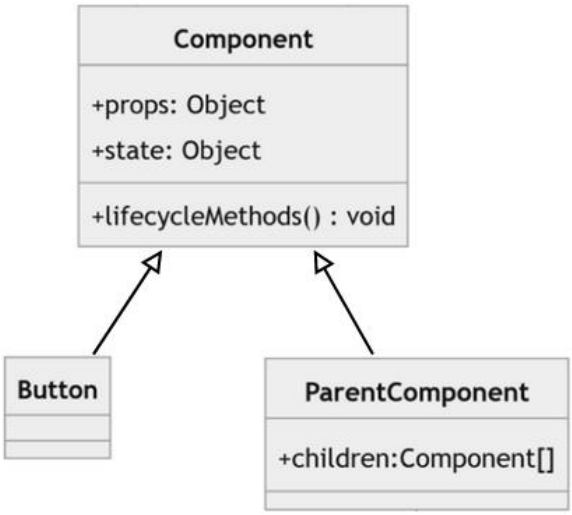
Dzięki swojej unikalnej architekturze i podejściu do detekcji obiektów YOLO jest w stanie przetwarzać obrazy szybko i efektywnie, co pozwala na oszczędność zasobów i czasu. W przypadku tego prototypu, wykorzystanie YOLO pozwoliło na zwiększenie wydajności systemu, jednocześnie utrzymując wysoką jakość wyników.

Hugging Face to platforma, która dostarcza implementacje wielu popularnych modeli uczenia maszynowego w przyjaznym dla użytkownika formacie. Modele Hugging Face mogą być łatwo integrowane w aplikacjach i wykorzystywane do zadań takich jak detekcja obiektów, przetwarzanie języka naturalnego i wiele innych.

Wprowadzenie modelu detekcji obrazów do projektu stanowi kluczowy element usprawnienia procesu zbierania obrazów w ramach skrapowania. Model ten umożliwia użytkownikowi automatyczne wykrywanie i selekcję istotnych obiektów na dostarczonych obrazach. Dzięki temu użytkownik może precyzyjnie określić, które obrazy zawierają interesujące go elementy, eliminując konieczność ręcznej selekcji lub analizy dużej ilości nieistotnych danych w procesie skrapowania. To nie tylko przyspiesza proces, ale także pozwala na gromadzenie bardziej wartościowych i spójnych danych.

4.15. Architektura oparta na komponentach

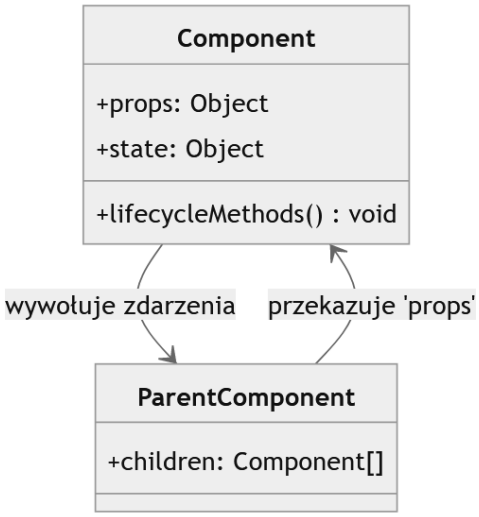
Architektura oparta na komponentach [36] (*z ang. „Component based architecture”*) polega na podziale części systemu w samodzielne jednostki lub komponenty, z których każdy ma swoje własne zadanie. Ten styl jest wykorzystywany w wytwarzaniu oprogramowania w nowoczesnych frameworkach frontendowych, takich jak React, Vue.js i Angular. Interfejs użytkownika jest podzielony na niezależne komponenty wielokrotnego użytku, z których każdy zarządza własnym stanem i właściwościami. Komponenty mogą być postrzegane jako funkcje, które pobierają dane wejściowe (w React zwane "*props*") i zwracają element interfejsu użytkownika. Komponenty mogą również posiadać metody cyklu życia, które pozwalają na uruchamianie kodu w określonych punktach cyklu życia komponentu, na przykład, gdy jest on po raz pierwszy wstawiany do DOM lub tuż przed jego usunięciem. Na rysunku 7. pokazana jest podstawowa struktura komponentów i ich dziedziczenie.



Rysunek 7. Diagram przedstawiający dziedziczenie komponentów React i ich podstawowe elementy. Źródło: opracowanie własne.

Inną korzyścią jest wyraźne rozdzielenie obowiązków. Każdy komponent jest odpowiedzialny za pojedynczą funkcjonalność. Zachęca to również programistów do myślenia w sposób modularny, rozkładając skomplikowane interfejsy użytkownika na mniejsze, łatwiejsze do zarządzania

komponenty. Dzięki temu kod jest łatwiejszy do zrozumienia, przetestowania i utrzymania. Architektura oparta na komponentach ułatwia również lepszą organizację kodu, z komponentami zorganizowanymi hierarchicznie. W aplikacji React architektura ta jest połączona z jednokierunkowym przepływem danych, w którym komponenty nadrzędne przekazują właściwości swoim komponentom podrzędnym. Jeśli komponent podrzędny musi zmienić dane, wysyła zdarzenie do komponentu nadrzędnego, który dokonuje zmiany. Na rysunku 8. widoczny jest diagram ze sposobem komunikacji podrzędnego i nadrzędnego komponentu.



Rysunek 8. Diagram przedstawiający sposób komunikacji pomiędzy komponentami React. Źródło: opracowanie własne.

4.16. Webpack

Webpack [37] jest narzędziem wykorzystywanym w procesie budowania i pakowania aplikacji webowych. Jego głównym zadaniem jest zarządzanie zależnościami oraz transformacja różnych typów plików źródłowych, takich jak JavaScript, CSS, obrazy i inne. Działa on jako zaawansowany modułowy *bundler*[10], umożliwiający programistom efektywne ładowanie strony poprzez utworzenie zoptymalizowanego zestawu plików/pliku wyjściowego, który zawiera wszystkie zależności aplikacji w jednym miejscu.

Webpack oferuje również wsparcie dla innych zaawansowanych funkcji TypeScript, takich jak importowanie i korzystanie z modułów, zarządzanie typami, generowanie deklaracji typów i wiele innych. Dodatkowo Webpack umożliwia konfigurację specjalnych reguł i loaderów do obsługi plików TypeScript, takich jak ts-loader [38], który przekształca kod TypeScriptu na JavaScript. To pozwala na płynne integrowanie TypeScriptu z resztą projektu i zapewnia łatwe budowanie i importowanie jako rozszerzenia Chromium aplikacji.

Narzędzie obsługuje także innowacyjne funkcje, takie jak rozszczepianie kodu (*code splitting*) i opóźnione ładowanie, które przyczyniają się do poprawy wydajności i szybkości ładowania aplikacji. Dodatkowo Webpack integruje się z narzędziami deweloperskimi, takimi jak „webpack-dev-server”, umożliwiając automatyczne odświeżanie przeglądarki podczas edycji plików.

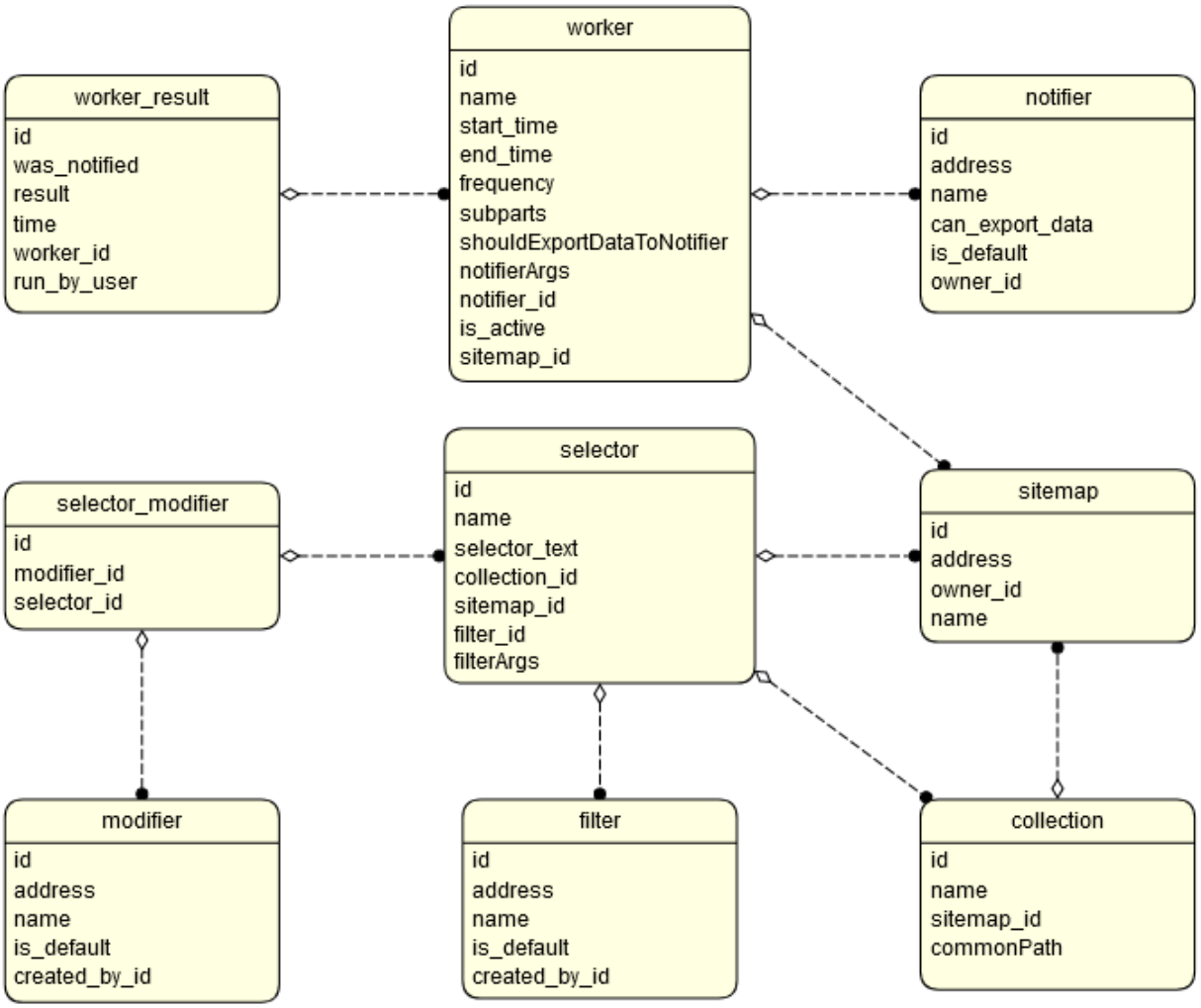
[10] Bundler (ang. „*module bundler*”) to narzędzie służące do łączenia różnych modułów lub plików źródłowych w pakiet, który może być łatwo załadowany i uruchomiony w środowisku wykonawczym, takim jak w tym przypadku przeglądarka internetowa.

5. Implementacja prototypu

W niniejszym rozdziale przedstawiono szczegóły dotyczące implementacji prototypu narzędzia do pozyskiwania danych ze stron internetowych. Opisano modele bazy danych, implementację części klienckiej, która jest rozszerzeniem do przeglądarek opartych na Chromium i część serwerową odpowiedzialną za przetwarzanie żądań, wykonywanie skrapowania, zarządzanie harmonogramami.

5.1. Model danych

Modele służą do przechowywania i organizowania danych, które odnoszą się do map witryny, kolekcji, filtrów, modyfikatorów, selektorów, notyfikatorów, robotników (*workers*) oraz wyników ich działania. Każdy model ma określone role i relacje, które pozwalają na skuteczne zarządzanie procesem skrapowania i przetwarzania danych. Relacje poszczególnych encji widoczne są na rysunku 9.



Rysunek 9. Diagram związków encji rozwiązania w bazie danych. Źródło: opracowanie własne.

Opis każdego z modeli:

- `Sitemap`: reprezentuje mapę strony należącą do użytkownika, posiadającą unikalny adres URL i nazwę. Każda mapa strony jest związana z konkretnym użytkownikiem.
- `Selector`: jest związany z mapą strony i opcjonalnie z kolekcją. Reprezentuje on selektor CSS. Może być powiązany z wieloma modyfikatorami i pojedynczym filtrem.

- `Collection`: kolekcja jest związana z mapą strony i reprezentuje zestaw selektorów CSS podobnych elementów na stronie internetowej. Selektory te muszą zawierać w sobie wspólny selektor `'commonPath'` z tej encji.
- `Filter`: stosuje określony warunek do pobranych przy użyciu selektora danych. Model ten ma niestandardową metodę `filter (self, data, args)`, gdzie stosuje ona zdefiniowany warunek (taki jak *contains, equals, greater than, less than*) do danych wejściowych.
- `Modifier`: reprezentuje funkcję lub operację modyfikującą zeskrapowane dane. Zawiera zestaw predefiniowanych operacji takich jak *UPPERCASE, lowercase, Remove interpunctions* itp. Każdy modyfikator może również wskazywać na niestandardową operację hostowaną pod innym adresem URL (w polu `address`).
- `SelectorModifier`: jest to model pośredniczący dla relacji wiele-do-wielu pomiędzy selektorami a modyfikatorami. Do jednego selektora może być przypisane wiele modyfikatorów, aby zmieniać dane kilkukrotnie.
- `Notifier`: reprezentuje sposób powiadamiania użytkownika o zdarzeniu spełnienia warunku przez robotnika. Powiadomienia mogą być wysyłane za pośrednictwem e-maila lub dowolnego innego punktu końcowego API (określonego w atrybucie `address`).
- `Worker`: reprezentuje zadanie skrapowania. Jest związany z użytkownikiem i mapą strony, ma nazwę, określa częstotliwość (w minutach) i może opcjonalnie wskazywać na `notifier`, który będzie używany do powiadamiania użytkownika o spełnieniu warunku w trakcie skrapowania. Zawiera też pole typu boolean `is_active` wskazujące, czy zadanie jest aktualnie aktywne.
- `WorkerResult`: Przechowuje wynik wykonania zadania przez robotnika. Jest związany z pracownikiem i selektorem, a także zawiera wynik zadania skrapowania, czas jego wykonania oraz wartość boolean wskazującą, czy został manualnie uruchomiony przez użytkownika.

Zaprezentowane modele umożliwiają użytkownikom definiowanie i organizowanie różnych elementów, warunków, modyfikacji i powiadomień związanych z procesem skrapowania. Ich odpowiednie powiązania i relacje zapewniają spójność i elastyczność w konfiguracji oraz wykonywaniu prac skrapowania. Dzięki temu użytkownicy mogą łatwo tworzyć, modyfikować i kontrolować zadania skrapowania, dostosowując je do swoich specyficznych potrzeb i wymagań. Model danych stanowi podstawę dla implementacji systemu do skrapowania stron internetowych, umożliwiając efektywne zarządzanie i przetwarzanie danych w sposób intuicyjny i elastyczny.

5.2. Graficzny interfejs użytkownika – rozszerzenie przeglądarki

Kliencka część systemu została skonstruowana z wykorzystaniem technologii React i TypeScript. Integracja tych dwóch narzędzi umożliwia tworzenie dynamicznego interfejsu użytkownika z silnie typowanym kodem. Dzięki wykorzystaniu architektury bazującej na komponentach React pozwala na wielokrotne użycie kodu oraz modularność, co sprzyja czytelności kodu i łatwości w utrzymaniu aplikacji. Natomiast TypeScript wprowadza statyczne typy, zapewniając wyższą jakość kodu oraz wykrywanie błędów już na etapie rozwoju.

Rozwój i debugowanie rozszerzenia odbywały się przy użyciu narzędzia WebStorm, wszechstronnego środowiska programistycznego przeznaczonego dla języka JavaScript i jego różnych frameworków, w tym React i TypeScript. WebStorm okazał się niezastąpiony dzięki swojemu unikalnemu zestawowi funkcji, w tym możliwości wyboru panelu narzędzi deweloperskich rozszerzeń w Chromium jako celu debugowania.

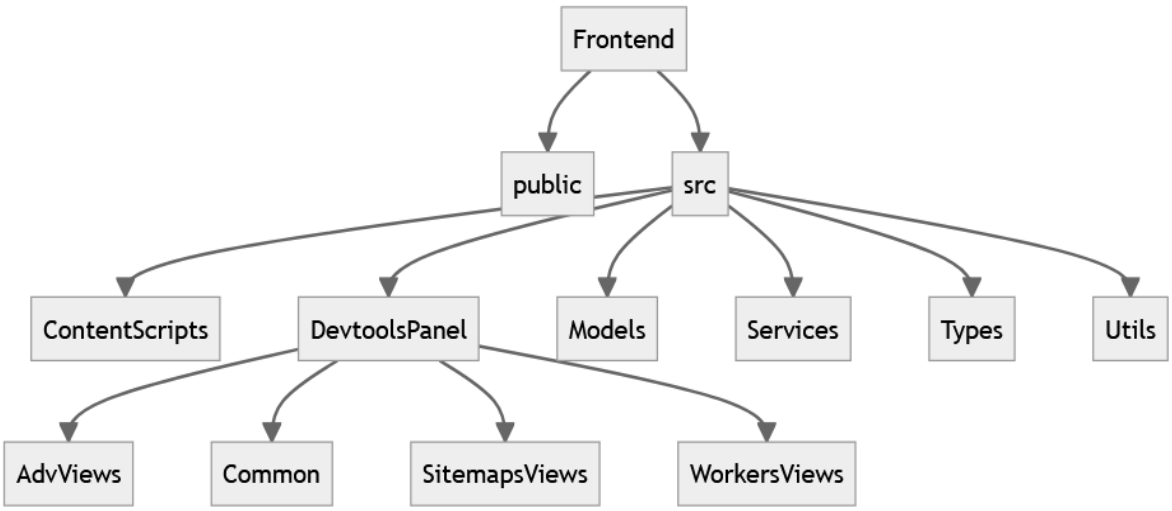
Do uproszczenia tworzenia interfejsu użytkownika została wykorzystana popularna biblioteka React-Bootstrap, oferująca wiele gotowych komponentów, takich jak okna modalne, przyciski czy formularze, które są powszechnie używane w całej aplikacji. Biblioteka zapewnia nie tylko spójność wyglądu interfejsu, ale także znacznie przyspiesza proces rozwoju.

5.2.1. Struktura projektu rozszerzenia

Rozszerzenia chromium [39] składają się z komponentów takich jak:

1. Popup - odpowiada za wyświetlanie interaktywnego wyskakującego okna dla użytkownika po kliknięciu ikonki rozszerzenia w prawym górnym rogu przeglądarki. Może zawierać interfejs użytkownika i umożliwiać interakcję z rozszerzeniem. W ramach tego projektu Popup nie pełni jednak żadnej funkcji poza informowaniem użytkownika, że do użycia rozszerzenia musi wejść do narzędzi deweloperskich przeglądarki.
2. Background Script - skrypt tła pełni rolę zarządcy rozszerzenia w trakcie, gdy nie jest aktywnie używane i nie potrzebuje dostępu do zawartości stron. Zazwyczaj rolą tego komponentu jest wykonywanie zadań w tle, czyli np. nasłuchuje zdarzeń, obsługuje żądania od innych komponentów oraz zarządza logiką biznesową rozszerzenia.
3. Content Scripts - skrypty treści w trakcie wykonywania mają dostęp do wczytanych stron internetowych i pozwalają na manipulację dokumentów oraz na interakcję z zawartością strony, np. zmienianie stylu, przechwytywanie zdarzeń i komunikację z innymi częściami składowymi rozszerzenia. To właśnie ten komponent pozwala na podświetlanie elementów na stronie i ekstrakcję selektorów.
4. Devtools Panel - panel narzędzi deweloperskich. Zazwyczaj dostarcza programistom narzędzi do inspekcji, debugowania oraz analizy struktury i zachowania stron internetowych. Każda zakładka w tym panelu jest wyświetlana jako strona HTML i rozszerzenia przeglądarki mogą wprowadzać własne zakładki do tego panelu. Takie podejście zostało też zastosowane w ramach tej pracy – do panelu devtools została dodana kolejna zakładka z większością interfejsu użytkownika.
5. Native Messaging - mechanizm ten umożliwia komunikację między rozszerzeniem Chrome a zewnętrzną natywną aplikacją, co pozwala na wykonywanie zadań, których rozszerzenie nie może samodzielnie realizować, poprzez przekazywanie danych między nimi. Brak zastosowania w tej pracy.
6. Storage - umożliwia przechowywanie danych rozszerzenia, takich jak preferencje użytkownika, ustawienia, historię itp., które mogą być odczytywane i zapisywane przez różne komponenty rozszerzenia. W przypadku tego rozwiązania wykorzystywany jest do przechowywania danych sesji zalogowanego użytkownika.
7. Message Passing - pozwala na przesyłanie komunikatów między różnymi komponentami rozszerzenia, umożliwiając wymianę danych, powiadamianie o zdarzeniach, przekazywanie poleceń i synchronizację działań pomiędzy nimi.

