SPRAWOZDANIE z przedmiotu
„Projekt - zespołowe przedsięwzięcie programistyczne”

Data wykonania projektu:	**01.02.24 - 25.03.24**
Rok studiów:	**III**
Semestr:	***6.***
Grupa studencka:	**1b**

Projekt nr **1**

Temat: **Wyszukiwarka Nokaut**

Autorzy:
Kubica Patrycja
Kmiecik Kamila
Waluszek Dawid
Walaszczyk Oskar

1. Skład zespołu

1.1. Developer – Kamila Kmiecik, Dawid Waluszek
1.2. Tester – Oskar Walaszczyk
1.3. Ux Designer – Patrycja Kubica
1.4. Szef – Dawid Waluszek

2. Definicja wymagań funkcjonalnych

2.1. Historyjki
2.1.1. Jako użytkownik, chcę, aby aplikacja automatycznie wyszukiwała dla wskazanej, nazwę i cenę produktu z platformy Nokaut, aby łatwo uzyskać te informacje bez konieczności ręcznego przeszukiwania strony.
2.1.2. Jako użytkownik, chcę, aby aplikacja cyklicznie aktualizowała wyniki, aby informacje były zawsze aktualne i dostępne.

2.2. Opis działania

Aplikacja działa jako scraper, który automatycznie przeszukuje platformę Nokaut w celu znalezienia pierwszego linku, nazwy i ceny produktu (dodano również obraz). Aplikacja jest hostowana na serwerze i jest zaprogramowana tak, aby cyklicznie aktualizować wyniki, zapewniając w ten sposób najnowsze informacje.

3. Wymagania niefunkcjonalne

Wymaganie	**Opis**
Nieprzerwana dostępność	Aplikacja musi być stale dostępna, aby użytkownicy mogli uzyskać informacje o produkcie w dowolnym momencie.
Dokładność	Aplikacja musi zapewniać dokładne informacje o produkcie – co najmniej link, nazwa i cena.
Responsywność	Interfejs użytkownika powinien być responsywny, co oznacza, że powinien dobrze wyglądać i działać na różnych rozdzielczościach ekranu i rozmiarach urządzeń
Użyteczność	Aplikacja powinna być łatwa w użyciu, z intuicyjnym interfejsem użytkownika, który pozwala na szybkie i łatwe znalezienie i wykorzystanie potrzebnych funkcji.

4. Wybór technologii realizacji

4.1. Biblioteki:
C#: HtmlAgillityPack, Xunit
Node.js: Axios, Express, MsSQL, Node-Cron
4.2. Frameworki - Bootstrap
4.3. API – aplikacja WebAPI w .NET7

4.4. Platforma hostująca (rodzaj aplikacji – desktop, web, mobilna) – Aplikacja desktopowa

4.5. Bazy danych – relacyjna baza MSSQL

5. Wybór narzędzi realizacji

Narzędzie	Wykorzystanie
Visual Studio Code	Projekt interfejsu aplikacji Hosting serwera
Visual Studio	Projekt backendu aplikacji Testowanie aplikacji
Microsoft SQL Management Studio	Utworzenie bazy danych
Git, Github	Kontrola wersji oraz zdalnego dostępu do solucji

6. Repozytorium Kodu (bitbucket, github, gitlab ....) opis zastosowania

Repozytorium na platformie GitHub zostało utworzone dla całego zespołu projektowego pracującego nad wyszukiwarką produktów. Każdy członek zespołu ma obowiązek commitowania swoich wykonanych zadań oraz zmian do repozytorium z czytelną treścią. Poza branchem **main** wydzielono osobny branch **ConnectingFrontend**, w którym developer połączyć projekt frontendu z działającą aplikacją. Branche zostały zmergowane po wykonaniu zadania.

Link do repozytorium

7. Narzędzia do zarządzania projektem

W początkowej fazie pracy nad projektem, przetestowano platformę Jira jako narzędzie do zarządzania projektem. Zdecydowano się na to rozwiązanie, mając nadzieję na kompleksowe wsparcie w śledzeniu postępu prac, zarządzaniu zadaniami oraz komunikacji w zespole.

