

Ultra Price Tracker

Aplikacja Internetowa śledząca ceny produktów szukanych przez użytkowników.

Web Scraper wyszukujący informacje o produkcie w różnych stronach internetowych.

Maciej Owsianny

Gracjan Trawiński

Łukasz Wełnic

Spis treści

Spis treści	1
Opis aplikacji To check	2
Podział prac TODO	2
Wykorzystane technologie TODO	3
Architektura TODO	3
Funkcjonalności	3
Szczegóły implementacyjne	3
Trudności w implementacji	3
Instrukcja użytkowania aplikacji	3

1. Opis aplikacji

Ultra Price Tracker to aplikacja internetowa umożliwiająca śledzenie cen produktów elektronicznych (m.in. RTV, AGD, smartfony, itd.). Po wprowadzeniu nazwy szukanego produktu lub linku do produktu (w sklepie obsługiwanym przez aplikację) wyświetlane są dane na jego temat. Informacje te są pozyskiwane za pomocą web scrapera. Wyświetlane dane dotyczą nazwy produktu oraz historii jego cen w określonych sklepach internetowych. Sklepy internetowe obsługiwane przez aplikację: morele.net, komputronik.pl, euro.com.pl, mediaexpert.pl.

2. Podział prac

Członek zespołu	Zakres obowiązków	Uwagi
Maciej Owsiany	Implementacja aplikacji internetowej wyświetlającej dane pozyskane za pomocą web scrapera oraz odpowiedzialnej za komunikację użytkownik - serwer. Implementacja serwera REST api	
Gracjan Trawiński	Implementacja funkcjonalności pobierania aktualnych cen produktów ze stron internetowych. Implementacja funkcjonalności wyszukiwania produktu w sklepach Internetowych przy wprowadzeniu wyrażenia (np. "Smartfon X30 Pro"). Implementacja funkcjonalności wyszukiwania produktu w sklepach Internetowych przy wprowadzeniu URL obsługiwanego sklepu.	Aby funkcje programu działały poprawnie wymagany jest Chromedriver w wersji zgodnej z przeglądarką Google Chrome.

	Implementacja funkcjonalności wyszukującej brakujących cen w przypadku braku informacji o danym produkcie w określonych sklepach.	
Łukasz Wełnic	Implementacja funkcjonalności i współpracy aplikacji z bazą danych MongoDB. Utworzenie bazy danych w chmurze www.mongodb.com . Tworzenie instancji produktów w bazie z inkrementującym objectId, jego modyfikacje (np. dodawanie ceny z danego dnia do sklepu) oraz pobieranie danych z bazy.	

Tabela 1. Zakres obowiązków.

Powyższa tabela określa zakres obowiązków każdego z członków zespołu związanych z realizacją projektu.

3. Wykorzystane technologie

Cały system składa się z dwóch części - serwera oraz aplikacji internetowej.

Serwer został zaimplementowany w oparciu o język programowania Python oraz bazę danych MongoDB. Aplikacja internetowa została zaimplementowana wykorzystując framework Angular.

3.1. Interfejs użytkownika

Angular - platforma programistyczna do tworzenia wydajnych i wyrafinowanych aplikacji jednostronicowych. Wykorzystuje język programowania TypeScript.

Dodatkowe biblioteki:

- NgRx - framework wspomagający tworzenie reaktywnych aplikacji za pomocą frameworku Angular. Udostępnia biblioteki umożliwiające:
 - Zarządzanie stanem globalnym i lokalnym.
 - Narzędzia programistyczne ułatwiające debugowanie aplikacji oraz podgląd stanu.
- MdBootstrap - biblioteka CSS udostępniająca zestaw narzędzi ułatwiających tworzenie interfejsu graficznego aplikacji internetowych. Wspomaga stylizację aplikacji tworzonych za pomocą frameworku Angular.
- Chart.js - to darmowa biblioteka JavaScript typu open source do wizualizacji danych obsługująca wiele typów wykresów.
- Ng2-charts - biblioteka do tworzenia reaktywnych oraz responsywnych wykresów wykorzystująca bibliotekę Chart.js. Wspomaga tworzenie wykresów z wykorzystaniem frameworka Angular.

3.2. Serwer

Python - wysokopoziomowy, imperatywny język programowania - został wybrany ze względu na rozbudowany pakiet bibliotek, szybkość działania oraz automatyczne zarządzanie pamięcią. Python jest dostępny na wielu platformach oraz posiada w pełni dynamiczny system typów. Python dobrze spełnia zadanie w przetwarzaniu danych oraz jest łatwy w odczycie i interpretacji. Ten język programowania jest szeroko wykorzystywany w *Web Scraping-u*.