Każdy komponent należy traktować jako oddzielny byt, który, aby przesłać informacje do drugiego, musi wykorzystać API przeglądarki. Z tego względu struktura kodu została podzielona na foldery odpowiadające użytym komponentom w ramach rozszerzenia. Struktura katalogów znajduje się na rysunku 10.



Rysunek 10. Struktura katalogów w części klienckiej. Źródło: opracowanie własne.

Folder `public` zawiera pliki, które są tylko dla rozszerzenia i nie powinny być kompilowane razem z resztą plików TypeScript. Są to na przykład proste skrypty JS, zdjęcia, ikony, plik konfiguracyjny Manifest. Pliki z tego folderu są kopiowane do folderu wyjściowego bez zmian.

Folder `src` to główny katalog, w którym znajduje się cały kod TypeScript. Zawiera on kilka ważnych podkatalogów i plików:

- `ContentScripts` - zawiera skrypty, które są wstrzykiwane do stron odwiedzanych przez użytkownika.
- `DevtoolsPanel` - zawiera większość interfejsu klienta - wszystkie widoki, które znajdują się w panelu narzędzi deweloperskich rozszerzenia. Jest on dalej podzielony na kilka podfolderów:
    - `AdvViews` - zawiera widoki przeznaczone dla użytkowników z umiejętnościami programistycznymi. Tu znajduje się implementacja widoków dla interfejsu niestandardowych modyfikatorów i notyfikatorów.
    - `Common` - zawiera wspólne komponenty lub narzędzia, które są używane w wielu widokach w `DevtoolsPanel`. Obejmuje to wspólne układy, współdzielone style, funkcje narzędziowe i wspólne kontrolki typu przyciski.
    - `SitemapsViews` - zawiera widoki związane z mapami strony. Te widoki wyświetlają dane mapy strony, umożliwiają użytkownikowi nawigację po mapie strony lub dostarczają funkcjonalności związane z manipulacją lub analizą map strony.
    - `WorkersViews` - zawiera widoki, które są związane z robotnikami. Te widoki wyświetlają informacje o aktywnych pracownikach, pozwalają na definiowanie nowych pracowników i umożliwiają użytkownikowi edycję ich atrybutów i ustawień.
- `Models` - zawiera modele danych i struktur wykorzystywanych przy komunikacji z serwerem.
- `Services` - zawiera różne usługi, z których korzysta rozszerzenie. Są to na przykład instancje nadzorujące status zalogowania użytkownika.
- `Types` - (ze względu na wykorzystanie TypeScript) ten katalog zawiera definicje typów używane w rozszerzeniu. Te typy pomagają zapewnić, że komponenty i usługi są używane poprawnie, wskazując błędy na etapie kompilacji.
- `Utils` - zawiera pomocnicze skrypty, które są używane w całym rozszerzeniu. Są to funkcje do formatowania danych, walidacji lub jakiekolwiek inne rodzaje wspólnej logiki, która nie należy do konkretnego komponentu.

5.2.2. Wybór elementów na stronie

Proces ten wykorzystuje interfejs Modelu Obiektowego Dokumentu (DOM), który reprezentuje strukturę dokumentów HTML i XML. Dzięki przeglądaniu DOM, rozszerzenie może uzyskać dostęp i manipulować elementami HTML według potrzeb. Wykorzystując te możliwości w ramach pracy zaimplementowano interfejs umożliwiający zaznaczanie składników HTML bezpośrednio na wczytanej stronie.

Komunikacja pomiędzy skryptami dokumentu strony (*Content scripts*) a rozszerzeniem w narzędziach deweloperskich przeglądarki odbywa się z wykorzystaniem wbudowanych w Chromium funkcji takich jak np. `chrome.runtime.onMessage.addListener` i `chrome.tabs.sendMessage` [40].

Gdy użytkownik zdecyduje, że chce rozpocząć wybór komponentów na stronie, z których chce sczytywać dane i wciśnie przycisk, który ma mu to umożliwić, Content Script otrzyma powiadomienie przekazane przez API przeglądarki. Na podstawie tego powiadomienia, Content Script dodaje do styli strony nowy styl CSS, który uniemożliwia interakcję ze stroną. Dzięki temu na przykład, podczas zaznaczania linków do skrapowania, użytkownik nie przechodzi na stronę z linku. Dodatkowo Content Script dodaje styl, który jest aktywowany, gdy użytkownik najedzie kursorem na komponenty HTML (*onHover*). Dzięki temu użytkownik widzi, który element aktualnie wybiera. Podobny schemat działania jest obecny, gdy użytkownik chce włączyć podgląd wybranych już komponentów.

Dodatkowo gdy użytkownik chce zaznaczyć nowy komponent, Content Script dodaje na dole strony pasek narzędzi. Na tym pasku pojawiają się informacje o zaznaczonym elemencie HTML oraz przycisk *„Accept”*. Po naciśnięciu tego przycisku wybór jest zatwierdzany, wcześniej dodane style są wyłączane, a informacja zwrotna z wybranym komponentem i jego selektorem jest przekazywana do części panelu narzędzi deweloperskich (*Devtools*).

5.2.1. Tworzenie selektorów CSS do skrapowania stron internetowych

Aby przeglądać Model Obiektowy Dokumentu (DOM) i wybierać konkretne elementy HTML na stronie internetowej, wykorzystuje się selektory CSS. Są to wzorce używane do wyboru elementów, którym chcemy nadać styl lub w tym przypadku, z których chcemy pozyskać dane. Narzędzie do skrapowania umożliwia użytkownikom tworzenie i dostosowywanie tych selektorów w celu precyzyjnego określania miejsca, z którego chcą pozyskać dane.

Selektory CSS mogą być proste i oparte na typach elementów, identyfikatorach, klasach lub atrybutach. Na przykład, selektor *‘div’* wybierze wszystkie elementy div w dokumencie, selektor `'.nazwaKlasy'` wybierze wszystkie elementy o klasie „`nazwaKlasy`”, a `'#idElementu'` wybierze element o identyfikatorze "`idElementu`".

Jednak w przypadku bardziej skomplikowanych i precyzyjnych wyborów można użyć pseudo-klas CSS, takich jak `':nth-child(n)'`. Selektor ten dopasowuje każdy element, który jest n-tym dzieckiem swojego rodzica, niezależnie od jego typu. Dzięki temu można wybrać konkretne elementy wewnątrz większej grupy [17].

W kontekście narzędzia do skrapowania, selektory CSS są tworzone, gdy użytkownik wybiera elementy na stronie internetowej. Narzędzie przechodzi wtedy w górę drzewa DOM od wybranego elementu do elementu nadrzędnego, tworząc „ścieżkę”, która stanowi selektor CSS. Na przykład, jeśli użytkownik wybiera drugie dziecko komponentu `div`, system wygeneruje selektor takiego rodzaju `div:nth-child(2)`.

Listing 3. prezentuje funkcję odpowiedzialną za tworzenie selektora CSS na podstawie wybranego komponentu HTML. Główna pętla skryptu przechodzi w górę drzewa DOM, dodając kolejne elementy do listy z odpowiednimi oznaczeniami `:nth-child()`, a następnie łączy powstałą listę w selektor wstawiając pomiędzy poszczególne elementy znak „ > ”

Listing 3. Funkcja TS odpowiedzialna za wydobywanie selektora CSS na podstawie wybranego elementu HTML na stronie. Źródło: opracowanie własne.

```
function getCSSPath(el: Element): { pathString: string |
undefined, pathArr: string[] }
{
    if (!(el instanceof Element))
        return {pathString: "", pathArr: []};
    const path: string[] = [];
    while (el.nodeType === Node.ELEMENT_NODE) {
        let selector = el.nodeName.toLowerCase();
        const siblings = Array.from(el.parentNode!.children);
        const index = siblings.indexOf(el);
        selector += ":nth-child(" + (index + 1) + ")";
        path.unshift(selector);
        el = el.parentNode as Element;
    }
    return {pathString: path.join(" > "), pathArr: path};
}
```

5.2.2. Tworzenie selektorów CSS dla kolekcji elementów

Podczas pracy z kolekcjami elementów, proces skrapowania staje się bardziej złożony. Kolekcja to zbiór podobnych elementów na stronie internetowej, takich jak pozycje na liście czy komórki w tabeli.

Narzędzie do skrapowania, aby efektywnie funkcjonować z kolekcjami, musi generować dokładny selektor CSS, który wybiera wszystkie elementy w kolekcji, nie obejmując żadnych innych. Proces ten zawiera stworzenie selektora CSS dla reprezentatywnego elementu w kolekcji, a następnie uogólnienie go, aby objąć wszystkie elementy z kolekcji.

Przykładowo, jeśli mamy listę aktorów na stronie filmu, gdzie każdy aktor jest reprezentowany przez element `<div>`, zawierający jego imię i datę urodzenia, narzędzie może wygenerować selektor CSS oparty na ścieżkach, taki jak: `div:nth-child(1) > div:nth-child(2)`. W tym przypadku wygenerowany selektor odnosi się do konkretnej pozycji w strukturze drzewa DOM, wykorzystując indeksy `:nth-child()`. Ten selektor precyzyjnie wskazuje na elementy z imieniem i datą urodzenia pierwszego aktora na liście.

Podczas definiowania kolekcji wykorzystane jest wyrażenie regularne „`/#\w+|\[\w+\]|:nth-child\(\d+\)/g`” jako krok pośredni. Służy do ekstrakcji ogólnego selektora, który pozwoli podświetlić wszystkie dostępne elementy, które mogą być częścią kolekcji z komponentem wybranym na początku. Taki selektor jest zbyt ogólny i służy jedynie do podświetlenia potencjalnych elementów w kolekcji.

Użytkownik musi następnie wybrać kolejny podświetlony element, który posłuży jako punkt odniesienia do tworzenia ogólnego selektora kolekcji. Na podstawie tego wybranego elementu oraz pierwotnego selektora, narzędzie jest w stanie wygenerować bardziej precyzyjny selektor, który dokładnie określa składowe kolekcji.

W listingu 4. Zaprezentowana jest funkcja TypeScript odpowiedzialna za łączenie dwóch selektorów w jeden wskazujący powtarzające się elementy. Dodatkowo zwracany jest argument `commonPart` będący częścią wspólną dla tej kolekcji. Zostanie on zapisany w nowo utworzonym obiekcie `Collection` w celu ułatwienia dodawania do niej kolejnych zbiorów elementów.

Listing 4. Funkcja TS odpowiedzialna za łączenie selektorów w wspólny selektor umożliwiający wskazywanie kolekcji elementów HTML. Źródło: opracowanie własne.

```
function mergeSelectors(selector1: string, selector2: string): {
collectionSelector: string , commonPart: string } {

        const path1Arr = selector1.split(" > ")
        const path2Arr = selector2.split(" > ")
        const collectionSelector: string[] = [];
        const commonPart : string[] = [];
        let endOfCommon = false;

        for (let i = 0; i < path2Arr.length; i++) {
            if (path2Arr[i] === path1Arr[i]){
                if (!endOfCommon){
                    commonPart.push(path2Arr[i])
                }
                collectionSelector.push(path2Arr[i])
            } else {
                endOfCommon = true;

                let genericPart = path2Arr[i]
                .replace(/#\w+|\[\w+\]|:nth-child\(\d+\)/g, "")

                commonPart.push(genericPart)
                collectionSelector.push(genericPart)
            }
        }
    return {collectionSelector: collectionSelector.join(" > "),
commonPart: commonPart.join(" > ")}
}
```

5.2.3. Komunikacja z serwerem i uwierzytelnianie użytkowników

Rozszerzenie komunikuje się z serwerem za pomocą protokołu HTTP. Ta komunikacja obejmuje wysyłanie żądań do skrobania i odbieranie danych zebranych ze skanowanych stron, a także umożliwia realizację innych interakcji.

Proces autentykacji użytkowników odbywa się poprzez przechowywanie informacji o logowaniu (*Bearer Token*) w lokalnym magazynie rozszerzenia (*local storage*), co zapewnia kontynuację sesji użytkownika tak długo, jak przeglądarka pozostaje aktywna i klucz wygenerowany przez serwer nie wygaśnie. Warto podkreślić, że te dane są izolowane od pozostałej części stron internetowych, co gwarantuje bezpieczeństwo informacji uwierzytelniających użytkownika.

Warstwa komunikacji wykorzystuje AJAX, co umożliwia asynchroniczną interakcję aplikacji klienckiej z serwerem, poprawiając tym samym komfort użytkowania poprzez eliminowanie konieczności odświeżania stron rozszerzenia podczas wymiany danych.

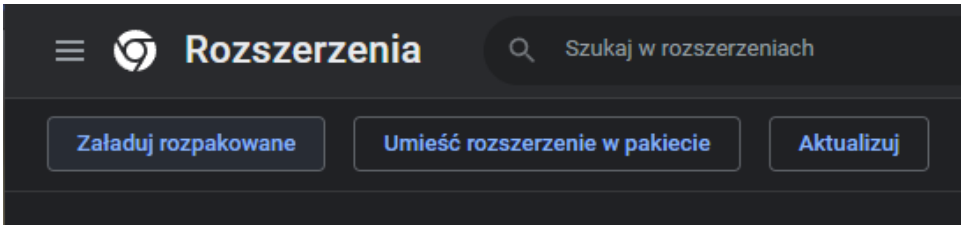
5.2.4. Budowanie rozszerzenia i importowanie do Chrome

Aby wdrożyć rozszerzenie, kod jest kompilowany do JavaScript i pakowany w formacie zrozumiałym dla przeglądarki. Rozszerzenie oprócz kodu musi zawierać plik manifestu (w tym przypadku w wersji v3), który jest plikiem w formacie JSON zawierającym metadane dotyczące rozszerzenia, takie jak nazwa, wersja, uprawnienia oraz skrypty, które muszą zostać uruchomione. Treść pliku manifest zaprezentowana jest w listingu 5.

Listing 5. Treść pliku *manifest.json* rozszerzenia. Źródło: opracowanie własne.

```
{
  "name": "Smart Scrapper",
  "description": "Scrap pages however you like",
  "version": "1.0",
  "host_permissions": [ "<all_urls>" ],
  "permissions": [
    "tabs",
    "notifications",
    "storage",
    "unlimitedStorage",
    "downloads"
  ],
  "devtools_page": "DevtoolsPanel/devtools.html",
  "manifest_version": 3,
  "action": {
    "default_popup": "Popup.html",
    "default_icon": "128.png"
  },
  "content_scripts": [
    {
      "matches": ["<all_urls>" ],
      "css": [ "ContentScripts/ElementSelector.css" ],
      "js": [
        "ContentScripts/ContentManager.js",
        "Utils/CssSelector.js",
        "ContentScripts/ContentSelector.js"
      ],
      "run_at": "document_idle",
      "all_frames": true
    }
  ],
  "background": {
    "service_worker": "Background/BackgroundWorker.js"
  }
}
```

Aby zaimportować rozszerzenie do przeglądarki Chrome, użytkownik musi przejść do strony rozszerzeń w przeglądarce, włączyć tryb programisty i załadować rozpakowane rozszerzenie (po kliknięciu przycisku *„załaduj rozszerzenie”* widocznego na rysunku 11, wskazując katalog z zapakowanym kodem. Jest to rozwiązanie przyjęte na okres pracy nad rozwiązaniem – docelowo rozszerzenie powinno być dostępne z poziomu sklepu z rozszerzeniami.



Rysunek 11. Ładowanie rozpakowanego rozszerzenia do chrome. Źródło: [7]

5.3. Usługa serwerowa z funkcją skrapowania

Usługa serwerowa składa się z dwóch głównych komponentów: podstawowego – odpowiedzialnego za api i podstawową logikę oraz z asynchronicznego, który jest odpowiedzialny za pozyskiwanie danych w tle.

5.3.1. Struktura projektu

W projekcie Django wykorzystującym Celery do przetwarzania zadań asynchronicznych struktura kodu jest zazwyczaj organizowana w taki sposób, aby rozdzielić obowiązki i promować modularność oraz skalowalność. Poniżej przedstawiono krótki opis głównych komponentów:

- Projekt Django - to główny kontener dla aplikacji. Zawiera on ustawienia, konfigurację URL i wszelki kod specyficzny dla aplikacji. W tym przypadku jest to reprezentowane przez katalog `'backend'`, zawiera on: pliki takie jak:
    - `'settings.py'` zawiera wszystkie konfiguracje projektu Django.
    - `'urls.py'` to miejsce, gdzie jest definiowane routing URL dla projektu.
    - `'celery.py'` to miejsce, gdzie jest inicjalizowana aplikacja Celery i ładowana jest jej konfiguracja z ustawień Django.
- Aplikacje Django - Django zachęca do dzielenia projektu na indywidualne aplikacje, które są pakietami Pythona, które podążają za pewnymi konwencjami. Aplikacje Django mogą być wielokrotnie używane w różnych projektach. W tym przypadku aplikacja to `'scrapper_api'`. Aplikacja zawiera:
    - `'models.py'`, gdzie są definiowane modele danych.
    - `'views.py'`, gdzie są definiowane widoki, które obsługują żądania HTTP.
    - `'admin.py'`, gdzie jest definiowany interfejs administracyjny dla aplikacji.
    - `'tasks.py'`, gdzie są definiowane zadania Celery. Są to funkcje, które mają być uruchamiane asynchronicznie.
- Celery - Celery to rozproszona kolejka zadań, która pozwala na asynchroniczne uruchamianie zadań wymagających dużych zasobów obliczeniowych lub czasochłonnych. W projekcie Django zazwyczaj zadania Celery są definiowane w pliku `'tasks.py'`. Zadania te mogą być następnie wywoływane z widoków lub modeli bez blokowania wykonania reszty aplikacji.
- Migracje - Django dostarcza abstrakcyjne API bazy danych, które pozwala na tworzenie, pobieranie, aktualizowanie i usuwanie rekordów. Gdy są dokonywane zmiany w modelach danych, Django generuje plik migracji (w katalogu `'migrations'`), który opisuje, jak zmodyfikować schemat bazy danych, aby odpowiadał obecnemu stanowi modeli.
- Utils - plik `'utils.py'` zawiera funkcje i klasy pomocnicze, które są używane w różnych częściach aplikacji.
- Manage.py - to narzędzie konsolowe, które umożliwia interakcję z projektem Django na różne sposoby, takie jak uruchamianie serwera deweloperskiego, testów, tworzenie lub stosowanie migracji itp.

5.3.2. Część podstawowa warstwy serwerowej

Usługa ta udostępnia szereg punktów końcowych (*endpoints*), które umożliwiają zarządzanie różnymi aspektami systemu. Każdy z nich jest dostępny poprzez określony URL i obsługuje odpowiednie metody HTTP (GET, POST, PUT, DELETE) do manipulowania danymi. Punkty dostępowe te są zaimplementowane za pomocą Django REST Framework, a każdy z nich jest klasą dziedziczącą po `viewsets.ModelViewSet`, co zapewnia efektywne i łatwe zarządzanie danymi.