Niestety, mimo początkowego optymizmu, ostatecznie uznano, że darmowe rozwiązanie dostarczane przez Jirę nie spełnia oczekiwań i wymagań – problem z nadawaniem uprawnień do administrowania projektem. W trakcie testowania zauważono kilka ograniczeń, które utrudniały skuteczne zarządzanie projektem oraz komunikację w zespole. Brak niektórych funkcjonalności, niewystarczające wsparcie dla potrzeb oraz ograniczenia w darmowej wersji były głównymi czynnikami, które skłoniły do podjęcia decyzji o porzuceniu tego rozwiązania.

8. Narzędzia do komunikacji

Wykonawcy projektu komunikowali się poprzez stworzony przez szefa projektu serwer na platformie Discord. Rozwiązania problemów dotyczących tylko części składu zespołu (np. problemy dotyczące tylko developerów) były przedyskutowane poprzez osobisty czas lub podczas Code Review.

9. UML, UseCasy

9.1. Use Case 1: Pobieranie informacji o produkcie

Aktor główny: Aplikacja

Cel: Pobranie pierwszego linku, nazwy i ceny produktu z platformy Nokaut.

Scenariusz:

1. Aplikacja uruchamia scraper.
2. Scraper przeszukuje platformę Nokaut.
3. Scraper znajduje i pobiera pierwszy link, nazwę i cenę produktu.
4. Informacje są zapisywane i aktualizowane w aplikacji.

9.2. Use Case 2: Cykliczne aktualizowanie wyników

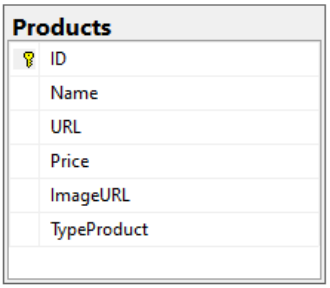
Aktor główny: Aplikacja

Cel: Regularne aktualizowanie informacji o produkcie.

Scenariusz:

1. Aplikacja jest ustawiona na cykliczne uruchamianie scrapera.
2. W określonych odstępach czasu, scraper jest uruchamiany.
3. Scraper pobiera najnowsze informacje o produkcie.
4. Informacje są aktualizowane w aplikacji.

10. Schemat bazy danych



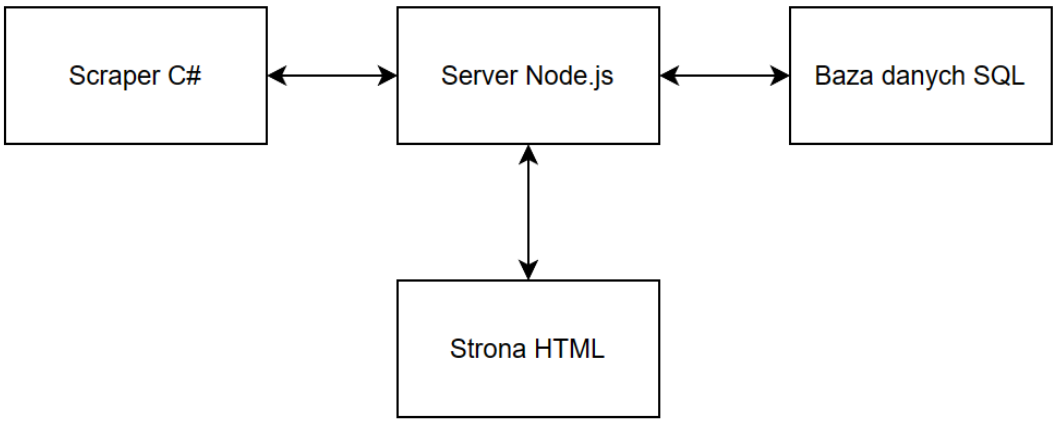
W aplikacji wykorzystano prostą bazę danych z jedną tabelą.