Dodatkowe biblioteki:

- Selenium - umożliwia obsługę narzędzia wykorzystywanego w celu automatyzacji wykonywania operacji przez przeglądarkę. Zostało ono wykorzystane w celu wyszukiwania produktów na stronach internetowych a następnie zwracaniu wyników wyszukiwania w postaci strony do pozostałych funkcji.
- BeautifulSoup4 - biblioteka języka python wykorzystywana do ekstrakcji danych z plików HTML oraz XML. Jest kompatybilny z wieloma parserami co pozwala na jego szerokie zastosowanie. Biblioteka została wybrana ze względu na szybkość działania oraz na kompleksowość rozwiązania. Oferuje

również obszerną dokumentację pozwalającą na lepsze wykorzystanie funkcjonalności biblioteki. W aplikacji umożliwia odczytanie danych ze stron pozyskanych przez inne funkcje (te oparte o Selenium).

- Fuzzywuzzy - biblioteka oparta na obliczaniu odległości Levenshteina służąca do porównywania ciągów znaków. Wykorzystywany jest przy szukaniu produktu, którego nazwa najbardziej pasuje do frazy wprowadzonej przez użytkownika.

3.3. Baza danych

Mongo DB - otwarty, nierelacyjny system zarządzania bazą danych napisany w języku C++. Został wybrany ze względu na łatwość obsługi, bardzo dobrą dokumentację zawartą na stronie producenta oraz przechowywanie danych w postaci dokumentów. Umożliwia aplikacji naturalne ich przetwarzanie, przy zachowaniu możliwości tworzenia hierarchii oraz indeksowania. Dzięki temu bardzo łatwo odnaleźć się w odczytywaniu tego typu dokumentów.

3.4. Repozytorium

GitHub - hostingowy serwis internetowy przeznaczony dla projektów programistycznych wykorzystujących system kontroli wersji Git. GitHub udostępnia darmowy hosting programów o otwartym kodzie źródłowym i prywatnych repozytoriów. Serwis oferuje również możliwość zarządzania przepływem danych, wgląd w historię zmian oraz szereg możliwości zarządzania projektem oraz zadaniami z nim związanymi.

3.5. Oprogramowanie

JetBrains PyCharm - wieloplatformowe zintegrowane środowisko programistyczne przeznaczona dla języka Python. Pozwala na analizę i edycję kodu źródłowego, uruchamianie testów jednostkowych, posiada graficzny debugger oraz umożliwia integrację z VCS (ang. *Version Control System*). Ułatwia instalowanie zewnętrznych bibliotek. Oprogramowanie to zostało wybrane ze względu na przyjazność

użytkowania oraz ze względu na licencję wersji Professional dostarczoną przez uczelnię.

Visual Studio Code - darmowy, desktopowy edytor programistycznych kodów źródłowych z kolorowaniem składni dla wielu języków, stworzony przez Microsoft oraz dystrybuowany na licencji MIT o otwartym kodzie źródłowym.

4. Architektura

Aplikacja została stworzona w oparciu o architekturę klient - serwer. Dzięki takiemu rozwiązaniu możliwe jest korzystanie z aplikacji przez wielu klientów jednocześnie. Komunikacja pomiędzy klientem, a serwerem jest realizowana za pomocą interfejsu web api udostępnianego przez serwer. Zadaniem aplikacji klienckiej jest wysyłanie zapytań do serwera w celu otrzymania żądanych danych. Serwer oczekuje na zapytania wysłane przez klientów, przetwarza żądanie oraz przesyła klientowi odpowiedź.

5. Funkcjonalności

- Interfejs użytkownika
 - umożliwia wysyłanie zapytań do serwera w celu pozyskania danych nt. szukanego produktu.
 - zapewnia wyświetlanie danych dotyczących produktu tj. jego nazwę, obecną cenę oraz historię cen w kontekście określonego sklepu internetowego analizowanego przez system
 - umożliwia podgląd wykresu zmian cen na przestrzeni czasu
 - udostępnia możliwość podglądu produktu na stronie internetowej sklepu udostępniającego produkt
- Wyszukiwanie informacji o produkcie - dane o produkcie są wyszukiwane na podstawie frazy wprowadzonej przez użytkownika. Wyszukiwanie odbywa się na dwa sposoby:
 1. W przypadku, gdy baza danych zawiera informacje nt. adresów Url przedmiotu znajdującego się w obsługiwanych sklepach, za

pomocą **Beautiful Soup** pozyskiwane są informacje o produkcie.

2. W przypadku, gdy brakuje adresów w bazie danych, wykorzystywane jest narzędzie **selenium** do pozyskania odpowiednich adresów Url, a następnie wykonywane są czynności z kroku 1.