Zgodnie z konwencją Django, terminy „widoki” (*Views*) lub „kontrolery widoków” (*View Controllers*) są częściej używane do opisu funkcji lub klas, które obsługują żądania HTTP i zwracają odpowiedzi HTTP. Punkty dostępu są zwykle definiowane przez adresy URL, które są mapowane na te widoki.

Zatem część podstawowa udostępnia następujące widoki API:

- SitemapView: Ten widok umożliwia zarządzanie mapami witryn. Możemy tworzyć, odczytywać, aktualizować i usuwać mapy witryn. Dostęp do tego widoku jest możliwy poprzez URL `'/sitemap/'`.
- CollectionView: Ten widok umożliwia zarządzanie kolekcjami. Możemy tworzyć, odczytywać, aktualizować i usuwać kolekcje. Dostęp do tego widoku jest możliwy poprzez URL `'/collection/'`.
- SelectorView: Ten widok umożliwia zarządzanie selektorami. Możemy tworzyć, odczytywać, aktualizować i usuwać selektory. Dostęp do tego widoku jest możliwy poprzez URL `'/selector/'`.
- NotifierView: Ten widok umożliwia zarządzanie powiadomieniami. Możemy tworzyć, odczytywać, aktualizować i usuwać powiadomienia. Dostęp do tego widoku jest możliwy poprzez URL `'/notifier/'`.
- ModifierView: Ten widok umożliwia zarządzanie modyfikatorami. Możemy tworzyć, odczytywać, aktualizować i usuwać modyfikatory. Dostęp do tego widoku jest możliwy poprzez URL `'/modifier/'`.
- FilterView: Ten widok umożliwia zarządzanie filtrami. Możemy tworzyć, odczytywać, aktualizować i usuwać filtry. Dostęp do tego widoku jest możliwy poprzez URL `'/filter/'`.
- WorkerView: Ten widok umożliwia zarządzanie robotnikami. Możemy tworzyć, odczytywać, aktualizować i usuwać robotników. Dostęp do tego widoku jest możliwy poprzez URL `'/worker/'`.
- UserList: Ten widok umożliwia zarządzanie użytkownikami. Możemy tworzyć, odczytywać, aktualizować i usuwać użytkowników. Dostęp do tego widoku jest możliwy poprzez URL `'/users/'`.

5.3.3. Część asynchroniczna – odpowiedzialna za skanowanie „w tle”

Komponent asynchroniczny projektu odgrywa kluczową rolę w zarządzaniu zadaniami skrapowania wykonywanymi w tle. Jego rdzeniem jest Celery - system kolejkowania zadań stworzony dla Pythona, który specjalizuje się w efektywnym zarządzaniu zadaniami w tle.

Listing 6. prezentuje skrypt Python wywoływany przez Celery Beat cyklicznie. Odpowiedzialny jest za uruchomienie wszystkich robotników, które powinny uruchomić się zgodnie z harmonogramu.

Listing 6. Skrypt Python wywoływane cyklicznie przez Celery Beat. Źródło: opracowanie własne.

```
@app.task
def scrapeAll():
    from scrapper_api.models import Worker
    from scrapper_api.tasks import scrape_worker
    allWorkers = Worker.objects.all()
    for worker in allWorkers:
        if (is_current_datetime_valid
           (worker.start_time,
            worker.end_time,
            worker.frequency)
          and worker.is_active):
            scrape_worker(worker=worker,scrappedByUser=False)
```

Na najbardziej podstawowym poziomie, proces, za który odpowiedzialny jest komponent asynchroniczny, obejmuje następujące kroki:

1. Pobranie odpowiedniego URL: jest to punkt wyjścia dla każdego zadania skrapowania. URL jest odniesieniem do konkretnej strony internetowej, z której mają być pobrane dane.
2. Ładowanie zawartości HTML strony: po pobraniu URL cała zawartość HTML strony jest ładowana za pomocą Selenium.
3. Przetwarzanie pojedynczych URL lub iteracja po wielu podczęściach: w zależności od struktury zadania skrapowania, komponent asynchroniczny może przetwarzać pojedynczy URL lub iterować po wielu podczęściach (*subparts*). W tym ostatnim przypadku URL jest uzupełniany z każdą iteracją przez kolejną podczęść z listy.
4. Wybór elementów na podstawie powiązanych obiektów selektorów: dla każdego URL, elementy na stronie są wybierane na podstawie selektorów CSS z powiązanych obiektów selektorów.
5. Stosowanie obiektów modyfikatorów i filtrów: po wyborze elementów, stosowane są obiekty modyfikatorów i filtrów. Modyfikatory przekształcają surowe dane zeskrapowane ze strony. Krok ten jest opisany dogłębniej w następnym podrozdziale.
6. Zapisywanie wyników i opcjonalnie powiadamianie użytkownika: na końcu, wyniki są zapisywane jako obiekty encji `WorkerResult`. Każdy z tych obiektów reprezentuje pojedynczy wynik skrapowania i jest przechowywany w systemie dla późniejszego użycia. Ostatecznie, jeśli w trakcie skrapowania dane przekształcone przez modyfikatory spełniły warunek opisywany przy pomocy filtra i argumentu to użytkownik zostaje powiadomiony w wybrany przez niego sposób.

Listing 7. przedstawia część skryptu odpowiedzialnego za skrapowanie po stronie serwera. Uruchamiany jest sterownik Selenium, który następnie wczytuje dane ze strony i przetwarza je przy pomocy BeautifulSoup. Lista `selected_elements` zawiera wszystkie elementy, na które wskazuje selektor CSS zawarty w obiekcie `Selector`. W kodzie widoczny jest też sposób, w jaki podmieniane są podczęści URL przy użyciu wyrażenia regularnego.

Listing 7. Skrypt Python odpowiedzialny za skrapowanie po stronie serwerowej. Źródło: opracowanie własne.

```
for subpart in worker.subparts:
    replacement = subpart
    pattern = r"\{.*?\}"
    modified_url = re.sub(pattern, replacement,  site_url)
    driver.get(modified_url)
    html = driver.page_source
    soup = BeautifulSoup(html, 'html.parser')

    for selector in selectors:
        selected_elements = soup.select(selector.selector_text)
```

Komponent asynchroniczny, poprzez wykorzystanie Celery i Celery Beat w połączeniu z Selenium, oferuje możliwość wykonania skomplikowanych zadań skrapowania w efektywny i łatwy do zarządzania sposób. Dzięki temu zadania te mogą być realizowane w tle, co minimalizuje zakłócenia głównych operacji systemu i zapewnia płynność działania całego projektu.

5.3.4. Przetwarzanie danych przez modyfikatory

Gdy dane zostaną zebrane ze strony internetowej, należy poddać je procesowi modyfikacji za pomocą wybranych modyfikatorów. W kontekście tego konkretnego projektu modyfikatory domyślne pełnią funkcję metod, które są wykonywane w określonej sekwencji na każdym ze zeskrapowanych danych wskazanych przez selektor.

Podczas tego procesu, kluczową rolę odgrywa lista modyfikatorów, którą użytkownik ma możliwość zdefiniowania według swoich indywidualnych potrzeb. Każda zeskrapowana ze strony dana jest następnie przekazywana przez serię modyfikatorów, które są wykonywane jeden po drugim, zgodnie z ustaloną przez użytkownika kolejnością. To oznacza, że modyfikator, który jest pierwszy na liście, zacznie przetwarzać dane jako pierwszy, a następnie przekaże wynik do kolejnego modyfikatora na liście.

Jednakże, istnieje także możliwość zastosowania niestandardowych modyfikatorów, które są zdefiniowane przez użytkownika i zawierają atrybut URL. W takim przypadku proces przetwarzania nie odbywa się lokalnie. Zamiast tego, zeskrapowane dane (lub dane będące wynikiem przekształcenia poprzedniego modyfikatora) są wysyłane do serwera, na którym znajduje się niestandardowy modyfikator. Dane te są następnie przetwarzane na tym serwerze, a wyniki są zwracane z powrotem do głównego systemu.

Ten mechanizm umożliwia zastosowanie bardziej zaawansowanych lub specjalistycznych metod przetwarzania, które nie są dostępne w domyślnym zestawie modyfikatorów. Wprowadza to dodatkową warstwę elastyczności i dostosowywania się do potrzeb użytkownika, co może być przydatne w bardziej skomplikowanych scenariuszach skrapingu.

Proces ten jest kontynuowany aż do momentu, gdy wszystkie modyfikatory na liście zostaną zastosowane. Ostatecznie, dane, które zostały przetworzone przez ostatni modyfikator na liście, są uznawane za końcowy wynik tego etapu przetwarzania. W tym kontekście kolejność modyfikatorów na liście jest kluczowym aspektem procesu, ponieważ decyduje o sekwencji, w której dane są modyfikowane.

Listing 8. prezentuje w jaki sposób kolejne modyfikatory wprowadzają modyfikacje na pobranych elementach.

Listing 8. Funkcja w Python odpowiedzialna za wywołanie wszystkich modyfikatorów na skrapowanych danych w ramach skrapowania cyklicznego. Źródło: opracowanie własne.

```
def iterate_over_modifiers(modifiers, scrapped_data):
   if(modifiers.count() == 0):
        return scrapped_elements
    mods = modifiers.all()
    for i in range(modifiers.count()):
        for j in range(len(scrapped_data)):
            try:
                scrapped_data[j]=mods[i].modify(scrapped_data[j])
            except:
                scrapped_data[j]=None
    return scrapped_data
```

5.4. Baza danych

Do interakcji z bazą danych używany jest Django ORM (*Object-Relational Mapping*). Pozwala programistom na interakcję z bazą danych, tak jak robiliby to przy pomocy SQL. Innymi słowy, to sposób na tworzenie, pobieranie, aktualizowanie i usuwanie rekordów w bazie danych z poziomu skryptów Python. Niemniej jednak złożone zapytania i operacje mogą nadal wymagać surowego SQL-a do celów optymalizacji. Django wspiera różne systemy zarządzania bazami danych, takie jak PostgreSQL, MySQL, SQLite i inne, a konfigurację bazy danych można łatwo dostosować za pomocą ustawień Django. W ramach tego projektu skorzystano z domyślnego systemu bazy danych Django, czyli SQLite.

Definicja modeli w Django jest realizowana za pomocą języka Python. Każdy model jest reprezentowany jako klasa dziedzicząca po `django.db.models.Model`, a każde pole modelu po stronie bazy danych jest reprezentowane jako instancja klasy `Field` (np. `CharField` dla krótkich tekstów, `IntegerField` dla liczb całkowitych itd.). Każdy model mapuje na pojedynczą tabelę bazy danych, co ułatwia ich zarządzanie i manipulację.

W listingu 9. przedstawiono definicję modelu `Sitemap`, który reprezentuje mapę strony należącą do użytkownika. Model ten posiada pola takie jak `owner`, `address` i `name`, które zostaną odzwierciedlone jako kolumny w tabeli bazy danych.

Listing 9. Przykładowa definicja modelu w Django. Źródło: opracowanie własne.

```
    class Sitemap(models.Model):
        owner     =     models.ForeignKey(settings.AUTH_USER_MODEL,
        on_delete = models.CASCADE )
        address = models.URLField( max_length=600 )
        name = models.CharField( max_length=600 )
        def _str_(self):
            return self.name
```

Migracje są kluczowym elementem zarządzania zmianami modeli. Po wprowadzeniu zmian można utworzyć migrację za pomocą polecenia `makemigrations`. Django wygeneruje skrypt Pythona reprezentujący te zmiany. Na przykład, po dodaniu nowego modelu do aplikacji, Django stworzy nowy plik migracji zawierający instrukcje tworzenia nowej tabeli w bazie danych. Przykładowy kod wygenerowany przez migracje Django przedstawiony jest w listingu 10. Zaprezentowana migracja

zmienia typ pola `frequency` w modelu `Worker` na `IntegerField`. Migracje pozwalają na elastyczne dostosowywanie struktury bazy danych w miarę potrzeb, zapewniając integralność danych i zachowując kompatybilność z aplikacją.

Listing 10. Przykładowy kod migracji w Django. Źródło: opracowanie własne.

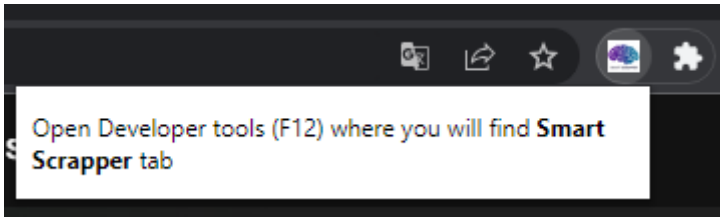
```
class Migration(migrations.Migration):

    dependencies=
[('scrapper_api', 0013_alter_worker_notifier'),]
    operations = [
        migrations.AlterField(
            model_name='worker',
            name='frequency',
            field=models.IntegerField(),
        ),]
```

Po wygenerowaniu migracji można ją zastosować na bazie danych za pomocą polecenia `migrate`. To polecenie przegląda wszystkie migracje, które nie zostały jeszcze zastosowane, i wykonuje je w odpowiedniej kolejności, gwarantując, że schemat bazy danych jest zgodny z aktualnym stanem modeli.

6. Zastosowanie prototypu

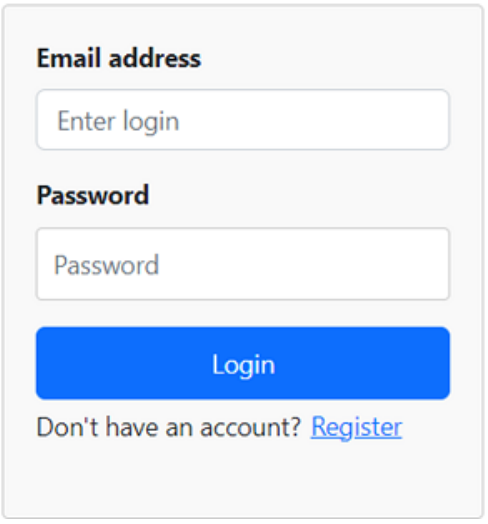
Rozwiązanie prezentuje unikalne i przyjazne dla użytkownika podejście do pozyskiwania danych ze stron internetowych, zapewniając płynną integrację z przeglądarkami opartymi na Chromium. Wykorzystuje panel devtools, który otwiera się po naciśnięciu klawisza F12, aby umożliwić użytkownikom definiowanie, zarządzanie i wykonywanie zadań pozyskiwania danych ze stron internetowych bez opuszczania środowiska przeglądarki.

Po zainstalowaniu rozszerzenia naturalnym, pierwszym odruchem użytkownika jest kliknięcie ikonki rozszerzenia widocznej w prawym górnym rogu przeglądarki, w grupie rozszerzeń. W przypadku tego prototypu użytkownik dostanie jedynie informację, że aby skorzystać z funkcjonalności systemu musi przejść do panelu narzędzi deweloperskich. Komunikat widoczny po kliknięciu ikonki rozszerzenia widoczny jest na rysunku 12.



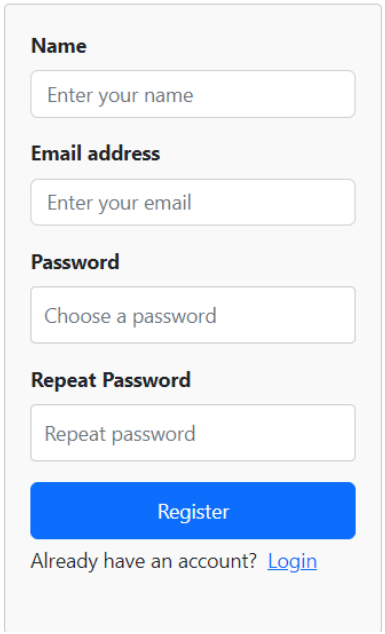
Rysunek 12. Popup rozszerzenia w przeglądarce Chrome. Źródło: [7]

Po otwarciu panelu narzędzi deweloperskich (np. klawiszem *F12*), obok zwykłych zakładek *„Elements”*, *„Console”*, *„Sources”* itp. pojawia się nowa zakładka o nazwie *„Smart Scraper”*. Po przejściu do tej zakładki użytkownicy są witani stroną logowania w celu uwierzytelnienia się. Formularz logowania ma prosty wygląd, jak widać na rysunku 13.



Rysunek 13. Formularz logowania. Źródło: opracowanie własne.

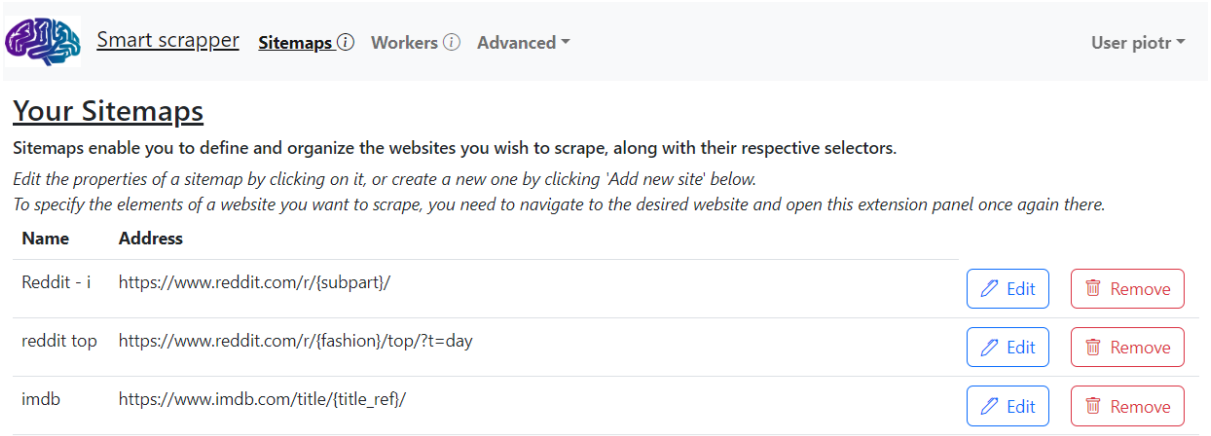
Nowi użytkownicy mogą zarejestrować konto, klikając przycisk *„Register”*, który przeniesie ich do widoku rejestracji. Widok rejestracji, zilustrowany na rys. 14, wymaga od użytkowników podania swoich danych w celu utworzenia nowego konta.



Rysunek 14. Formularz rejestracji użytkownika. Źródło: opracowanie własne.

6.1. Widok główny

Po zalogowaniu zaprezentowany jest widok główny widoczny na rysunku 15. Ten początkowy widok służy jako centrum, z którego użytkownicy mogą przechodzić do każdego pod-widoku przy pomocy paska nawigacyjnego na górze interfejsu. Został zaprojektowany tak, aby był prosty i intuicyjny, umożliwiając użytkownikom przegląd ich map witryn od razu po otworzeniu GUI oraz umożliwiając im szybkie przechodzenie do innych widoków.



Rysunek 15. Główny widok rozszerzenia. Źródło: opracowanie własne.

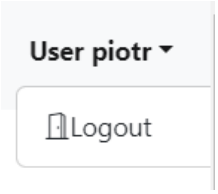
W górnej części interfejsu graficznego użytkownik znajdzie pasek nawigacyjny z ikoną i nazwą rozszerzenia oraz z odnośnikami do innych widoków. W narzędziu wyróżnić można cztery główne widoki, są to:

- *Sitemaps* – widok z listą map witryn,
- *Workers* – widok z listą zdefiniowanych przez użytkownika procesów cyklicznych (robotników),
- *Notifiers* – widok z listą zdefiniowanych przez użytkownika, niestandardowych punktów końcowych HTTP, które po dodaniu do robotnika, wywoływane są, gdy zostaną w trakcie jego

skanowania spełnione zdefiniowane przez użytkownika warunki. Widok dostępny jest po wybraniu odnośnika *„Custom Notifiers”* widocznego po kliknięciu i rozwinięciu listy pod odnośnikiem na pasku głównym *„Advanced”*,

- *Modifiers* – widok z listą zdefiniowanych przez użytkownika, niestandardowych punktów końcowych HTTP, które wywoływane są, gdy zostaną wybrane w trakcie definiowania cyklicznych procesów skrapowania w sekcji selektorów. Widok dostępny jest po wybraniu *„Custom Modifiers”* widocznego po kliknięciu i rozwinięciu listy pod odnośnikiem na pasku głównym *„Advanced”*.