11. Schemat class

```
8 references
public class Product
{
    1 reference
    public int ID { get; set; }

    1 reference
    public string Name { get; set; }

    1 reference
    public string URL { get; set; }

    1 reference
    public string imageUrl { get; set; }

    1 reference
    public PriceCy Price { get; set; }

    1 reference
    public string typeProduct { get; set; }

    2 references
    public Product(int _id, string _name, string _url, decimal _priceValue, string _imageUrl, string _typeProduct)
    {
        ID = _id;
        Name = _name;
        URL = _url;
        Price = new PriceCy(_priceValue);
        imageUrl = _imageUrl;
        typeProduct = _typeProduct;
    }

    0 references
    public Product()...
}
```

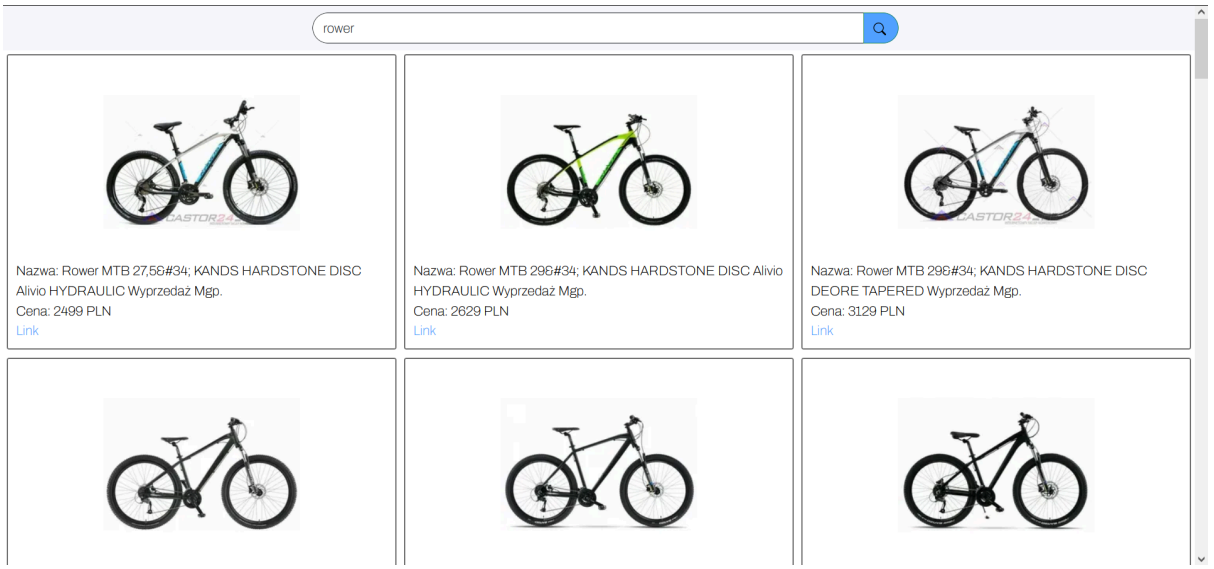
Klasa Product jest reprezentacją produktu w systemie, przechowującą jego kluczowe informacje, takie jak identyfikator, nazwa, adres URL, cena, adres URL obrazka oraz typ. Posiada konstruktory umożliwiające tworzenie obiektów klasy z określonymi wartościami, co ułatwia zarządzanie produktami w aplikacji. Dzięki polimorfizmowi, możliwe jest dynamiczne dostosowywanie pól i metod klasy. Całość zapewnia spójną i przejrzystą reprezentację produktów w systemie, ułatwiając zarządzanie nimi oraz umożliwiając szybkie i efektywne przetwarzanie danych z nimi związanych.

12. Schemat architektury



13. Projekt interfejsu

Zakładany interfejs strony internetowej.



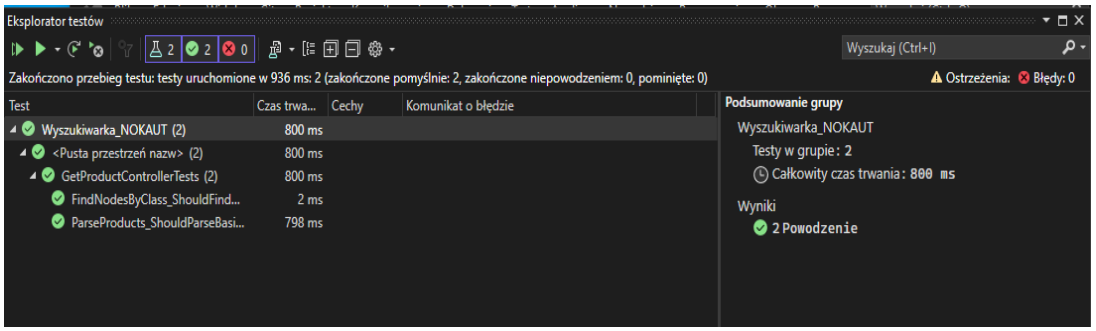
Rzeczywisty wygląd interfejsu.