- Aktualizowanie informacji o produkcie - jeżeli dane o produkcie są w bazie ich ceny są aktualizowane. W przypadku, w którym nie wszystkie dane o sklepach są uzupełnione wówczas program próbuje wyszukać brakujące dane a następnie aktualizuje wszystkie otrzymane rekordy w bazie danych. Kiedy w bazie istniejące dane są kompletne, wówczas informacje o produkcie są aktualizowane z wykorzystaniem linków do produktu z bazy danych.
-

6. Szczegóły implementacyjne

6.1 Podsystem obsługujący web scraper:

a) Wyszukiwanie danych w oparciu o wprowadzoną frazę

- Otrzymana fraza jest wyszukiwana w trzech sklepach o najbogatszym asortymencie wykorzystując do tego Selenium.
- Wyniki wyszukiwania są porównywane ze sobą, a ten najbardziej pasujący do wprowadzonego przez użytkownika wybierany jest jako parametr dla pozostałych stron.
- Pozostałe strony są przeszukiwane pod kątem znalezionej frazy przy wykorzystaniu Selenium.
- Źródła stron są następnie przekazywane do interpretacji i ekstrakcji danych z wykorzystaniem biblioteki BeautifulSoup4.
- Dane o produkcie przekazywane są dalej do bazy danych.

b) Aktualizowanie cen produktów

- Kolekcja danych z linkami do produktu w poszczególnych sklepach jest przekazywana do funkcji odpowiedzialnej za zwrócenie aktualnych cen produktów.
- Każdy link jest przesyłany do odpowiedniej funkcji, która jest uruchomiona w nowym procesie.
- Program odczytuje aktualne ceny produktów przy wykorzystaniu biblioteki BeautifulSoup4.
- Dane są zwracane w postaci kolekcji z najnowszymi danymi.

c) Wyszukiwanie brakujących informacji o produkcie

- Aplikacja otrzymując niekompletne dane próbuje sprawdzić, czy dany produkt pojawił się po pewnym czasie w sklepie.
- W celu pozyskania informacji o produkcie aplikacja wykorzystując Selenium próbuje wyszukać dany produkt na wszystkich stronach.
- Dane są zwracane w postaci kolekcji ze wszystkimi danymi.
- W przypadku braku obecności danego produktu, wówczas do danego sklepu w kolekcji przypisana jest wartość "**None**".

6.2 Podsystem obsługujący bazę danych.

a) Dodawanie produktu do bazy danych.

- Sprawdzenie czy obiekt nie znajduje się w bazie danych
- Dodanie produktu do bazy danych

b) Aktualizacja produktu w bazie danych

- Wyszukanie produktu na podstawie jego nazwy
- Aktualizacja danych

6.3 Graficzny interfejs użytkownika

Składa się z 4 komponentów, z których każdy obsługuje pewny fragment widoku:

a) Chart

- odpowiedzialny za tworzenie wykresu na podstawie otrzymanych danych

b) Item-display

- Otrzymywanie odpowiedzi serwera z danymi do wyświetlenia
 - Wyświetlenie informacji dot. wyszukiwanego produktu oraz komponentu Chart
- c) Navbar
- Widok menu nawigacji
- d) Search bar
- Komponent odpowiedzialny za przesyłanie zapytań do serwera

7. Trudności w implementacji.

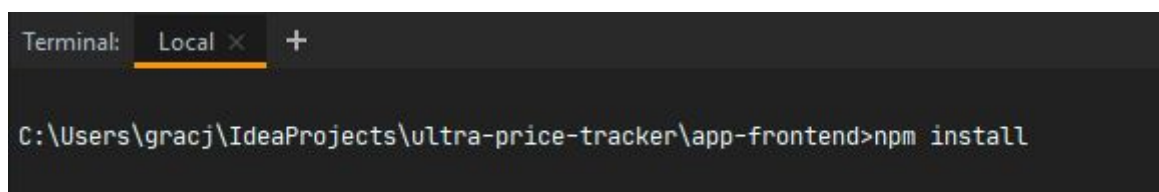
Głównym problemem działania aplikacji był czas wykonywania obsługi zlecenia. Na początku zajmowało to wiele sekund, ponieważ każdy ze sklepów był przeszukiwany po kolei. Rozwiązaniem tego problemu było wykorzystanie współbieżności. W obecnej wersji programu, wykorzystywana jest wieloprocusowość. Każdy ze sklepów jest obsługiwany w tym samym czasie posiadając swój własny proces co zredukowało kilkukrotnie czas oczekiwania na rezultaty.

8. Instrukcja użytkowania aplikacji

Uruchomienie całej aplikacji wymaga uruchomienia zarówno serwera jak i aplikacji internetowej.

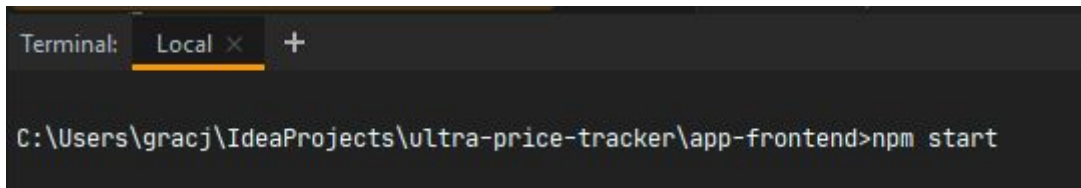
8.1. Uruchomienie aplikacji internetowej.