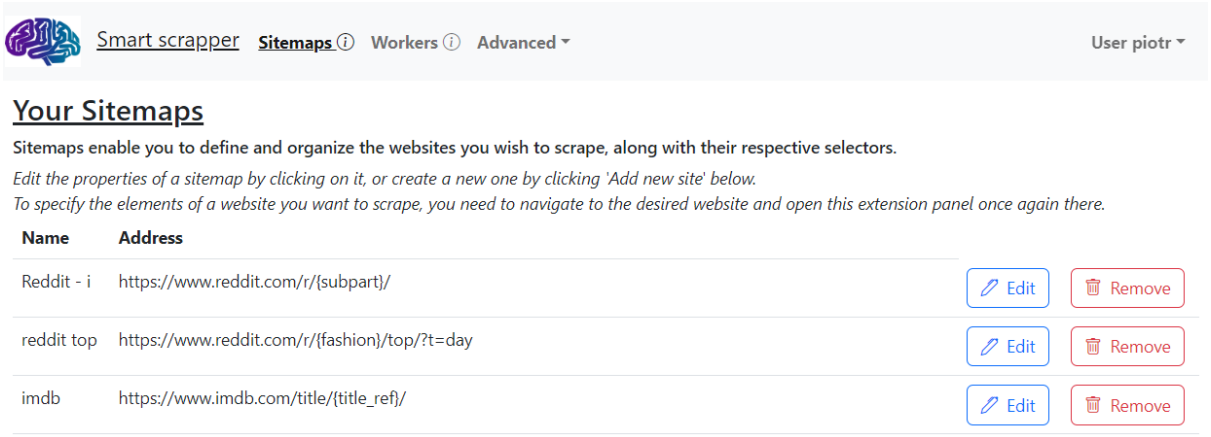
W prawej części paska nawigacyjnego znajduje się również przycisk z nazwą użytkownika umożliwiający wylogowanie się. Widoczny jest po naciśnięciu na nazwę użytkownika, tak jak widać to na rysunku 16.



Rysunek 16. Nazwa użytkownika na pasku nawigacyjnym i przycisk umożliwiający wylogowanie się. Źródło: opracowanie własne.

6.2. Lista map stron użytkownika

Po zalogowaniu pierwszym wczytanym widokiem jest widok zdefiniowanych map witryn (widoczny na rysunku 17). W tym kontekście reprezentuje instrukcję skrapowania stron internetowych, zawierającą nazwę ułatwiającą identyfikację, adres określający adres URL strony internetowej, która ma być skrapowana przez robotnika (z opcjonalnymi symbolami zastępczymi {subpart} dla dynamicznych adresów URL) oraz zestaw zdefiniowanych selektorów i ich kolekcji.



Rysunek 17. Widok listy map stron. Źródło: opracowanie własne.

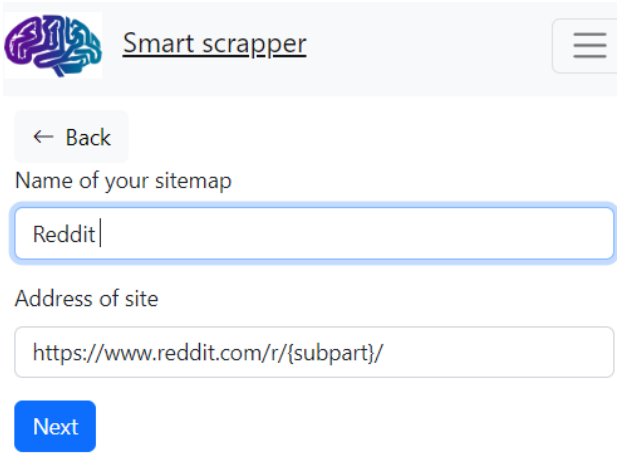
Lista map witryn przedstawia widok z lotu ptaka wszystkich skonfigurowanych przez użytkownika map witryn. Każda z nich jest odrębną jednostką, która zawiera określony zestaw instrukcji dotyczących skrapowania danych z określonej witryny internetowej. Na tej stronie użytkownik może przejść do poszczególnych map witryn, klikając ich wiersz. Spowoduje to wyświetlenie zdefiniowanych selektorów i kolekcji selektorów powiązanych z wybraną mapą oraz widoku pozwalającego na ich dalsze dodawanie (opisane w rozdziale 6.3).

Użytkownicy mogą również wykonywać czynności, takie jak usuwanie mapy witryny bezpośrednio z tej listy.

Aby dodać nową mapę witryny, użytkownik może kliknąć przycisk *„Add new sitemap”* widoczny pod główną listą.

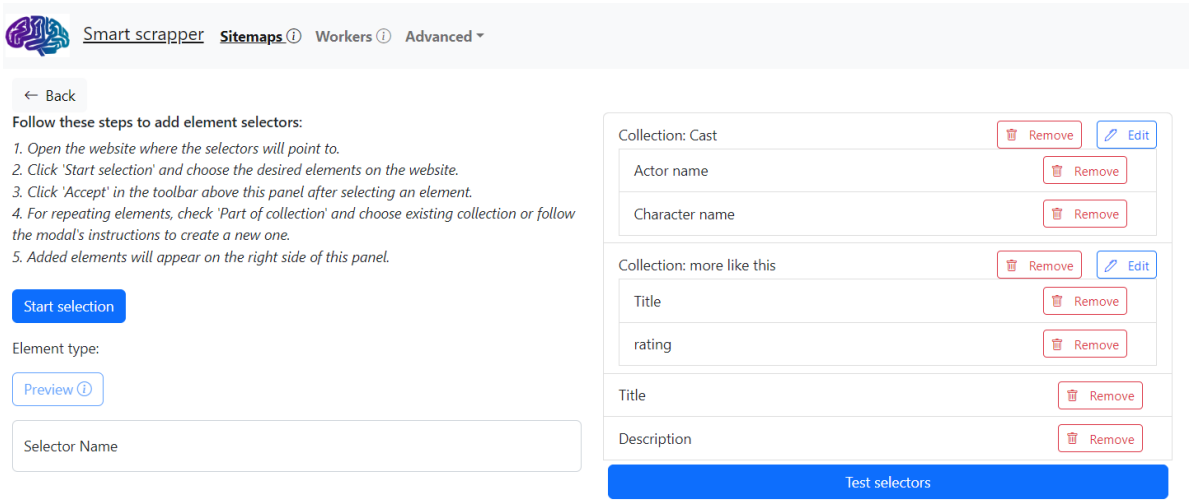
6.3. Tworzenie/edycja nowej mapy witryny

Po naciśnięciu przycisku *„Add new sitemap”* użytkownik zostanie przekierowany do widoku, w którym, aby kontynuować, musi wprowadzić podstawowe informacje o nowej mapie witryny, takie jak jej nazwa i adres. W razie potrzeby można również sparametryzować adres URL za pomocą symboli zastępczych (np. {subpart}), które można zastąpić określonymi wartościami podczas późniejszego definiowania robotnika. Formularz ten widoczny jest na rysunku 18.



Rysunek 18. Formularz podstawowych informacji mapy strony. Źródło: opracowanie własne.

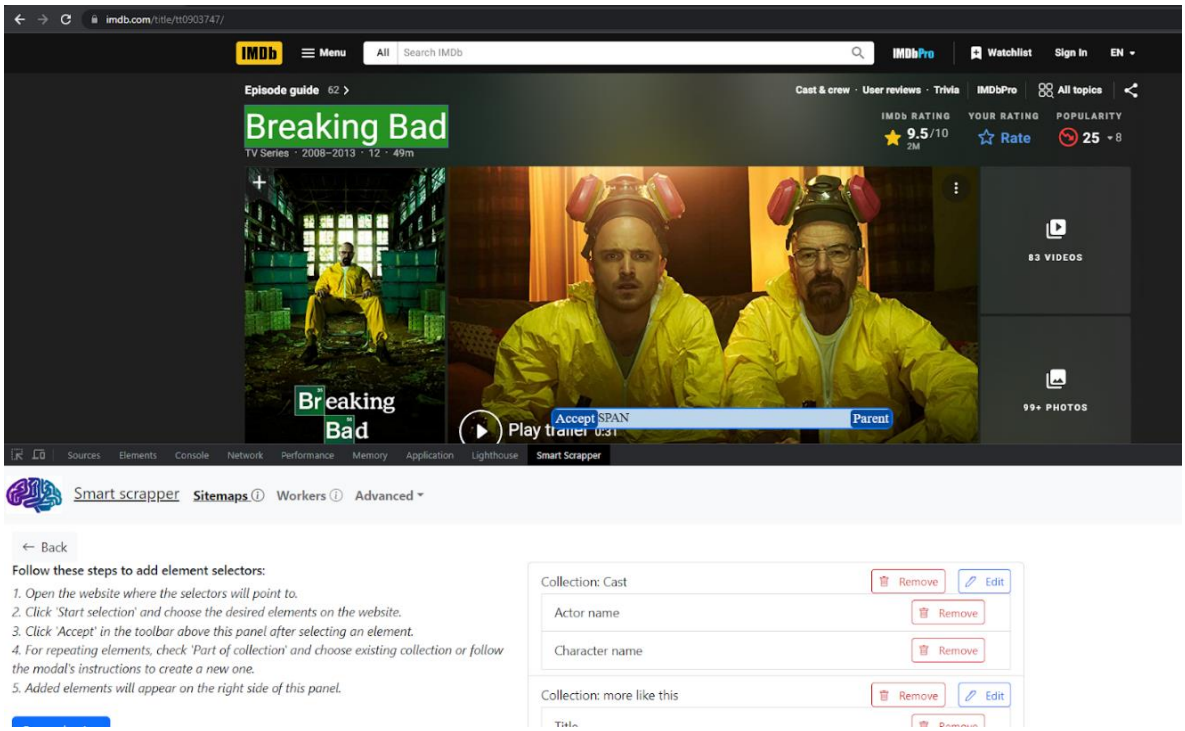
Następnie użytkownik zostanie przekierowany do widoku, w którym możliwe jest wybieranie elementów witryny, których selektory CSS mają zostać zapisane w mapie strony. Proces ten jest wysoce interaktywny, umożliwiając użytkownikom bezpośrednie wybieranie żądanych elementów ze strony internetowej widocznej nad panelem narzędzi deweloperskich. Następny widok (rysunek 19.) jest wyświetlany od razu, gdy z listy zostanie wybrana mapa strony do edycji.



Rysunek 19. Widok pozwalający na: dodawanie i podgląd nowych selektorów mapy strony (po lewej) i zarządzanie selektorami już zawartymi w mapie strony (po prawej). Źródło: opracowanie własne.

Dodawanie kolejnych selektorów do mapy witryny rozpoczyna się od kliknięcia przycisku *„Start Selection”*. Ta czynność włącza tryb selekcji, w którym użytkownicy mogą najeżdżać kursorem i wybierać różne elementy bezpośrednio na stronie internetowej widocznej nad panelem narzędzi deweloperskich.

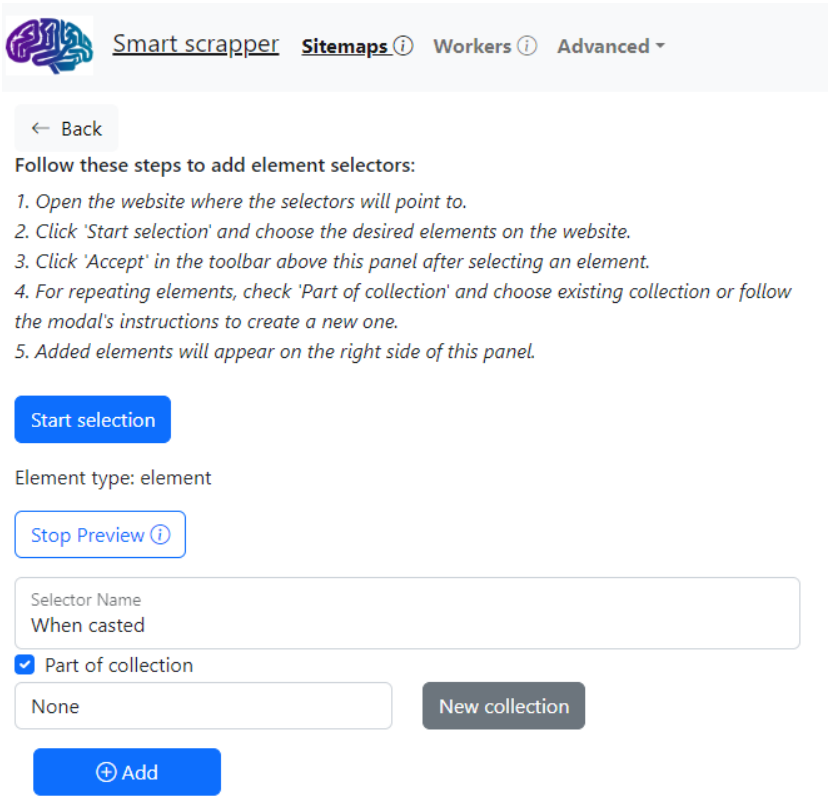
Po kliknięciu fragmentu witryny, który zawiera interesujące użytkownika dane, u dołu strony internetowej pojawia się okienko wyświetlające szczegóły wybranego komponentu HTML. Użytkownicy mogą następnie kliknąć *„Akceptuj”*, jeśli część witryny została wybrana prawidłowo. Określony w ten sposób element jest teraz gotowy do dodania do mapy witryny poprzez ponowne kliknięcie przycisku *„Add”* już w interfejsie graficznym poniżej. Na rysunku 20. przedstawiony jest etap wybierania elementu na witrynie. W tym przypadku użytkownik zaznaczył tytuł i może na pasku na dole strony wcisnąć przycisk *„Accept”*, aby dodać element do mapy strony, lub może nacisnąć *„Parent”*, aby zaznaczyć nadrzędny komponent, w którym zawiera się, w tym przypadku, element z tytułem.



Rysunek 20. Strona, na której użytkownik wybiera elementy do skrapowania z otwartym interfejsem graficzny rozszerzenia w panelu narzędzi deweloperskich. Źródło: opracowanie własne.

Przed dodaniem elementu do mapy witryny użytkownicy mają możliwość dodania tego elementu do kolekcji. Kolekcje są szczególnie przydatne do przechwytywania powtarzających się elementów na stronie internetowej, takich jak na przykład lista aktorów. Wskazywane są przez ogólny selektor CSS, taki jak np. …`>div:nth-child(1)>div:nth-child(4)>`***`div`***`>div:nth-child(4)>`… . To jest szczególnie przydatne, gdy użytkownik chce wybrać więcej niż jeden typ informacji z listy, na przykład nazwisko aktora i data jego urodzenia. Selektory te zapewniają strukturę rozgałęzień do nawigacji po różnych elementach kolekcji i są używane podczas dodawania nowych powtarzających się elementów.

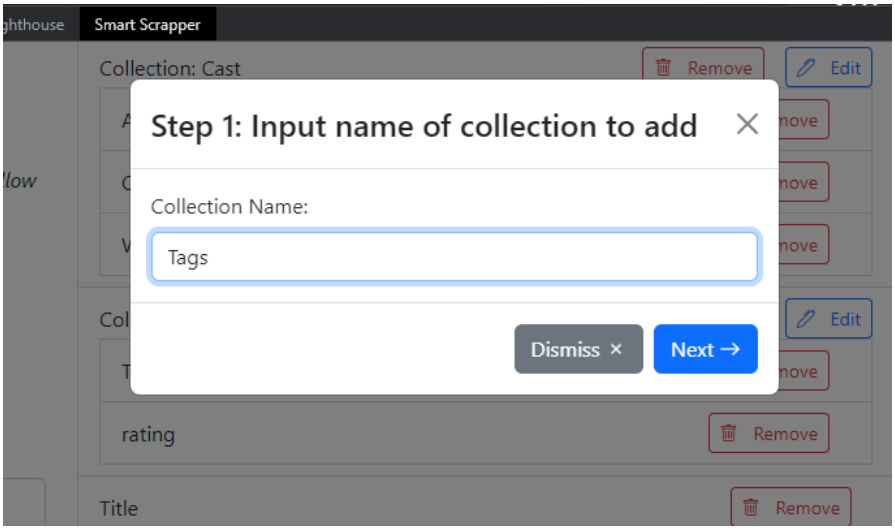
Jeśli istniejące kolekcje są już zdefiniowane, użytkownicy mogą wybrać jedną z nich z rozwijanej listy (na rysunku 21. - obok przycisku „*New collection”*).



Rysunek 21. Widok graficznego interfejsu służącego do dodawania selektorów i kolekcji do mapy strony. Źródło: opracowanie własne.

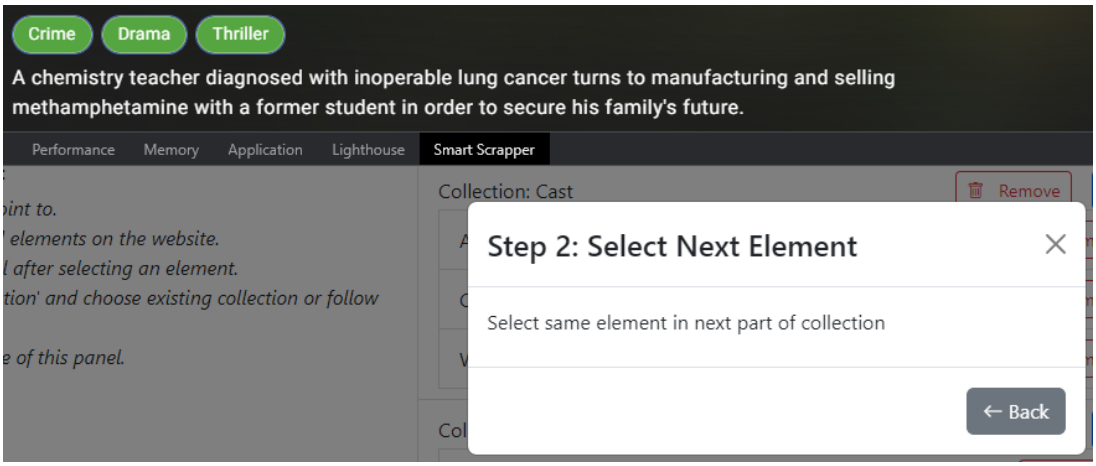
Jeśli natomiast żadna kolekcja nie jest dostępna, pojawi się okno modalne, umożliwiające użytkownikom zdefiniowanie nowej kolekcji. W przypadku gdy wybrany element nie odpowiada żadnej z już zdefiniowanych kolekcji, użytkownik sam może zdecydować, czy chce utworzyć nową kolekcję klikając przycisk *„New collection”*.

Proces definiowania kolekcji dostępny jest z poziomu okna modalnego przykrywającego aktualny interfejs rozszerzenia. Pierwszym etapem jest wprowadzenie nazwy nowej kolekcji. Interfejs dla tego etapu widoczny jest na rysunku 22.



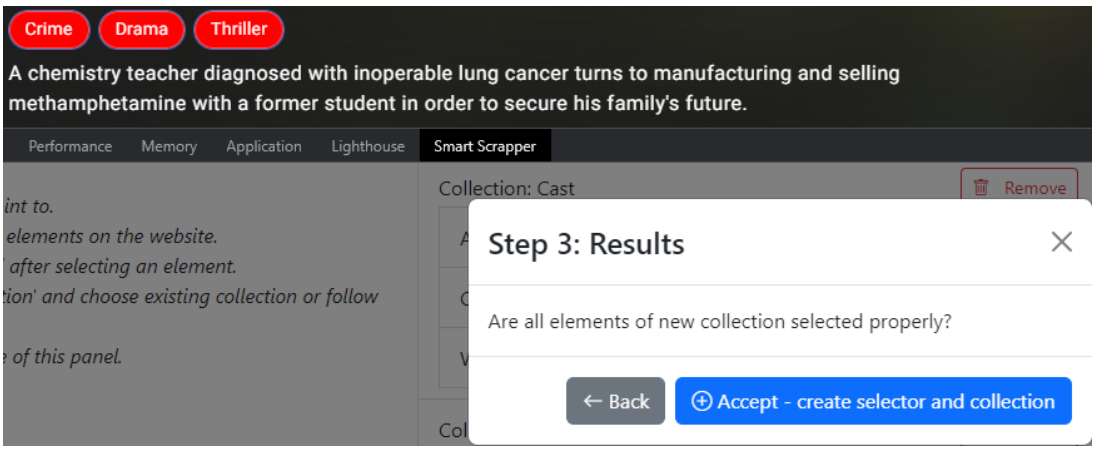
Rysunek 22. Proces dodawania nowej kolekcji selektorów – krok 1. Źródło: opracowanie własne.

Kolejnym etapem jest wybór następnego elementu z kolekcji, na którą ma wskazywać aktualnie tworzony selektor. Krok ten zobrazowany jest na rysunku 23. Aby zdefiniować kolekcję, użytkownik musi wybrać inny element, który należy do tej samej grupy (elementy te zostaną wyróżnione na podstawie ogólnego selektora CSS bez *„nth-of-type”* w celu ułatwienia wyboru użytkownikowi). Mimo tego, że uogólniony selektor jest często wystarczający, żeby poprawnie podświetlić wszystkie składowe kolekcji, które chce wybrać użytkownik to i tak konieczne jest manualne wybranie kolejnego elementu. W przeciwnym razie zdarza się czasami, że uogólniony selektor jest zbyt uniwersalny i wskazuje też elementy, których nie powinien.



Rysunek 23. Proces dodawania nowej kolekcji selektorów – krok 2. Źródło: opracowanie własne.

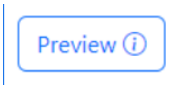
Po wyborze przez użytkownika kolejnego elementu, w ostatnim etapie definiowania kolekcji użytkownik jest proszony o sprawdzenie, czy zbiór jest podświetlony poprawnie (rys. 24). Jeśli wybór nie jest satysfakcjonujący i kolekcja nie jest podświetlona prawidłowo, użytkownik może kliknąć *"Back"*, aby powrócić do etapu wyboru. Jeśli natomiast składowe zostały zaznaczone prawidłowo, kliknięcie przycisku *"Accept"* spowoduje zamknięcie okna modalnego i zdefiniowana kolekcja zostanie dodana do listy istniejących w ramach mapy witryn kolekcji.



Rysunek 24. Proces dodawania nowej kolekcji selektorów – krok 3. Źródło: opracowanie własne.

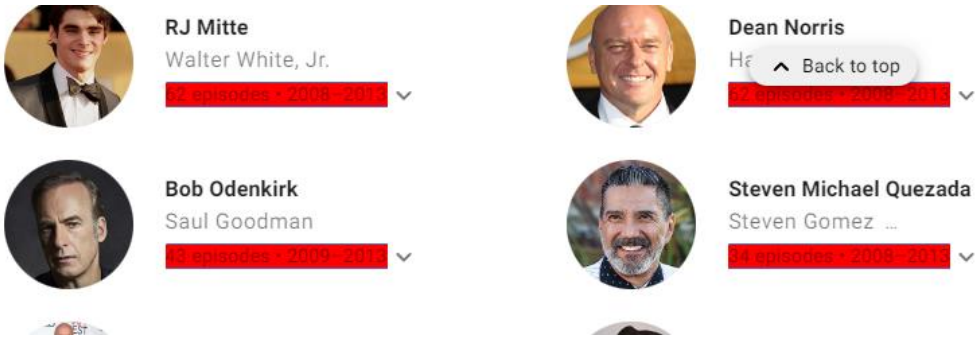
To praktyczne, interaktywne podejście do definiowania selektorów i kolekcji umożliwia użytkownikom dokładne określenie danych, które mają być skrapowane.

Dodatkowo po wybraniu selektora użytkownik może sprawdzić, czy jest on poprawny. W tym celu może użyć przycisku *„Preview”* (rysunek 25.) w głównym interfejsie definiowania selektorów mapy strony.



Rysunek 25. Przycisk *Preview* pozwalający na podświetlanie wybranych elementów selektora/kolekcji na stronie. Źródło: opracowanie własne.

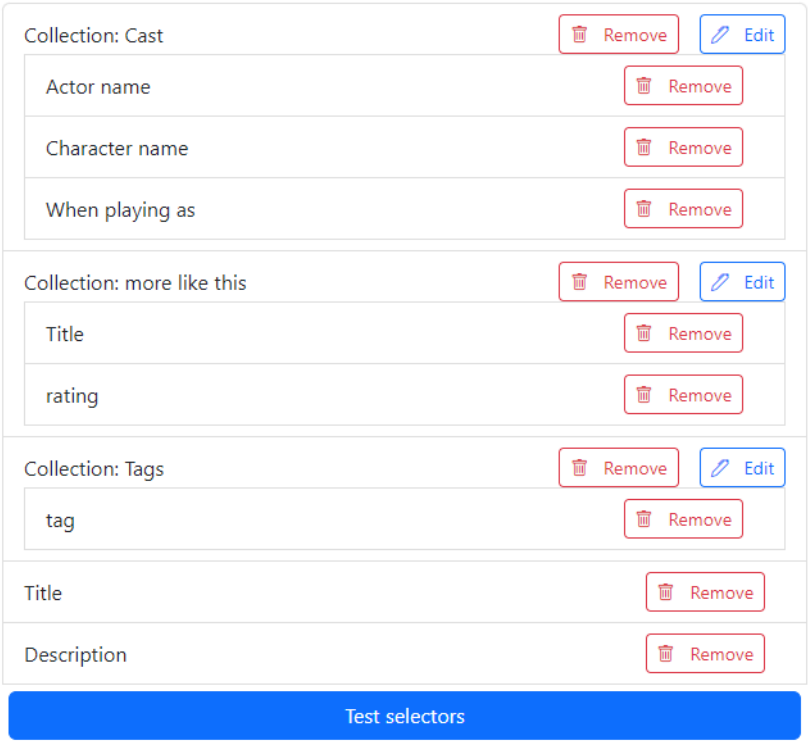
Po naciśnięciu tego przycisku na otwartej powyżej rozszerzenia stronie zostaną podświetlone elementy, na które wskazuje aktualnie definiowany selektor albo kolekcja selektorów tak jak jest to widoczne na rysunku 26.



Rysunek 26. Sposób podświetlania elementów kolekcji na stronie po naciśnięciu *Preview*. Źródło: opracowanie własne.

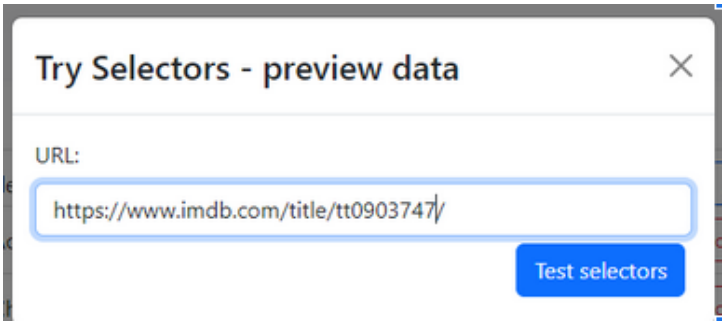
Po upewnieniu się, że selektor jest zdefiniowany poprawnie, użytkownik może kliknąć przycisk *„Add”* w głównym interfejsie do celu dodania selektora do mapy strony.

Zawierające się w mapie strony selektory widoczne są w prawej części widoku, tak jak jest to zaprezentowane na rysunku 27. W przedstawionej mapie witryny dodane są trzy kolekcje: *„Cast”*, *„more like this”* oraz *„Tags”*, a także tytuł i opis serialu.



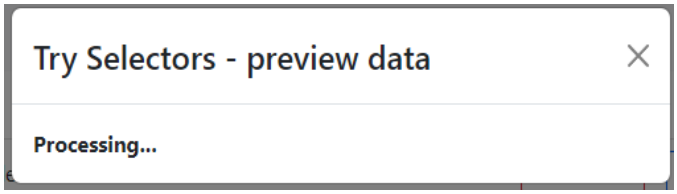
Rysunek 27. Wizualizacja selektorów CSS i ich kolekcji w mapie strony. Źródło: opracowanie własne.

Użytkownik z poziomu tej listy może usuwać poszczególne elementy i edytować nazwy kolekcji. Ponadto użytkownik może przetestować wszystkie selektory, klikając przycisk *„Test selectors”*. Po kliknięciu tego przycisku użytkownik zostanie poproszony o wprowadzenie URL, na którym mają zostać przetestowane selektory. Okno modalne do wprowadzenia URL widoczne jest na rysunku 28.



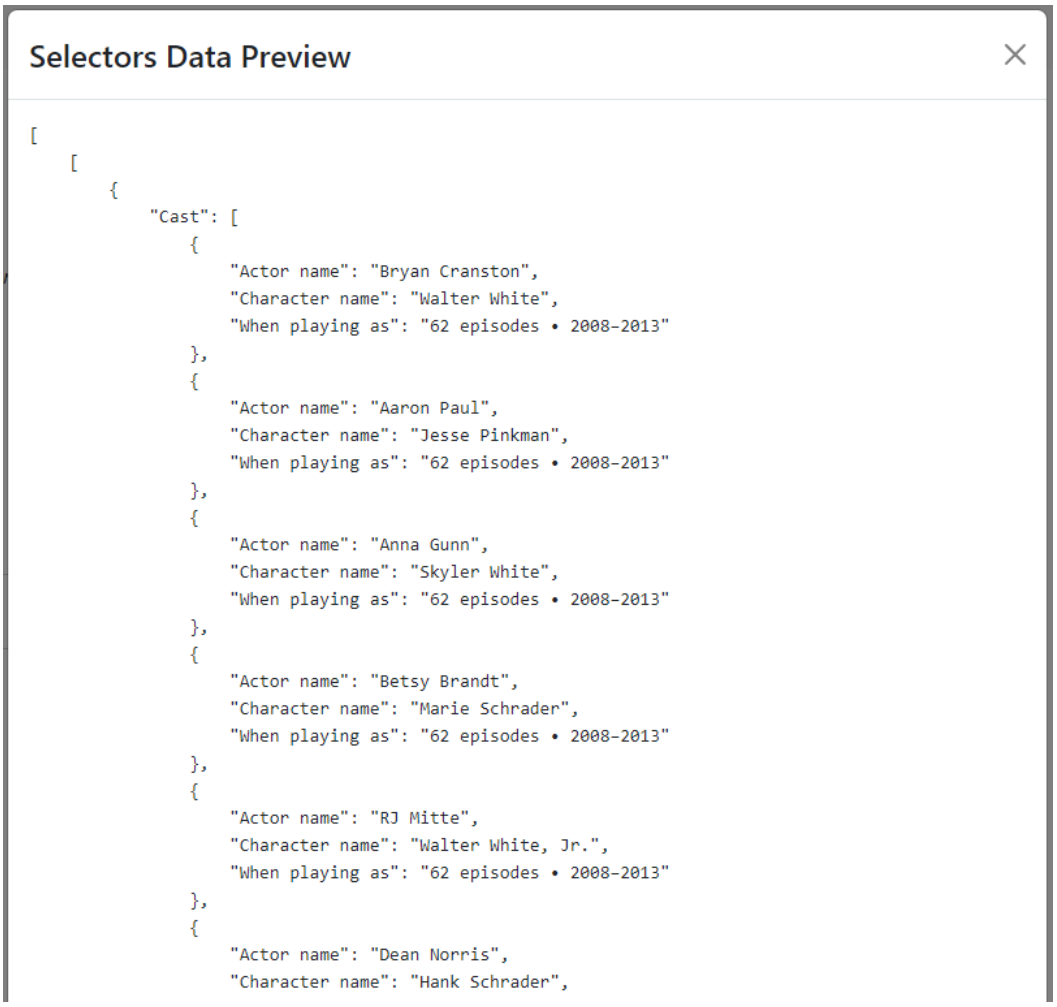
Rysunek 28. Okno modalne do sprawdzania selektorów z wprowadzonym linkiem do sprawdzenia. Źródło: opracowanie własne.

Po wprowadzeniu URL strony do zeskrapowania i wciśnięciu przycisku *„Test selectors”* zostanie wysłane zlecenie do serwera, a użytkownikowi zostanie pokazana informacja, że musi zaczekać na wynik (widoczne na rys. 29).



Rysunek 29. Okno modalne do sprawdzenia selektorów w mapie strony w trakcie oczekiwania na wynik z serwera. Źródło: opracowanie własne.

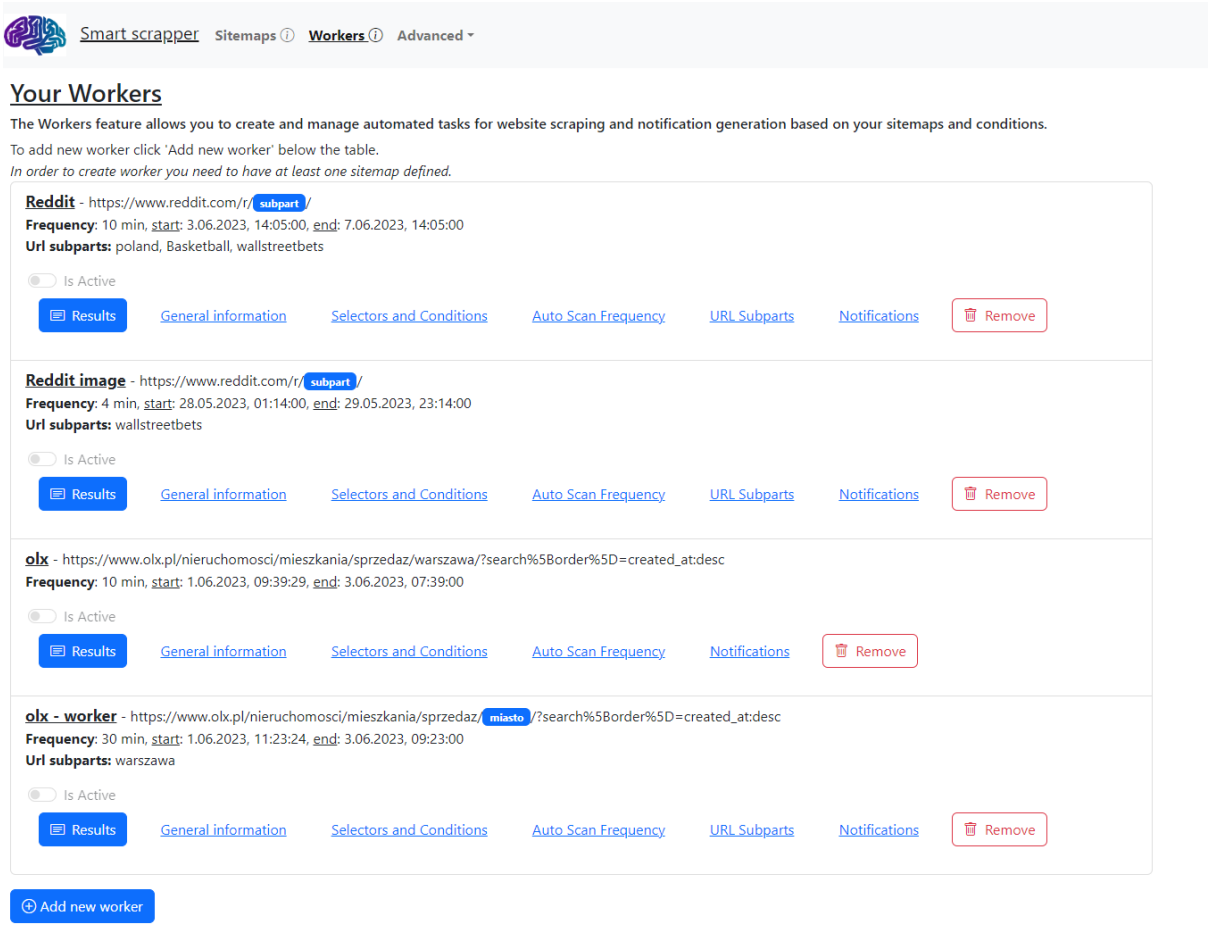
W trakcie oczekiwania usługa serwerowa wczyta stronę, pobierze z niej dane, na które wskazują selektory i wynik tego działania zostanie wyświetlony w interfejsie użytkownika, tak jak jest to widoczne na rysunku 30.



Rysunek 30. Okno modalne z podglądem zeskrapowanych danych na podstawie mapy strony. Źródło: opracowanie własne.

6.4. Lista robotników użytkownika

Następnym istotnym widokiem w aplikacji jest widok robotników (zadań cyklicznych) nazywanych w systemie jako *„Worker”*. Widok listy ze wszystkimi zdefiniowanymi pracownikami widoczny jest po naciśnięciu odnośnika *„Workers”* na głównym pasku nawigacyjnym. Lista dostępna jest po przejściu na podstronę i zapewnia podgląd wszystkich robotników użytkownika (rysunek 31.).



Rysunek 31. Lista robotników użytkownika. Źródło: opracowanie własne.

Dla każdego robotnika pokazana jest jego nazwa, powiązana mapa witryny, zamienne części URL, częstotliwość działania i okno czasowe, w którym ma pozostać aktywny.

Każdy robotnik na liście ma też kilka przycisków/odnośników, otwierających różne widoki do zarządzania różnymi jego właściwościami:

- Kliknięcie przycisku *„Results”* przenosi użytkownika do widoku wyników. W tym miejscu użytkownik może sprawdzić dane zebrane przez robotnika podczas jego operacji.
- Informacje ogólne (*„General information”*), Selektory i warunki (*„Selectors and Conditions”*), Częstotliwość automatycznego skanowania („*Auto Scan Frequency*”), Podczęści URL (*„URL Subparts”*), Powiadomienia (*„Notifications”*) – odnośniki te przenoszą użytkownika do odpowiednich widoków, które opisano w następnej sekcji dotyczącej dodawania nowego pracownika.
- Użytkownik może też wyłączyć i włączyć automatyczne skanowanie dwustanowym przyciskiem *„Is Active”*. Jest on dostępny, gdy aktualny czas spełnia warunki w ustawieniach *„Auto scan frequency”*.

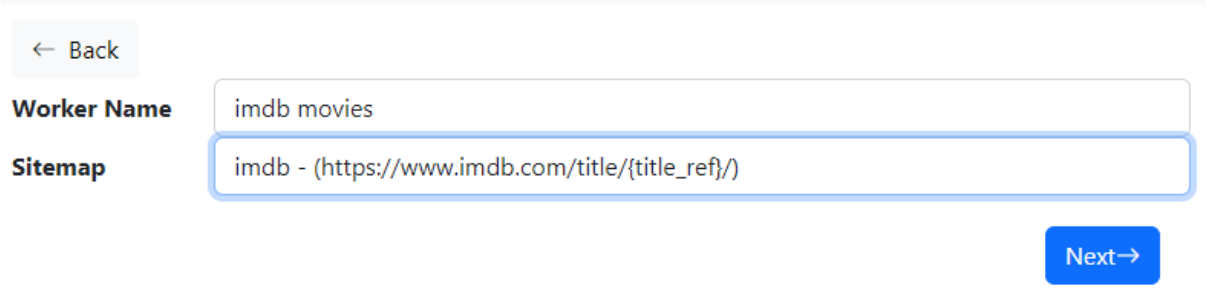
- Użytkownik może też usunąć robotnika wraz ze wszystkimi zebranymi przez niego danymi przy użyciu przycisku *„Remove”*.