W celu uruchomienia aplikacji internetowej należy posiadać zainstalowany node package manager (npm). Następnie w folderze **app-frontend** poleceniem **npm install** należy zainstalować wymagane biblioteki. Ostatnim krokiem jest uruchomienie polecenia **npm start** w celu uruchomienia serwera hostującego aplikację. Aplikacja jest domyślnie dostępna na porcie 4200 (<http://localhost:4200>).



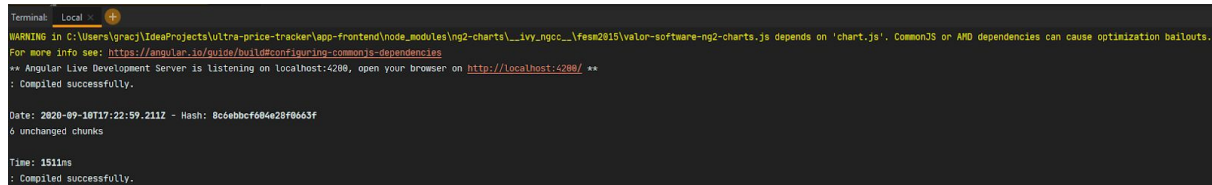
```
Terminal: Local x +  
C:\Users\gracj\IdeaProjects\ultra-price-tracker\app-frontend>npm install
```

Zrzut 1. Widok konsoli - polecenie **npm install**.



```
Terminal: Local x +  
  
C:\Users\gracj\IdeaProjects\ultra-price-tracker\app-frontend>npm start
```

Zrzut 2. Widok konsoli - polecenie **npm start**.

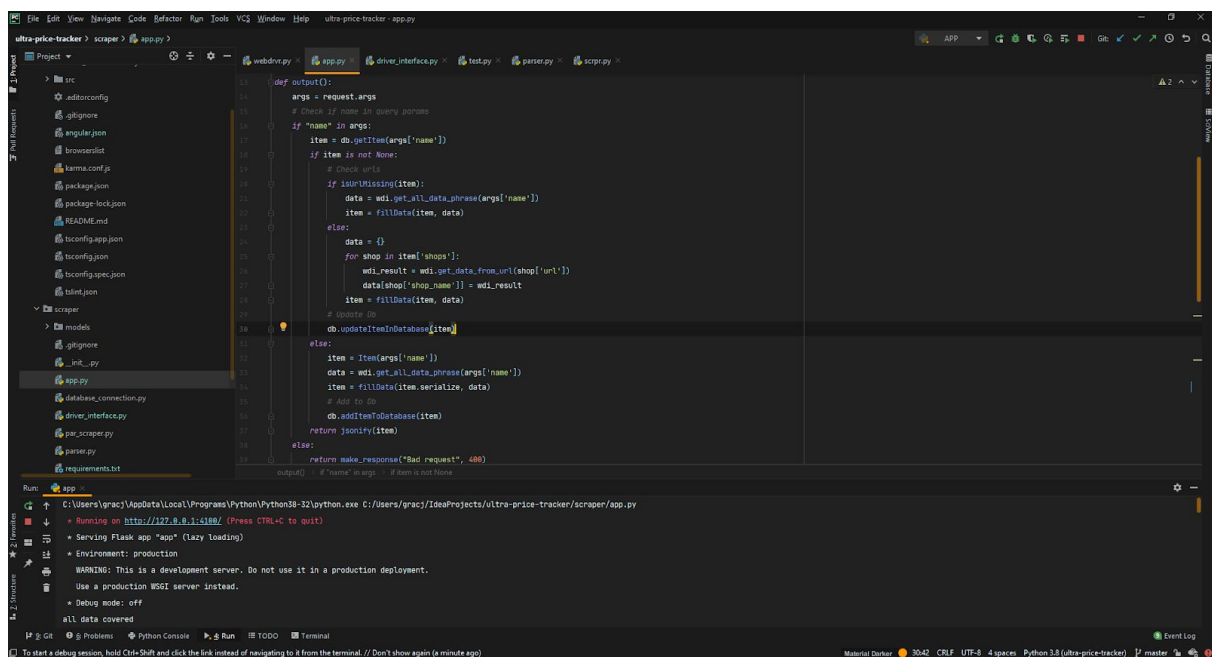


```
Terminal: Local x +  
  
WARNING in C:\Users\gracj\IdeaProjects\ultra-price-tracker\app-frontend\node_modules\ng2-charts\_ivy_ngcc_\fesm2015\valor-software-ng2-charts.js depends on 'chart.js'. CommonJS or AMD dependencies can cause optimization bailouts.  
For more info see: https://angular.io/guide/build#configuring-commonjs-dependencies  
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **  
: Compiled successfully.  
  
Date: 2020-09-10T17:22:59.211Z - Hash: 8c6ebbcf684e28f0663f  
6 unchanged chunks  
  
Time: 1511ms  
: Compiled successfully.
```

Zrzut 3. Widok konsoli - uruchomienie serwera hostującego aplikację internetową.

8.2. Uruchomienie serwera.

Włączenie serwera jest realizowane uruchamiając plik **app.py** znajdujący się w folderze **scraper**. Po włączeniu serwer nasłuchuje na porcie **4100**.

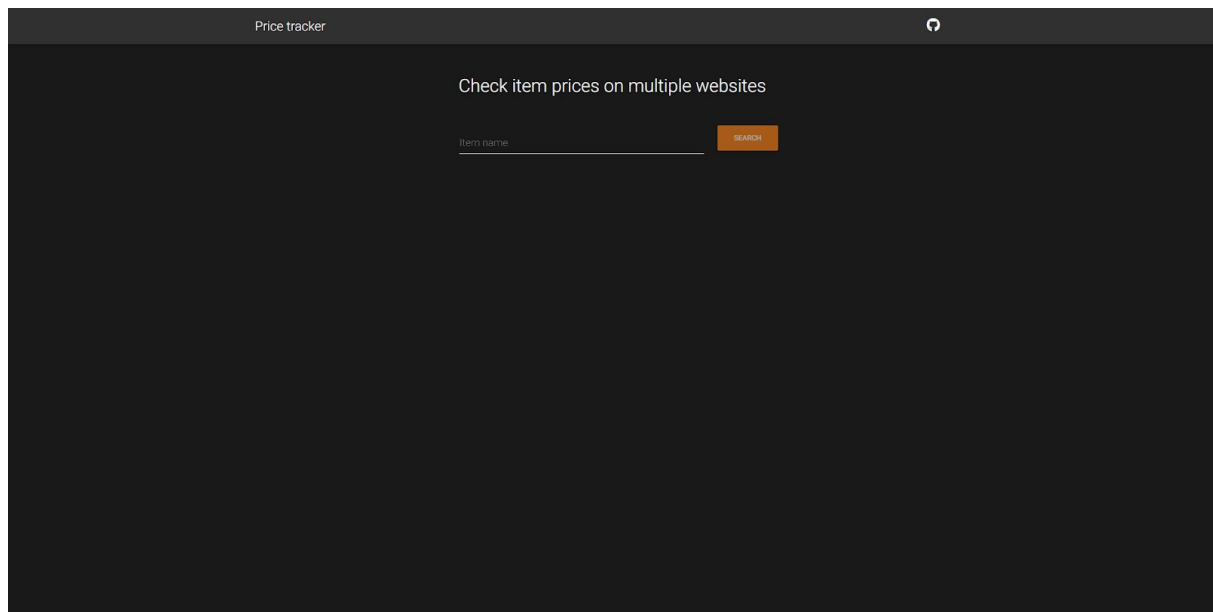


```
File Edit View Navigate Code Refactor Run Tools VCS Window Help ultra-price-tracker - app.py  
ultra-price-tracker > scraper > app.py  
Project  
src  
  addtconfig  
  gignore  
  angular.json  
  browserlist  
  karma.conf.js  
  package.json  
  package-lock.json  
  README.md  
  tsconfig.app.json  
  tsconfig.json  
  tsconfig.spec.json  
  tslint.json  
scraper  
  models  
  gignore  
  init_.py  
  app.py  
  database_connection.py  
  driver_interface.py  
  par_scraper.py  
  parser.py  
  requirements.txt  
Run  
  app  
  C:\Users\gracj\AppData\Local\Programs\Python\Python38-32\python.exe C:\Users\gracj\IdeaProjects\ultra-price-tracker\scraper\app.py  
  Running on http://127.0.0.1:4100/ (Press CTRL+C to quit)  
  Serving Flask app "app" (lazy loading)  
  Environment: production  
  WARNING: This is a development server. Do not use it in a production deployment.  
  Use a production WSGI server instead.  
  Debug mode: off  
  all data covered  
  To start a debug session, hold Ctrl-Shift and click the link instead of navigating to it from the terminal. // Don't show again (a minute ago)  
  Material Darker 3042 CRLF UTF-8 4 spaces Python 3.8 (ultra-price-tracker) master Event Log
```

Zrzut 4. Uruchomienie serwera.