6.5. Dodawanie i edycja robotnika

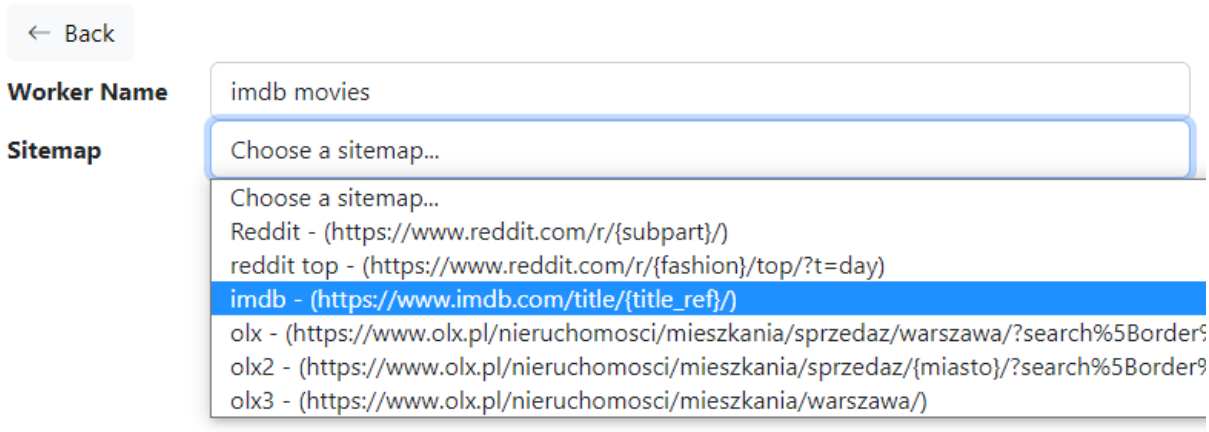
Aby rozpocząć nowe zadanie cykliczne skrapowania stron internetowych, użytkownik musi utworzyć nowego robotnika. Proces ten rozpoczyna się od kliknięcia przycisku *„+ Add new worker”* znajdującego się pod listą istniejących robotników. Następujący po tym przepływ pracy jest procedurą krok po kroku, prowadzącą użytkowników przez serię widoków opisanych w następnych podrozdziałach.

6.5.1. Informacje ogólne

W pierwszym kroku użytkownik proszony jest o wybranie mapy witryny z listy utworzonych przez niego map witryn i podanie unikalnej nazwy dla nowego robotnika (rysunek 32. i 33.).



Rysunek 32. Formularz dodawania robotnika – podstawowe dane. Źródło: opracowanie własne.

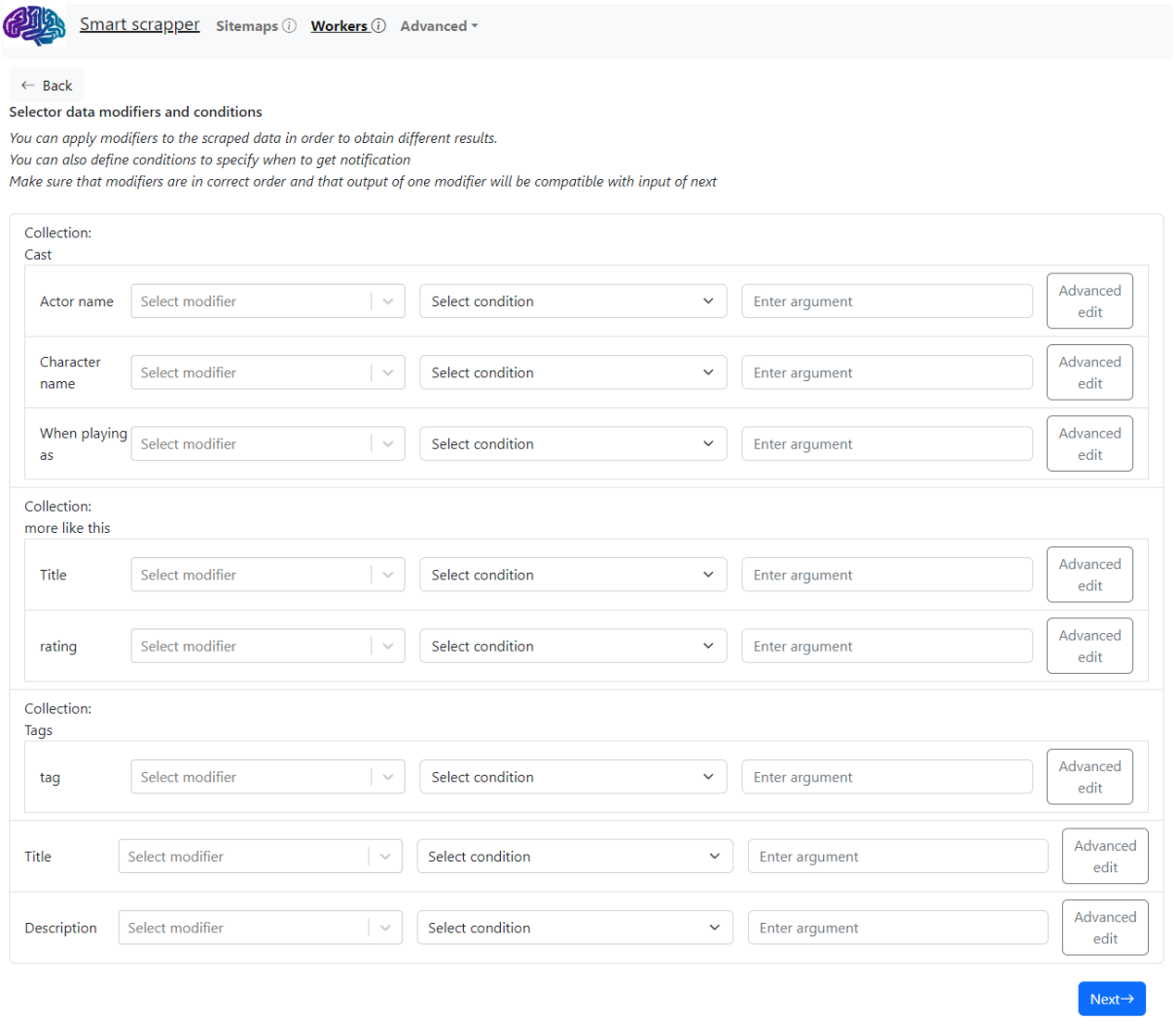


Rysunek 33. Formularz dodawania robotnika – wybór mapy strony. Źródło: opracowanie własne.

Wprowadzenie tych informacji jest konieczne, żeby przejść do następnego kroku tworzenia robotnika.

6.5.2. Selektory, modyfikatory i warunki

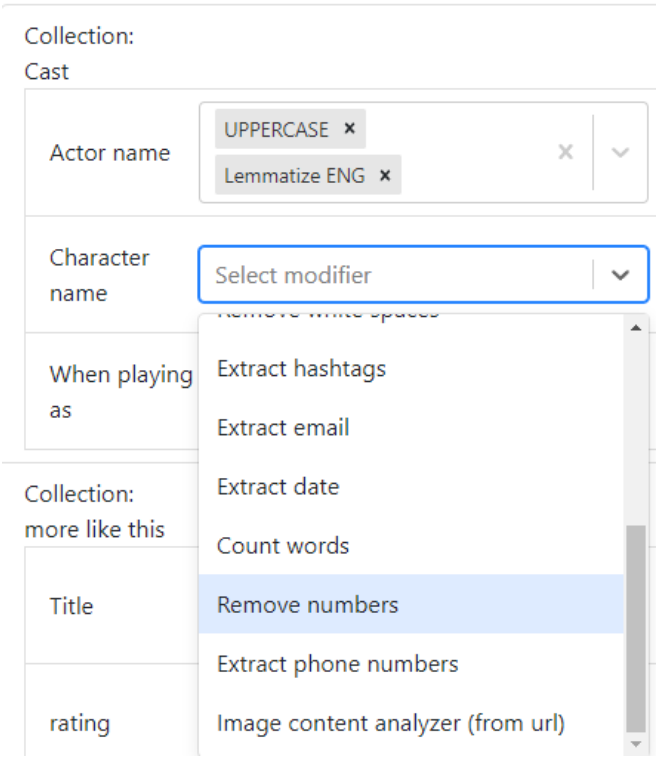
Następny widok umożliwia użytkownikowi dostosowanie modyfikatorów i filtrów dla każdego selektora w wybranej mapy witryny. Interfejs dla tego etapu widoczny jest na rysunku 34.



Rysunek 34. Formularz dodawania robotnika – widok ustawień modyfikatorów i filtrów. Źródło: opracowanie własne.

Użytkownik może modyfikować ścieżki selektorów CSS w tych obiektach selektorów (przy pomocy przycisku *„Advanced edit”*), ustawiać wymagane modyfikatory i definiować warunki, które muszą zostać spełnione, aby otrzymać powiadomienie. Możliwość dostosowania tych parametrów zapewnia, że dane zbierane przez pracownika są zgodne z potrzebami użytkownika.

Kolejność w wyborze modyfikatorów ma znaczenie, ponieważ dane są modyfikowane i przekazywane od jednego modyfikatora do kolejnego z listy. Wybór modyfikatorów w GUI jest widoczny na rysunku 35.



Rysunek 35. Formularz dodawania robotnika – wybieranie modyfikatorów. Źródło: opracowanie własne.

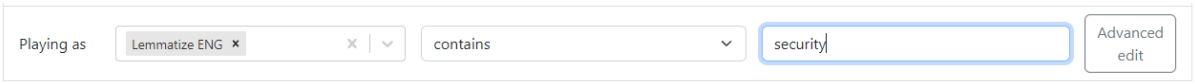
Lista modyfikatorów domyślnie zawiera takie operacje jak:

- Uppercase,
- Lowercase,
- Get numbers,
- Lemmatize PL,
- Lemmatize ENG,
- Remove interpunctions,
- Remove white spaces,
- Extract hashtags,
- Extract email,
- Extract date,
- Count words,
- Remove numbers,
- Extract phone number.

Modyfikatory te opisane są w podrozdziale 3.5 - *„Modyfikatory danych i filtry”*. Przy definiowaniu metod dla *„Lemmatize PL/ENG”* użyto modeli lematyzacji opisane w podrozdziale 4.13. - *„Modele NLP z biblioteki spaCy”*.

Dodatkowo lista ta może zawierać zdefiniowane przez użytkownika niestandardowe modyfikatory. W tym przypadku jest to *Image content analyzer*. Ten punkt końcowy wykorzystany jest jako przykład - akceptuje URL zdjęcia i zwraca wykryte na nim obiekty. Wykorzystany w nim model to *hustlv/yolo-tiny*, opisany w rozdziale o wybranych technologiach (4.14).

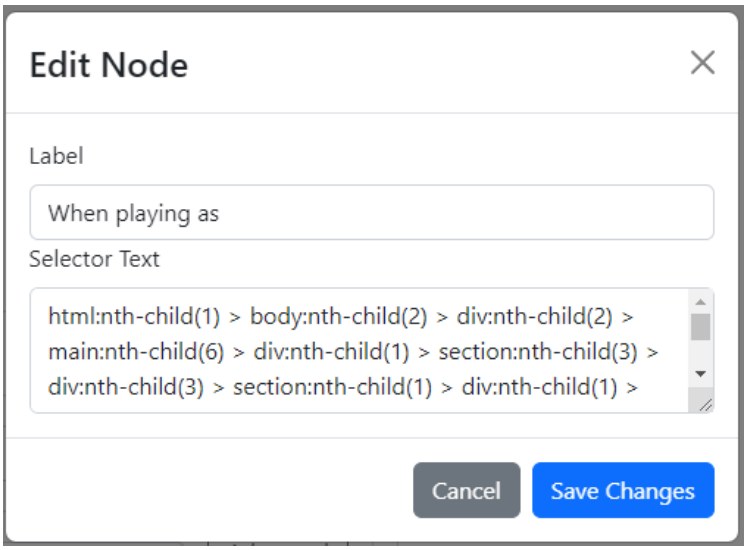
Następnie, użytkownik może wprowadzić warunek, który musi zostać spełniony, aby otrzymać powiadomienie. Przykładowy warunek jest widoczny na rysunku 36.



Rysunek 36. Formularz dodawania robotnika – ustawianie warunków powiadomień. Źródło: opracowanie własne.

W tym przypadku użytkownik dostanie powiadomienie, jeśli dane wyekstrahowane z pola *„Playing as”* i zmodyfikowane przy użyciu modyfikatora *„Lemmatize ENG”* zawierać będą słowo *„security”*.

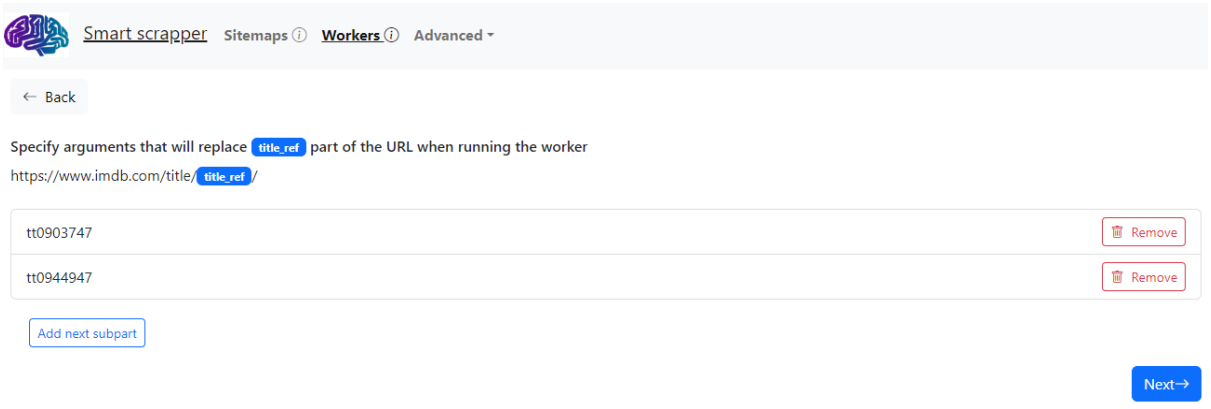
Użytkownik może też edytować ścieżki selektorów po naciśnięciu przycisku *„Advanced edit”*. Okno modalne do tej edycji jest zaprezentowane na rysunku 37.



Rysunek 37. Zaawansowany widok umożliwiający zmianę selektorów CSS w trakcie definiowania robotnika. Źródło: opracowanie własne.

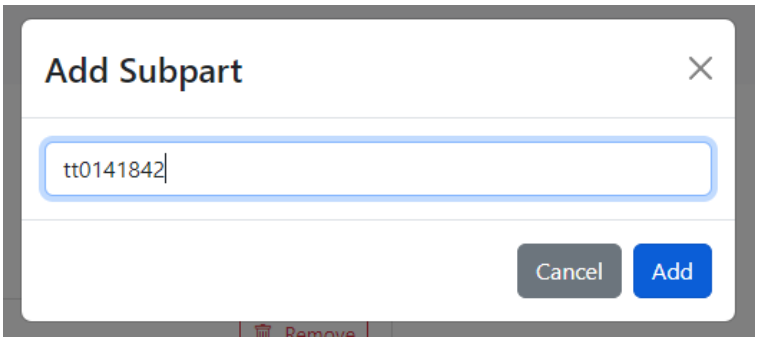
6.5.3. Definiowanie części zamiennych adresu URL

Jeśli adres URL mapy witryny zawiera symbole zastępcze (oznaczone klamrą {} jak np. {some_name}), widok *„URL Subparts”* pozwala użytkownikom określić zamienniki dla tych symboli zastępczych (zaprezentowane na rysunku 38.). W celu lepszego wskazania, który fragment URL będzie zmieniany przez elementy na liście w trakcie skanowania, część zamienna URL jest zastąpiona niebieskim prostokątem z nazwą pobraną spomiędzy klamer.



Rysunek 38. Interfejs zarządzania części zastępczych URL w trakcie definiowania robotnika. Źródło: opracowanie własne.

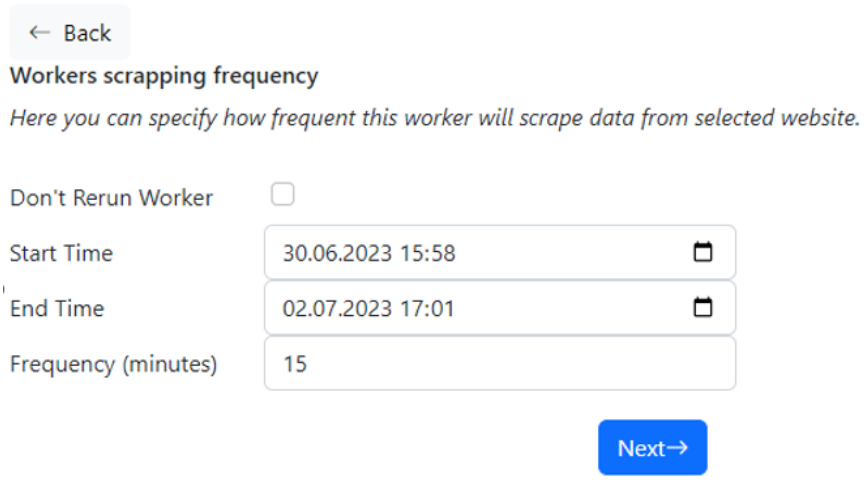
Te zamienniki są używane podczas skrapowania przez usługę serwerową, umożliwiając pracownikowi nawigację po różnych stronach witryny. Robotnik dla każdej części zamiennej z listy tworzy URL podstrony wstawiając tę część w miejsce, gdzie jest w URL słowo w klamrze. Okno modalne pozwalające na wprowadzenie kolejnego elementu zamiennego w URL jest widoczne na rysunku 39.



Rysunek 39. Okno modalne dodawania kolejnego elementu zamiennego dla {subpart} w URL. Źródło: opracowanie własne.

6.5.4. Częstotliwość automatycznego skanowania

W tym kroku użytkownicy są proszeni o wprowadzenie zakresu czasowego i częstotliwości operowania robotnika. Umożliwiający to formularz widoczny jest na rysunku 40.

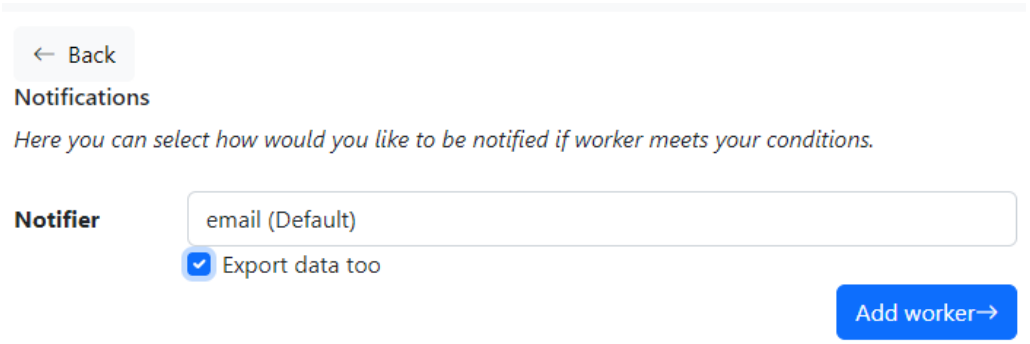


Rysunek 40. Formularz ustawień harmonogramu skanowania robotnika. Źródło: opracowanie własne.

Umożliwia to określenie częstotliwości skanowania i zdefiniowania okna czasowego, w którym pracownik powinien być aktywny. Dzięki temu można na przykład dostosować ekstrakcję danych do cyklu aktualizacji witryny lub własnych wymagań.

6.5.5. Powiadomienia

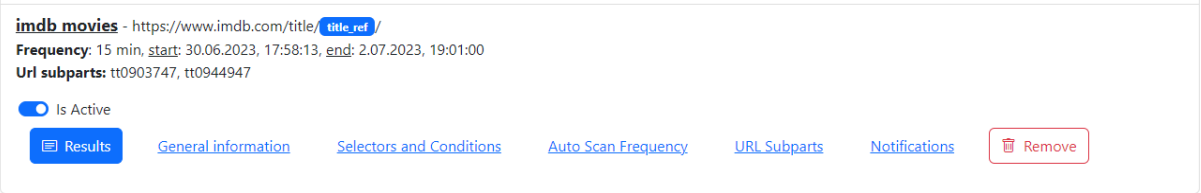
Widok *„Notifiers”* zapewnia użytkownikom elastyczność w ustawianiu preferencji dotyczących powiadomień.



Rysunek 41. Formularz wyboru sposobu powiadomień robotnika. Źródło: opracowanie własne.

W formularzu z rys. 41 istnieje możliwość wyboru preferowanego kanału powiadomień z listy. Powiadomienie zostanie wysłane, gdy robotnik spełni wcześniej zdefiniowane przez użytkowników warunki. To zapewnia, że są oni informowani o swoich wynikach skrapowania w wybrany przez nich sposób.

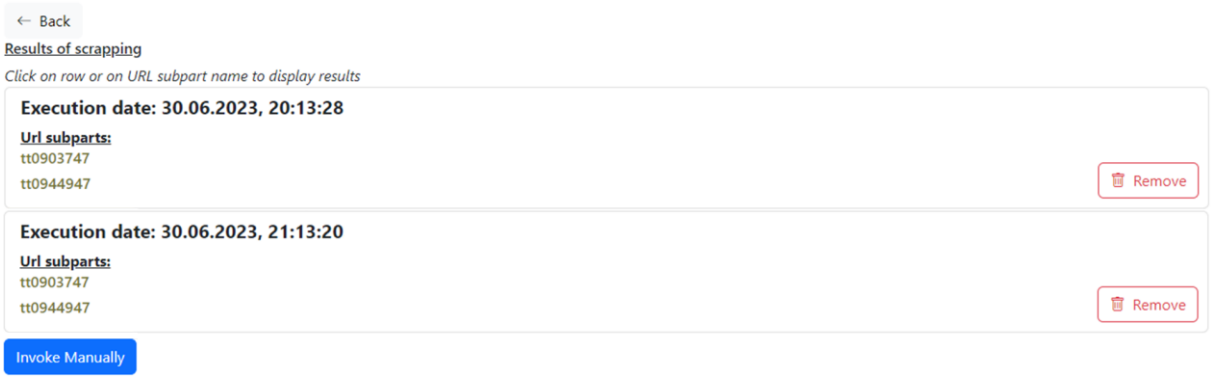
Gdy użytkownicy skonfigurują te poszczególne kroki według wymagań, mogą zapisać swoje ustawienia, a nowy robotnik zostanie dodany do listy pracowników użytkownika. Nowo dodany do listy robotnik widoczny jest na rysunku 42.



Rysunek 42. Nowo zdefiniowany wiersz w liście robotników użytkownika. Źródło: opracowanie własne.

6.6. Zarządzanie wynikami robotnika

Gdy robotnik aktywnie zbiera dane, wyniki każdej operacji są automatycznie dodawane do listy wyników (rysunek 43.).



Rysunek 43. Lista rezultatów skrapowania robotnika. Źródło: opracowanie własne.

Lista wyników, wizualnie zaprojektowana jako tabela, wyświetla kilka informacji dla każdego wyniku:

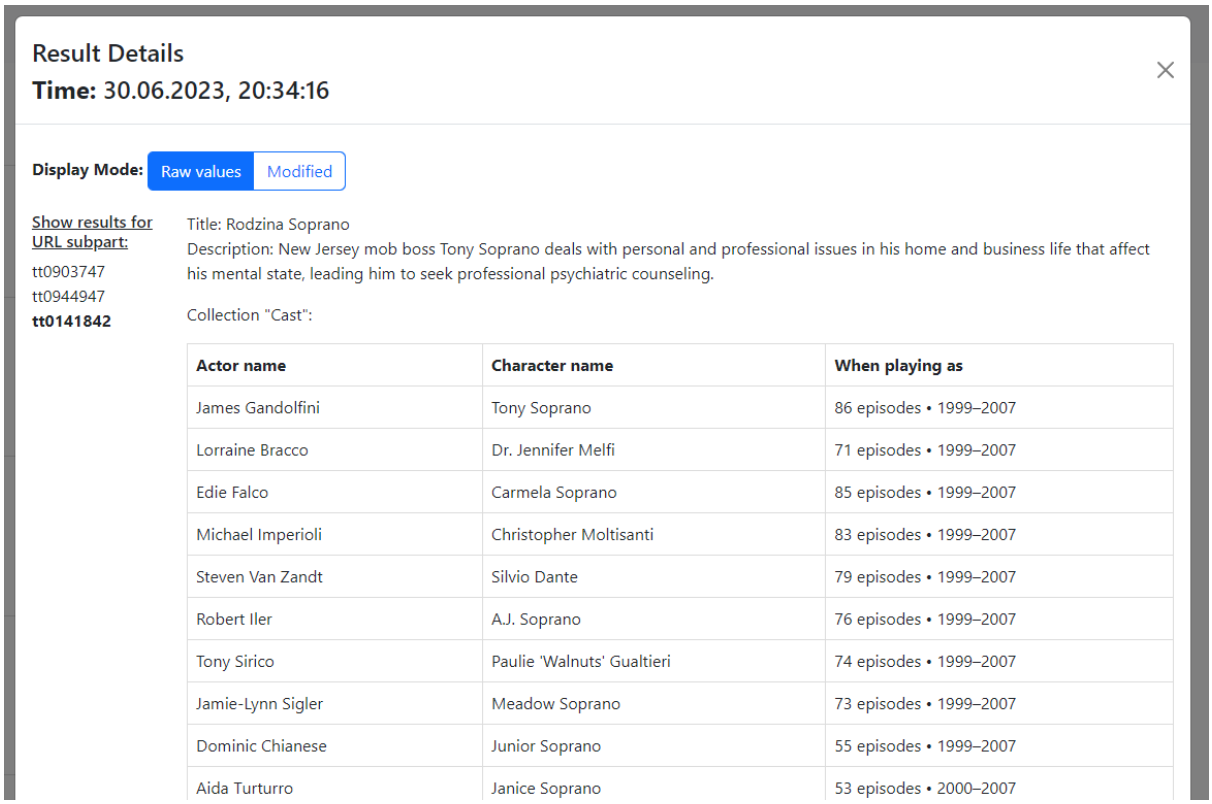
- Czas skrapowania - data i godzina przeprowadzenia operacji skrapingu.

- Podczęści URL - konkretne zamienne części adresu strony internetowej, które zostały wykorzystane podczas operacji.

Pod listą widoczny jest przycisk *„Inokve Manually”*, który umożliwia użytkownikowi manualne wywołanie skrapowania przez robotnika. Jest to przydatne, gdy użytkownik potrzebuje sprawdzić działanie robotnika i nie chce czekać na następne skanowanie z harmonogramu, lub gdy robotnik nie jest aktywny.

6.7. Przeglądanie wyników skrapowania robotników

Użytkownicy mogą zagłębić się w szczegóły poszczególnych wyników skrapowania, klikając określony wiersz na liście wyników. Ta czynność uruchamia okno modalne (rysunek 44.), które zawiera bardziej szczegółowe informacje na temat procesu skrapowania i zebranych danych.



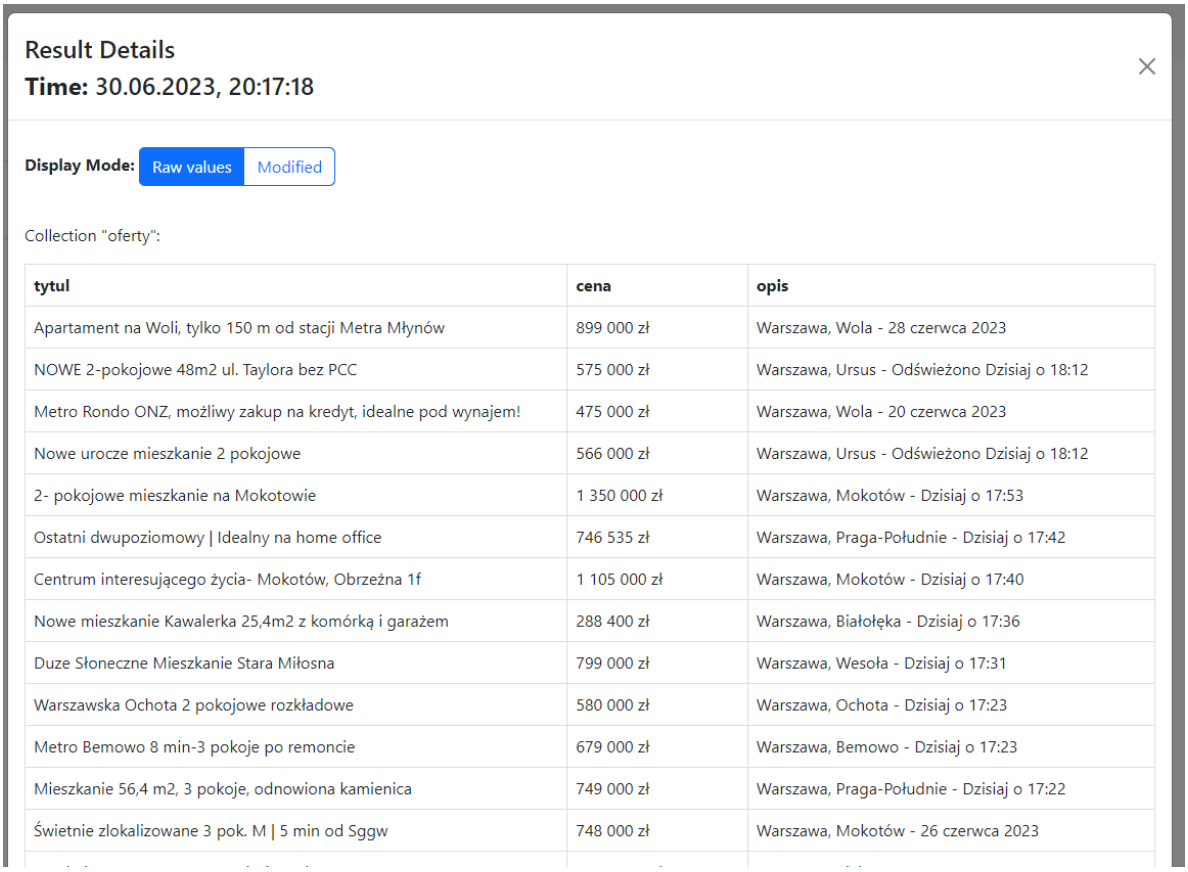
Rysunek 44. Okno modalne z wynikami skanowania robotnika. Źródło: opracowanie własne.

Domyślnie, po naciśnięciu na wiersz, okno modalne wyświetla dane dla pierwszej zamiennej części URL witryny. Jeśli istnieje więcej części, a użytkownik chce wyświetlić dane z innej, może to zrobić, klikając żądaną nazwę części w oknie modalnym lub z ogólnej listy wyników.

Użytkownik ma również możliwość wyeksportowania danych do CSV. W tym celu należy nacisnąć przycisk na dole okna modalnego i dane zostaną pobrane w domyślną lokalizację ustawioną w przeglądarce do pobierania danych. Przycisk znajduje się w dolnej część okna modalnego (zaprezentowany na rysunku 45).

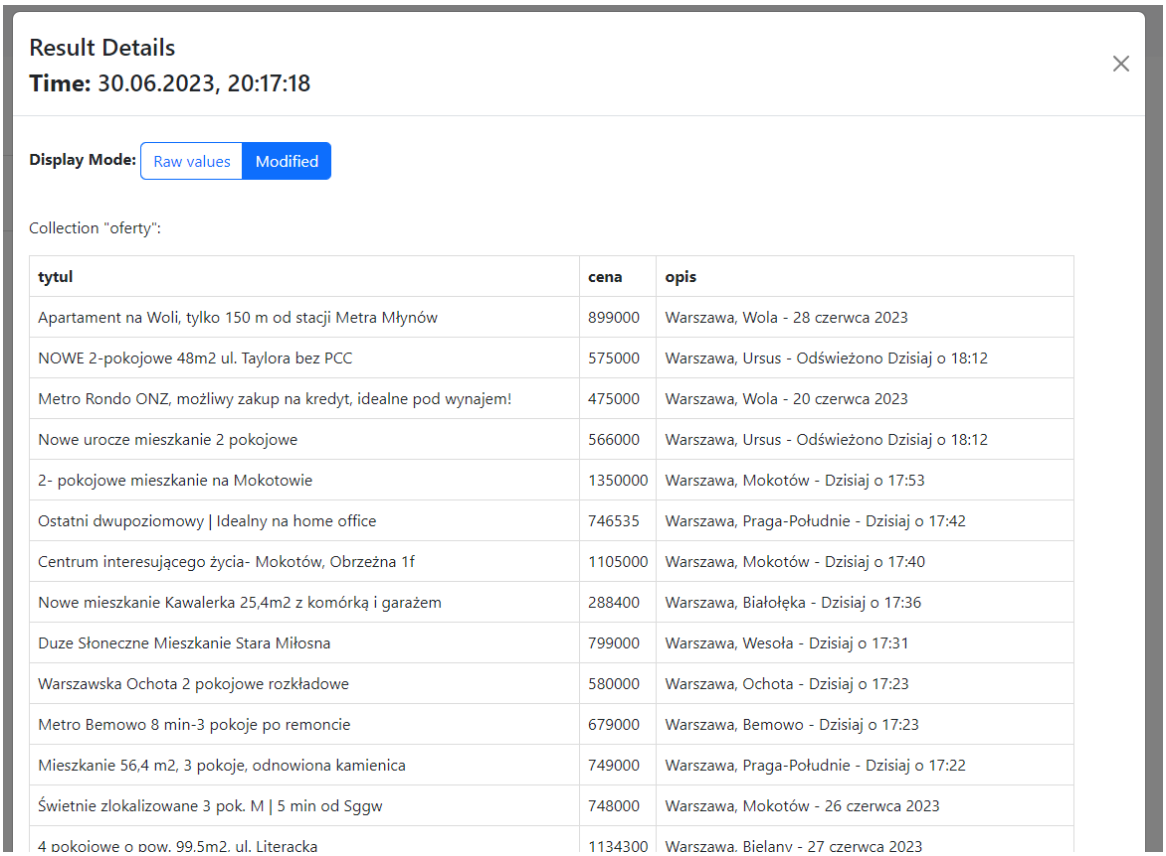
Rysunek 45. Dolna część okna modalnego z wynikami skrapowania robotnika. Źródło: opracowanie własne.

Przełącznik *„Raw value / Modified”* (widoczny na rysunku 46.) w oknie z wynikami użytkownik ma możliwość przełączania się między przeglądaniem niezmienionych danych pobranych bezpośrednio ze strony internetowej a zmodyfikowanymi danymi, które zostały przetworzone za pomocą wybranych w podrozdziale 6.5.2 - *„Selektory, modyfikatory i warunki”* modyfikatorów. Ta funkcja zapewnia wszechstronny widok danych, zaspokajając różne potrzeby analityczne. Widok z niezmienionymi danymi jest widoczny na rysunku 46, a z danymi zmodyfikowanymi zaprezentowany jest na rysunku 47.



tytul	cena	opis
Apartament na Woli, tylko 150 m od stacji Metra Młynów	899 000 zł	Warszawa, Wola - 28 czerwca 2023
NOWE 2-pokojowe 48m2 ul. Taylora bez PCC	575 000 zł	Warszawa, Ursus - Odświeżono Dzisiaj o 18:12
Metro Rondo ONZ, możliwy zakup na kredyt, idealne pod wynajem!	475 000 zł	Warszawa, Wola - 20 czerwca 2023
Nowe urocze mieszkanie 2 pokojowe	566 000 zł	Warszawa, Ursus - Odświeżono Dzisiaj o 18:12
2- pokojowe mieszkanie na Mokotowie	1 350 000 zł	Warszawa, Mokotów - Dzisiaj o 17:53
Ostatni dwupoziomowy | Idealny na home office	746 535 zł	Warszawa, Praga-Południe - Dzisiaj o 17:42
Centrum interesującego życia- Mokotów, Obrzeżna 1f	1 105 000 zł	Warszawa, Mokotów - Dzisiaj o 17:40
Nowe mieszkanie Kawalerka 25,4m2 z komórką i garażem	288 400 zł	Warszawa, Białołęka - Dzisiaj o 17:36
Duze Słoneczne Mieszkanie Stara Miłosna	799 000 zł	Warszawa, Wesoła - Dzisiaj o 17:31
Warszawska Ochota 2 pokojowe rozkładowe	580 000 zł	Warszawa, Ochota - Dzisiaj o 17:23
Metro Bemowo 8 min-3 pokoje po remoncie	679 000 zł	Warszawa, Bemowo - Dzisiaj o 17:23
Mieszkanie 56,4 m2, 3 pokoje, odnowiona kamienica	749 000 zł	Warszawa, Praga-Południe - Dzisiaj o 17:22
Świetnie zlokalizowane 3 pok. M | 5 min od Sggw	748 000 zł	Warszawa, Mokotów - 26 czerwca 2023

Rysunek 46. Okno modalne z wynikami skanowania robotnika – wybrany widok surowych danych. W tej wersji okna można również zobaczyć, jak wygląda okno skrapowania, gdy skrapowanie jest zaplanowane tylko dla jednej strony (brak wyboru podczęści URL po lewej stronie okna). Źródło: opracowanie własne.



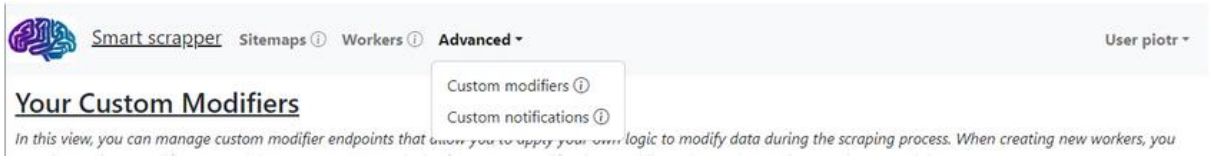
Rysunek 47. Okno modalne z wynikami skanowania robotnika – wybrany widok przetworzonych danych przez modyfikatory. Źródło: opracowanie własne.

Takie podejście do prezentacji danych zapewnia użytkownikom łatwe i kompleksowe zrozumienie wyników skrobania. Użytkownicy mogą nie tylko monitorować ogólną wydajność swoich robotników, ale mogą również zagłębić się w szczegóły poszczególnych operacji za pomocą jednego kliknięcia, uzyskując wgląd w surowe i przetworzone dane.

6.8. Funkcje zaawansowane: niestandardowe modyfikatory i sposoby powiadamiania

Dla użytkowników, którzy chcą mieć większą kontrolę i dostosować proces skrapowania danych do własnych potrzeb, system zapewnia możliwość tworzenia i zarządzania niestandardowymi modyfikatorami i sposobami powiadomień. Funkcja ta, dostępna za pośrednictwem trzeciego odnośnika w głównym pasku nawigacyjnym, jest dla użytkowników, którzy chcą rozszerzyć funkcjonalność systemu według swoich specyficznych potrzeb.