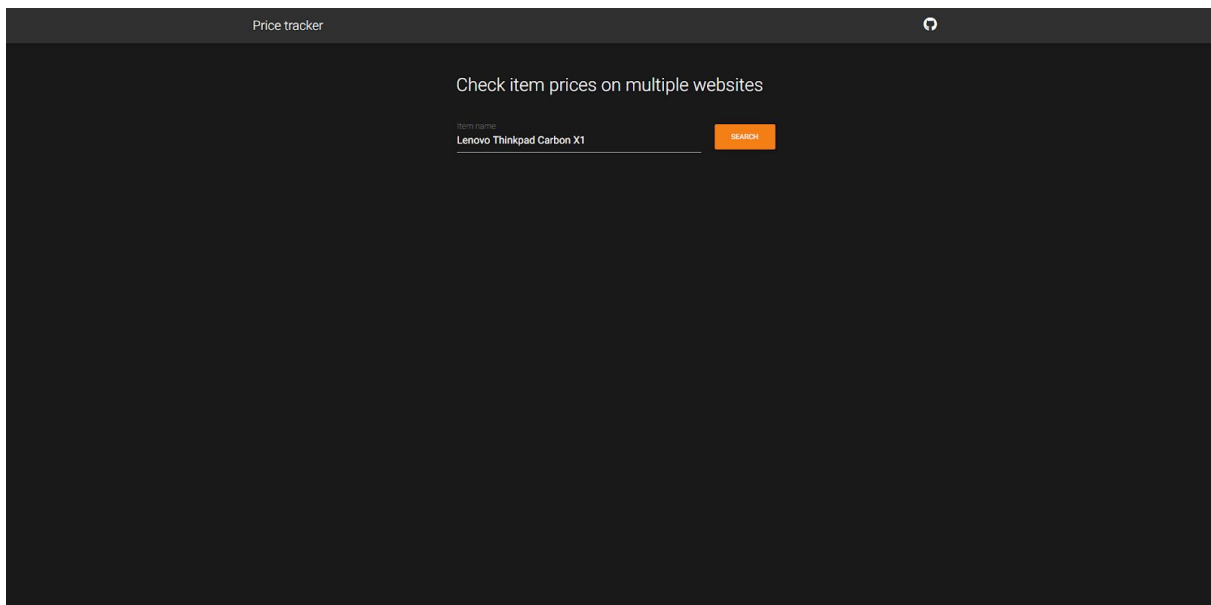
8.3. Działanie aplikacji.

Po wykonaniu punktu 8.1 na porcie 4200 uruchomiona zostaje aplikacja kliencka.



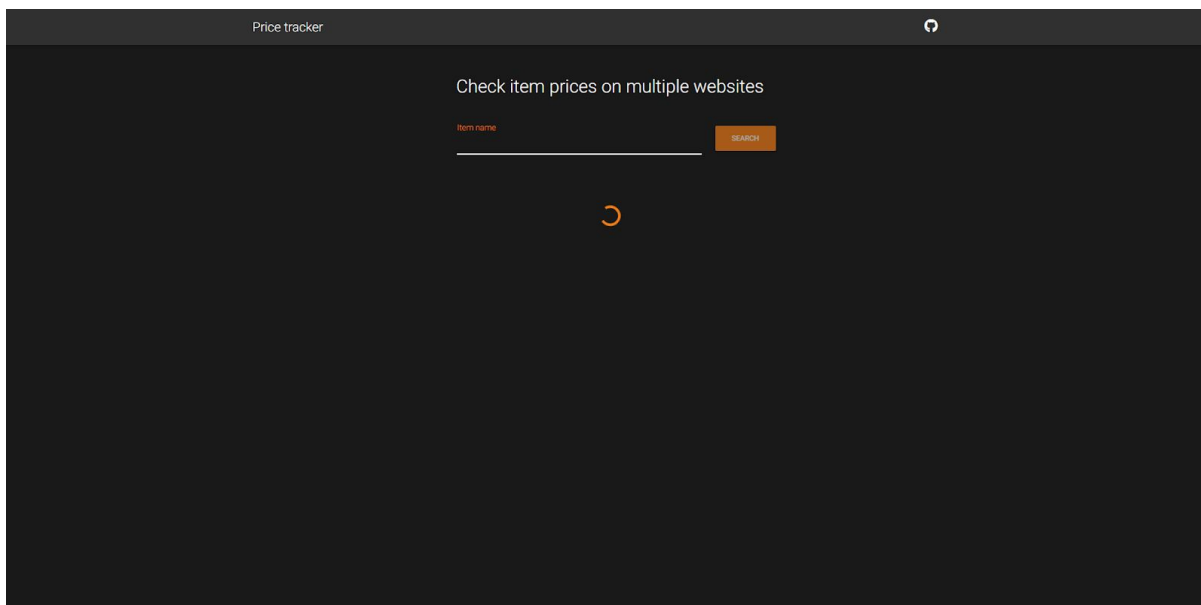
Zrzut 5. Główny widok aplikacji.

W polu wyszukiwania należy wprowadzić frazę przedmiotu, którego ceny chcemy sprawdzić. Po kliknięciu przycisku „search” zostanie przekazane żądanie do serwera.