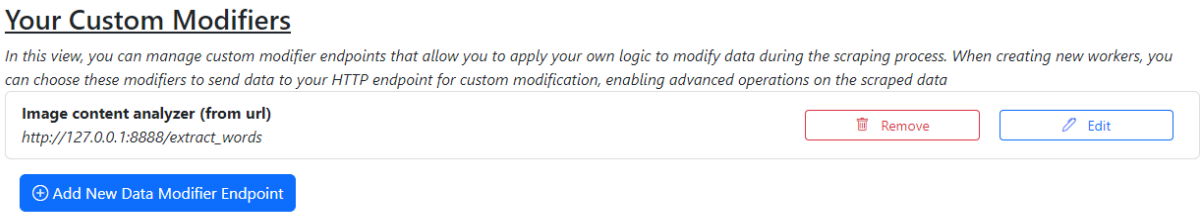
Po kliknięciu trzeciego odnośnika rozwijane menu pozwala użytkownikowi wybrać pomiędzy *„Custom Modifiers”* i *„Custom Notifiers”* (tak jak widoczne jest to na rysunku 48). Każdy wybór pokazuje adekwatny widok. Oba widoki zostaną opisane w dwóch następnych podrozdziałach.



Rysunek 48. Pasek nawigacji rozszerzenia z rozwiniętą listą odnośników do widoków zaawansowanych. Źródło: opracowanie własne.

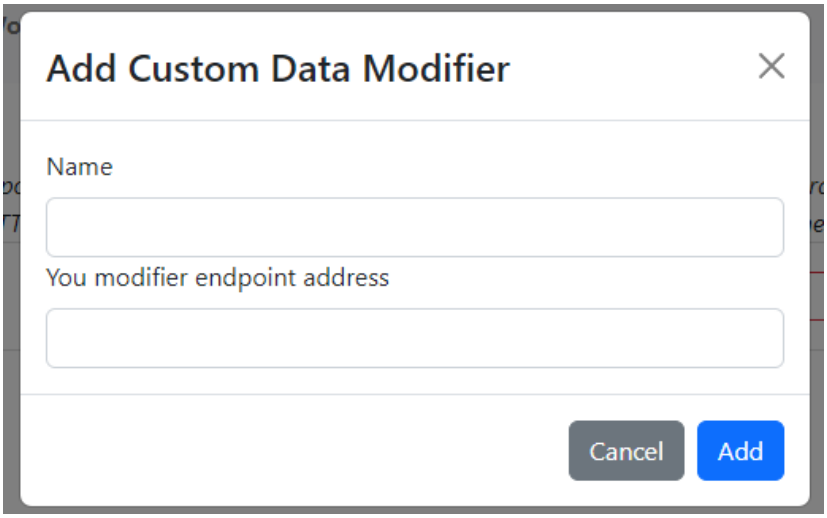
6.8.1. Modyfikatory niestandardowe

Wybór opcji *„Custom Modifiers”* w menu rozwijanym powoduje wyświetlenie listy niestandardowych modyfikatorów, które użytkownik już utworzył (rysunek 49). Każdy modyfikator można edytować lub w razie potrzeby usunąć. Oferuje to elastyczność w utrzymywaniu i aktualizowaniu modyfikatorów.



Rysunek 49. Lista niestandardowych modyfikatorów zdefiniowanych przez użytkownika. Źródło: opracowanie własne.

Aby utworzyć nowy modyfikator niestandardowy, użytkownik może nacisnąć przycisk *„Add New Data Modifier Endpoint”,* który otwiera okno modalne (rysunek 50). W tym oknie użytkownik może wprowadzić nazwę i adres nowego modyfikatora.

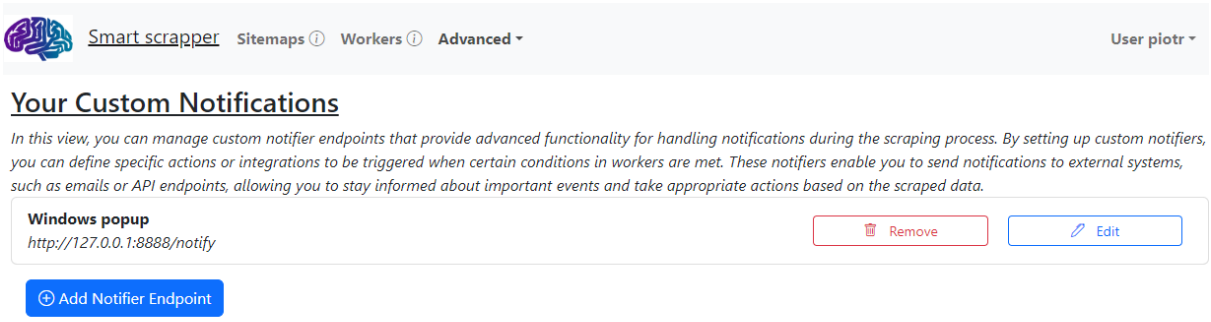


Rysunek 50. Okno modalne pozwalające na dodanie niestandardowego modyfikatora. Źródło: opracowanie własne.

Po utworzeniu te niestandardowe modyfikatory są dostępne do wyboru w sekcji opisanej w podrozdziale *„selektory, modyfikatory i warunki”* podczas procesu tworzenia lub edycji robotnika na etapie wyboru modyfikatorów dla poszczególnych selektorów w mapie witryny.

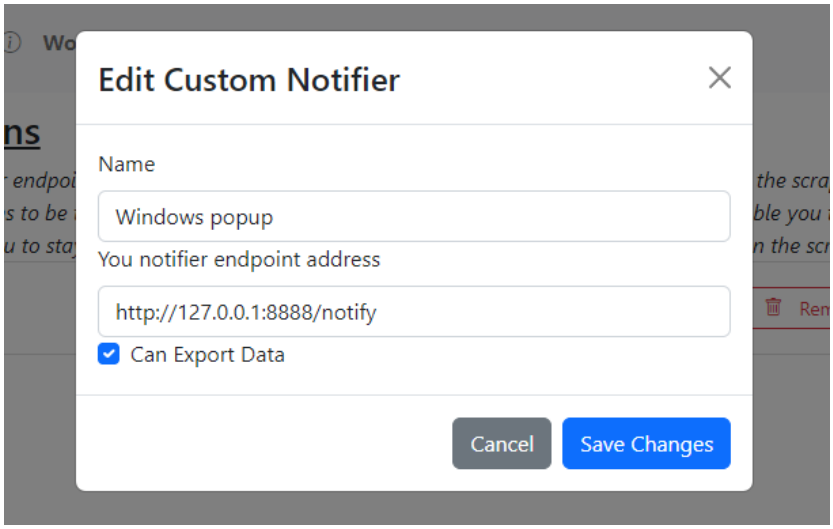
6.8.2. Niestandardowe powiadomienia

Jeśli z listy rozwijanej wybrano *„Custom notifiers”*, wyświetlany jest interfejs podobny do widoku modyfikatorów niestandardowych, ale ukierunkowany na tworzenie niestandardowych powiadomień i zarządzanie nimi (rysunek 51). Ponownie, użytkownicy mogą edytować lub usuwać istniejące obiekty powiadomień, lub dodawać nowe, klikając przycisk *„Add Notifier Endpoint”*.



Rysunek 51. Lista niestandardowych notyfikatorów zdefiniowanych przez użytkownika. Źródło: opracowanie własne.

W oknie modalnym tworzenia nowego punktu końcowego niestandardowego powiadomienia (widocznym na rysunku 52.) użytkownicy wprowadzają nazwę i adres, podobnie jak w przypadku niestandardowego modyfikatora. Jednak dodatkowa opcja *„Can Export Data”* pozwala użytkownikowi określić, czy punkt końcowy dostępny pod adresem może otrzymywać zeskrapowane dane, gdy robotnik spełnia określone warunki.



Rysunek 52. Okno modalne edycji niestandardowego punktu końcowego. Źródło: opracowanie własne.

Ta zaawansowana funkcjonalność zapewnia użytkownikom szeroką kontrolę nad procesami skrapowania danych i powiadamiania, umożliwiając im dostosowanie systemu do ich konkretnych potrzeb i zwiększenie możliwości narzędzia.

7. Podsumowanie

Ten rozdział skupia się na wynikach pracy, prezentując dogłębną analizę zalet i wad zaproponowanego rozwiązania. Ocena uwzględnia nie tylko bieżący stan prototypu, ale również możliwości jego dalszego rozwoju.

7.1. Wady i zalety rozwiązania

Opracowany prototyp narzędzia do pozyskiwania danych, udostępniany jako rozszerzenie do przeglądarek opartych na Chromium, ma na celu ułatwienie i usprawnienie procesu ekstrakcji informacji ze stron internetowych. Przy projektowaniu tego rozwiązania skupiono się na spełnieniu postawionych założeń i wymagań, aby zapewnić użytkownikom skuteczny instrument do skrapowania.

Głównym atutem tego rozwiązania jest możliwość pozyskiwania informacji z dynamicznych witryn internetowych oraz stosowanie zaawansowanych modyfikatorów na pobranych danych. System wprowadza funkcjonalność definiowania i wykorzystywania zadań cyklicznych, umożliwiając planowanie określonych działań skrapingu w określonym czasie i z określoną częstotliwością. Pozwala to na automatyzację procesu pozyskiwania informacji i regularne aktualizowanie danych z wybranych źródeł. Użytkownicy mogą określić również warunki, które muszą zostać spełnione w trakcie skrapowania cyklicznego, aby otrzymać powiadomienie.

Dodatkowo prototyp oferuje również możliwość wykorzystywania modeli nauczania maszynowego, takich jak lematyzacja i detekcja obiektów na obrazach w roli dodatkowych modyfikatorów. Dzięki temu zapewnia jeszcze większe możliwości pozyskiwania i przekształcania uzyskanych danych.

Interfejs graficzny jest intuicyjny i łatwy w obsłudze. Proces wybierania informacji polega na zaznaczaniu określonych przez użytkownika elementów na wczytanej w przeglądarce stronie. Umieszczenie interfejsu w panelu narzędzi deweloperskich przeglądarki, ułatwia nawigację, zapewniając jednoczesny podgląd GUI i witryny internetowej.

System umożliwia również definiowanie własnych punktów końcowych dla modyfikatorów danych i powiadomień. Zwiększa to elastyczność i umożliwia tworzenie bardziej spersonalizowanych i wyspecjalizowanych sposobów przekształcania informacji. Daje to bardziej zaawansowanym technicznie użytkownikom możliwość dostosowania przebiegu działania skrapowania i powiadamiania o jego wynikach do własnych potrzeb. Ponadto rozszerzenie oferuje możliwość eksportowania pozyskanych rekordów do formatu CSV oraz przeglądania zarówno surowych, jak i zmodyfikowanych danych pozyskanych przez zadania cykliczne.

Niemniej jednak, prototyp nie jest pozbawiony pewnych ograniczeń. Jednym z nich jest wrażliwość na zmiany na stronie - po zmianie struktury HTML strony, używane selektory mogą przestać wskazywać wybrane wcześniej elementy. Taki stan wymaga interwencji użytkownika. Ponadto brak systemu proxy uniemożliwia przekierowywanie zapytań o wczytanie stron na inne adresy IP, co może prowadzić do blokad dostępu przez niektóre witryny.

7.2. Dalszy rozwój

Podczas dalszego rozwoju prototypu, oprócz już wymienionych ograniczeń, które trzeba wyeliminować w pierwszej kolejności, istnieje jeszcze kilka kluczowych obszarów, które warto wziąć pod uwagę w celu zapewnienia jeszcze lepszego doświadczenia użytkownikom. Poniżej przedstawiono kilka propozycji usprawnień:

- Ograniczenie liczby zapytań skrapowania na użytkownika: w celu uniknięcia nadużywania systemu i możliwych blokad ze strony skrapowanych witryn, istotne jest wprowadzenie ograniczeń co do liczby żądań skrapowania na użytkownika. Wprowadzenie mechanizmu

liczenia i monitorowania liczby żądań, a także ustalenie optymalnego limitu liczby skrapowań dla użytkowników, pozwoli na ochronę przed nadmiernym wykorzystywaniem systemu.

- Rozbudowa modyfikatorów danych: rozbudowa modyfikatorów danych stanowi kolejny obszar, który wymaga dalszego udoskonalania. Należy poszerzyć zestaw już istniejących sposobów transformacji danych, aby umożliwić użytkownikom jeszcze większą możliwość dostosowania procesu skrapowania do ich indywidualnych potrzeb. Ponadto warto rozważyć wprowadzenie kolejnych modeli nauczania maszynowego jako dodatkowych modyfikatorów danych. Przykładowe modele, które można wprowadzić to:
    - Analiza sentymentu: model pomocny w identyfikacji i ekstrakcji opinii z danych tekstowych na stronach internetowych. Może to być przydatne w monitorowaniu mediów społecznościowych, monitorowaniu marek i recenzjach produktów.
    - Podsumowywanie tekstu: model NLP do generowania krótkiego podsumowania długich artykułów, postów na blogach lub innych danych tekstowych pobranych ze stron internetowych.
- Rozbudowanie zestawu opcji powiadomień dla użytkownika: poszerzenie zestawu opcji powiadomień użytkownika pozwoli na jeszcze lepszą personalizację procesu skrapowania. Możliwość wyboru formy powiadomienia, takiej jak powiadomienia pop-up na systemach Windows (które w przypadku aktualnej implementacji trzeba byłoby przenieść z niestandardowych notyfikatorów do zwykłych) lub powiadomienia na urządzenia mobilne, umożliwi użytkownikom dostęp do informacji o wynikach skrapowania w wygodniejszy sposób.
- Dodanie oddzielnej strony internetowej do zarządzania robotnikami: stworzenie dedykowanej strony, która otwiera się jako osobna karta w przeglądarce, umożliwi użytkownikom zarządzanie zadaniami cyklicznymi w wygodny sposób – bez konieczności używania rozszerzenia. Na tej stronie użytkownicy mieliby możliwość dodawania, edycji i usuwania zadań cyklicznych, a także monitorowania ich postępu i wyników. Strona ta mogłaby być używana nawet na urządzeniach mobilnych.
- Rozszerzenie możliwości eksportu danych: obecnie narzędzie umożliwia eksport danych do formatu CSV, jednak warto rozważyć dodanie wsparcia dla innych formatów, takich jak XML.
- Zrobienie portu rozszerzenia do przeglądarki Mozilla Firefox.
- Dodanie możliwości współdzielenia projektów pomiędzy użytkownikami.

7.3. Wnioski końcowe

Podsumowując, prototyp narzędzia do pozyskiwania danych, dostępny jako rozszerzenie do przeglądarek opartych na Chromium ma wiele zalet i możliwości, które przyczyniają się do usprawnienia procesu ekstrakcji informacji ze stron internetowych. Skrapowanie stron dynamicznych, funkcjonalność zadań cyklicznych, intuicyjny interfejs użytkownika oraz możliwość wykorzystywania modyfikatorów i powiadomień stanowią kluczowe atuty rozwiązania.

Dodatkowo narzędzie umożliwia wykorzystanie modeli nauczania maszynowego do modyfikacji pobranych danych, co rozszerza możliwości optymalizacji i przekształcania pozyskiwanych informacji. Otwiera to nowe perspektywy dla użytkowników, umożliwiając bardziej zaawansowane operacje i łatwiejszą analizę danych.

W celu dalszego rozwoju systemu istotne jest skupienie starań na eliminacji wcześniej wymienionych wad oraz wprowadzeniu proponowanych usprawnień. Warto też kontynuować rozwój narzędzia, uwzględniając potencjał wykorzystania innych modeli nauczania maszynowego.

Bibliografia

[1] R. Mitchell *Web Scraping with Python: Collecting More Data from the Modern Web 2nd Edition*, O'Reilly 2018

[2] Scrape, Meaning of scrape in English, https://dictionary.cambridge.org/dictionary/english/scrape [Dostęp: 20.06.2023]

[3] Ustawa z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. 1994 nr 24 poz. 83)

[4] Ustawa z dnia 27 lipca 2001 r. o ochronie baz danych (Dz.U. 2001 nr 128 poz. 1402)

[5] Dyrektywa 96/9/WE Parlamentu Europejskiego i Rady. z dnia 11 marca 1996 r. w sprawie ochrony prawnej baz danych

[6] ROZPORZĄDZENIE PARLAMENTU EUROPEJSKIEGO I RADY (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych)

[7] Chromium, https://www.chromium.org/chromium-projects/ [Dostęp: 02.07.2023]

[8] Zyte.com, https://www.zyte.com/ [Dostęp: 15.08.2023]

[9] WebScrapper, https://webscraper.io/ [Dostęp: 15.08.2023]

[10] Octoparse, https://www.octoparse.com/ [Dostęp: 15.08.2023]

[11] Datahen, https://www.datahen.com/ [Dostęp: 15.08.2023]

[12] RESTful, https://developer.ibm.com/articles/ws-restful/ [Dostęp: 22.06.2023]

[13] HTML: HyperText Markup Language, https://developer.mozilla.org/en-US/docs/Web/HTML [Dostęp: 22.06.2023]

[14] Document Object Model, https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction [Dostęp: 22.06.2023]

[15] Jesse James Garrett *Ajax: A New Approach to Web Applications*, Adaptive Path 2005

[16] K. Kwapisz *Web scraping Selenium*, https://kamil.kwapisz.pl/web-scraping-selenium/ [Dostęp: 15.08.2023]

[17] Cascading Style Sheets – CSS selectors (:nth-child()), https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child [Dostęp: 6.07.2023]

[18] Typescript, https://www.typescriptlang.org/ [Dostęp: 02.07.2023]

[19] ECMAScript Modules in Node.js https://www.typescriptlang.org/docs/handbook/esm-node.html [Dostęp: 13.08.2023]

[20] React, https://react.dev/ [Dostęp: 02.07.2023]

[21] React-Bootstrap, https://react-bootstrap.netlify.app/docs/ [Dostęp: 17.08.2023]

[22] Django, https://www.djangoproject.com/ [Dostęp: 19.08.2023]

[23] Django - faq, https://docs.djangoproject.com/pl/4.2/faq/general/ [Dostęp: 10.07.2023]

[24] Selenium, https://www.selenium.dev/documentation/ [Dostęp: 02.07.2023]

[25] Celery, https://docs.celeryq.dev/en/stable/ [Dostęp: 17.08.2023]

[26] Beautiful Soup Documentation, https://www.crummy.com/software/BeautifulSoup/bs4/doc/ [Dostęp: 19.08.2023]

[27] Cascading Style Sheets, https://developer.mozilla.org/en-US/docs/Web/CSS [Dostęp: 22.06.2023]

[28] Usługi REST, https://www.ibm.com/docs/pl/rtw/9.1.0?topic=mode-rest-services [Dostęp: 17.08.2023]

[29] Introduction to Web Services, https://netbeans.apache.org/kb/docs/websvc/intro-ws.html [Dostęp: 22.06.2023]

[30] Postman, https://learning.postman.com/docs [Dostęp: 22.06.2023]

[31] Webstorm, https://www.jetbrains.com/webstorm/

[32] SpaCy, https://spacy.io [Dostęp: 22.06.2023]

[33] Spacy - model pl_core_news_sm, https://spacy.io/models/pl [Dostęp: 12.07.2023]

[34] Quickstart: Detecting named entities (NER), https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/named-entity-recognition/quickstart [Dostęp: 12.07.2023]

[35] C. Wan, Y. Pang, S. Lan. *Overview of YOLO Object Detection Algorithm*. International Journal of Computing and Information Technology, Vol. 1, No. 2, page 11, 2022. DOI: 10.56028/ijcit.1.2.11

[36] Component Architecture, https://handsonreact.com/docs/component-architecture [Dostęp: 22.08.2023]

[37] Webpack, https://webpack.js.org/ [Dostęp: 21.08.2023]

[38] Ts-loader, *https://www.npmjs.com/package/ts-loader* [Dostęp: 21.08.2023]

[39] Chrome extensions, https://developer.chrome.com/docs/extensions/reference [Dostęp: 10.07.2023]

[40] Chrome extensions – Messaging, https://developer.chrome.com/docs/extensions/mv3/messaging/ [Dostęp: 10.07.2023]

Spis rysunków

Spis listingów