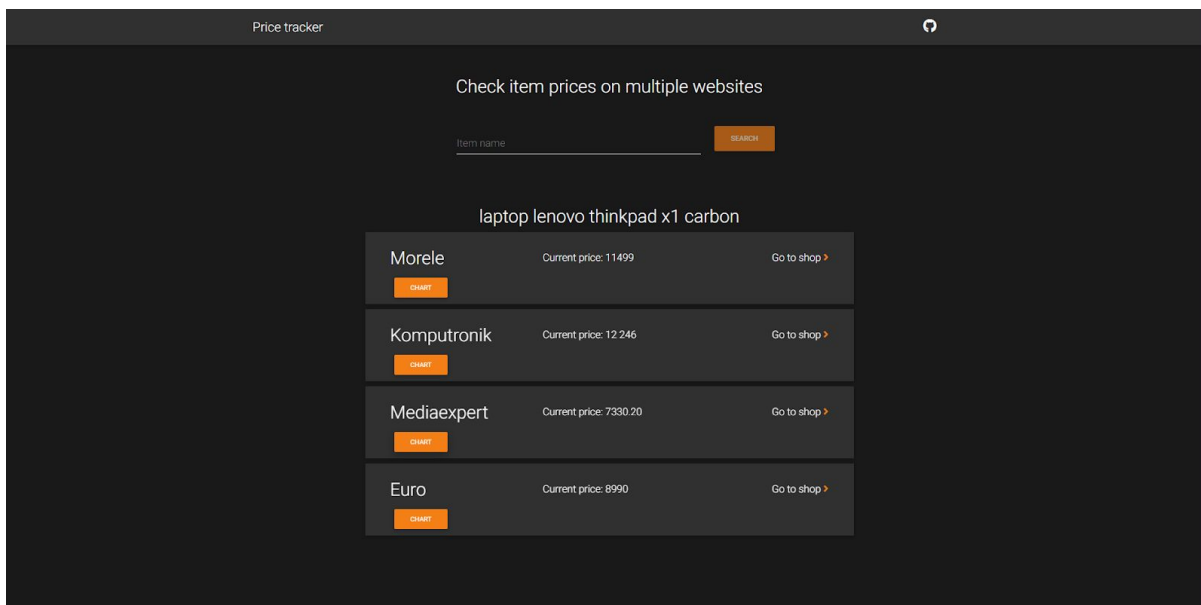
Zrzut 6. Główny widok aplikacji - wprowadzenie frazy.

Serwer przetwarza otrzymaną frazę. Jeśli w bazie danych znajduje się obiekt o podobnej nazwie to sprawdzane jest czy posiada on wszystkie niezbędne adresy Url sklepów. W przypadku, gdy brakuje adresów url, za pomocą Selenium, adresy są pozyskiwane z odpowiednich stron oraz uzupełniane w bazie danych. Następnie odpowiednie informacje są „scrapowane” z wykorzystaniem biblioteki **Beautiful Soup**. Jeśli adresy są już dostępne w bazie danych, etap „scrapowania” jest pomijany co znacznie skraca czas odpowiedzi serwera. Odpowiednie dane zostają zapisane w bazie danych oraz przesłane do klienta. W trakcie przetwarzania żądania wyświetlana jest animacja wczytywania.

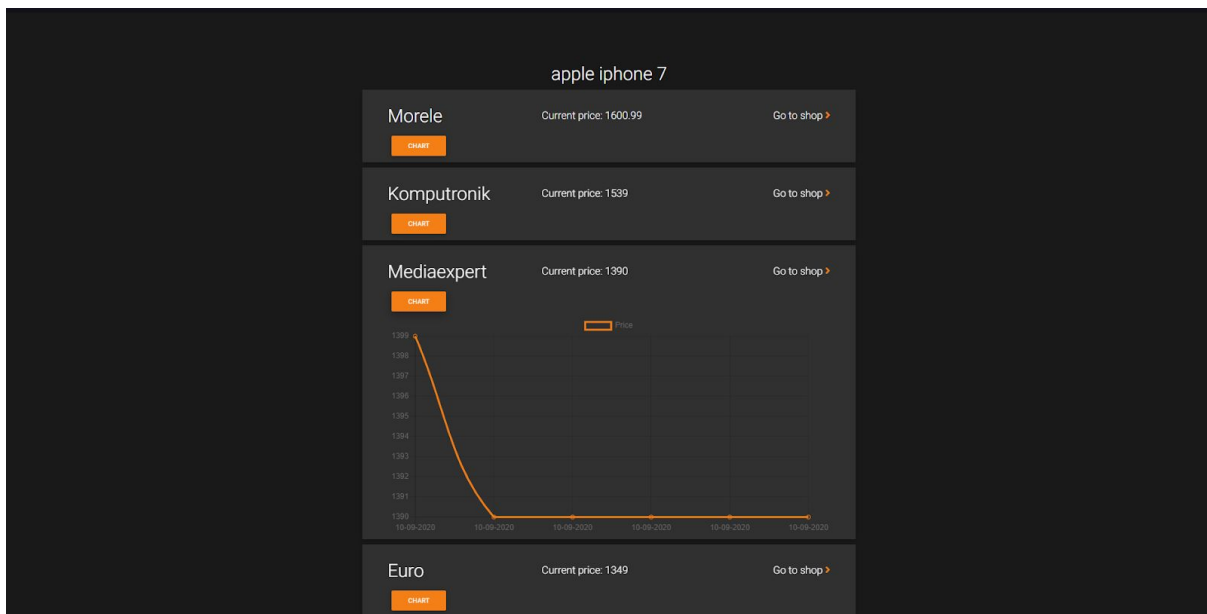


Zrzut 7. Główny widok aplikacji - animacja ładowania.

Po przetworzeniu żądania serwer przesyła dane do aplikacji klienckiej zawierające nazwę produktu, obecną cenę oraz historię zmian cen w postaci wykresu liniowego. Aplikacja kliencka udostępnia również możliwość odwiedzenia strony sklepu.

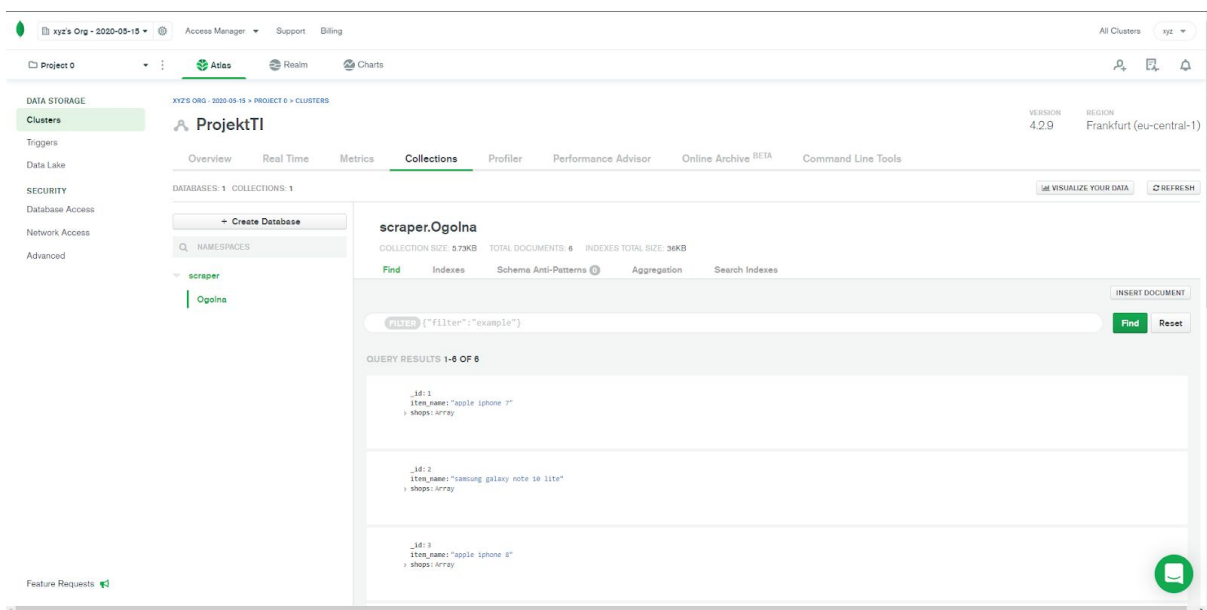


Zrzut 8. Wyświetlenie danych.



Zrzut 8. Wyświetlenie danych - opcja wyświetlenia wykresu.

Aktualne dane pobierane są na bieżąco z bazy danych



Zrzut 9. Widok rekordów w bazie danych.

```

_id: 1
item_name: "apple iphone 7"
shops: Array
  0: Object
    shop_name: "morele"
    url: "https://www.morele.net/smartfon-apple-iphone-7-32-gb-czarny-mn8x2pm-a-..."
    data: Array
      0: Object
      1: Object
        price: 1600.99
        date: "10-09-2020"
      2: Object
        price: 1600.99
        date: "10-09-2020"
      3: Object
        price: 1600.99
        date: "10-09-2020"
      4: Object
        price: 1600.99
        date: "10-09-2020"
      5: Object
        price: 1600.99
        date: "10-09-2020"
    1: Object
    2: Object
    3: Object
    shop_name: "euro"
    url: "http://www.euro.com.pl/telefon-y-komorkowe/apple-iphone-7-32gb-zloty.bh-..."
    data: Array
      0: Object
        price: 1349
        date: "10-09-2020"
      1: Object
        price: 1349
        date: "10-09-2020"
      2: Object
        price: 1349
        date: "10-09-2020"
      3: Object
        price: 1349
        date: "10-09-2020"
      4: Object
        price: 1349
        date: "10-09-2020"
      5: Object
        price: 1349
        date: "10-09-2020"

```

Zrzut 10. Szczegółowy widok rekordów w bazie danych wraz z atrybutami.