14. Testy

14.1. Jednostkowe

Przeprowadzone zostały testy jednostkowe dla naszego scrapera w technologii .NET. Nasze testy sprawdzają poprawność metod w kontrolerze “GetProductsControler”. Test nr 1 sprawdza metodę “FindNodesByClass” oraz sprawdza, czy metoda ta poprawnie znajduje węzły z klasą Title.
Test nr 2 sprawdza metodę “ParseProducts” i sprawdza czy metoda ta poprawnie parsuje informacje o wybranym produkcie.

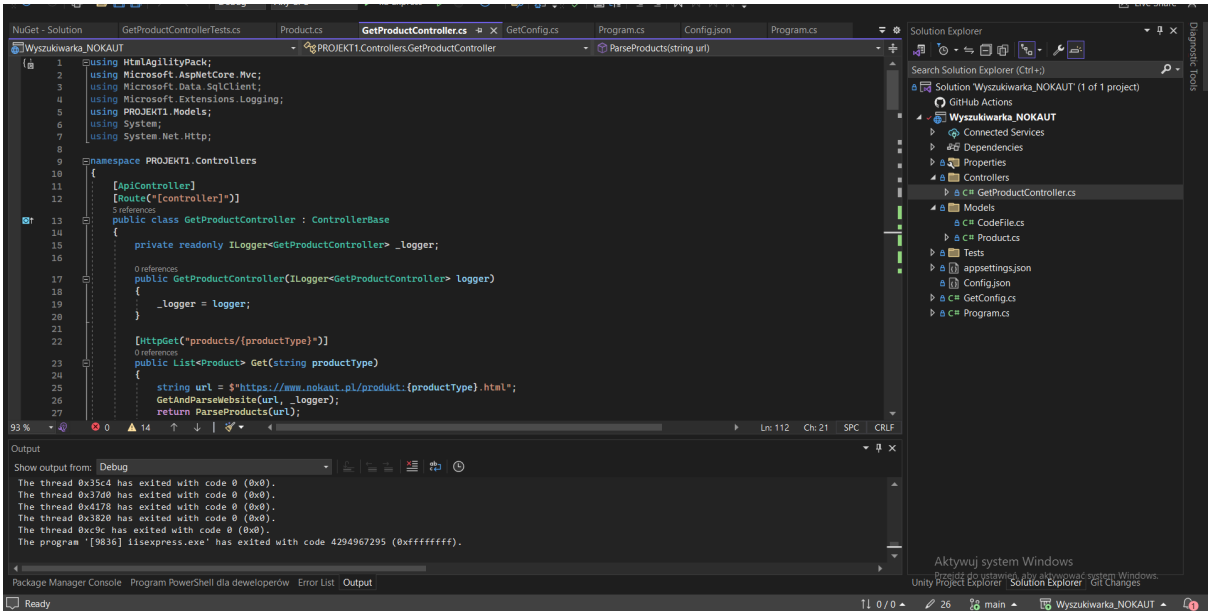
Przeprowadzone testy wykazują poprawność naszego kodu.



14.2. Funkcjonalne -

14.3. TDD - Aplikacja nie była tworzona jako Test-driven. Najpierw tworzono funkcjonalność, a potem testy.

15. Prezentacja stworzonego systemu



```
C:\Users\Dawid\source\repos\Wyszukiwarka_NOKAUT\design_strony>node server.js
Server is listening at http://localhost:3000
Connected to MSSQL database.
Connected to MSSQL database.
```

Uruchomienie servera Node na localhost:3000

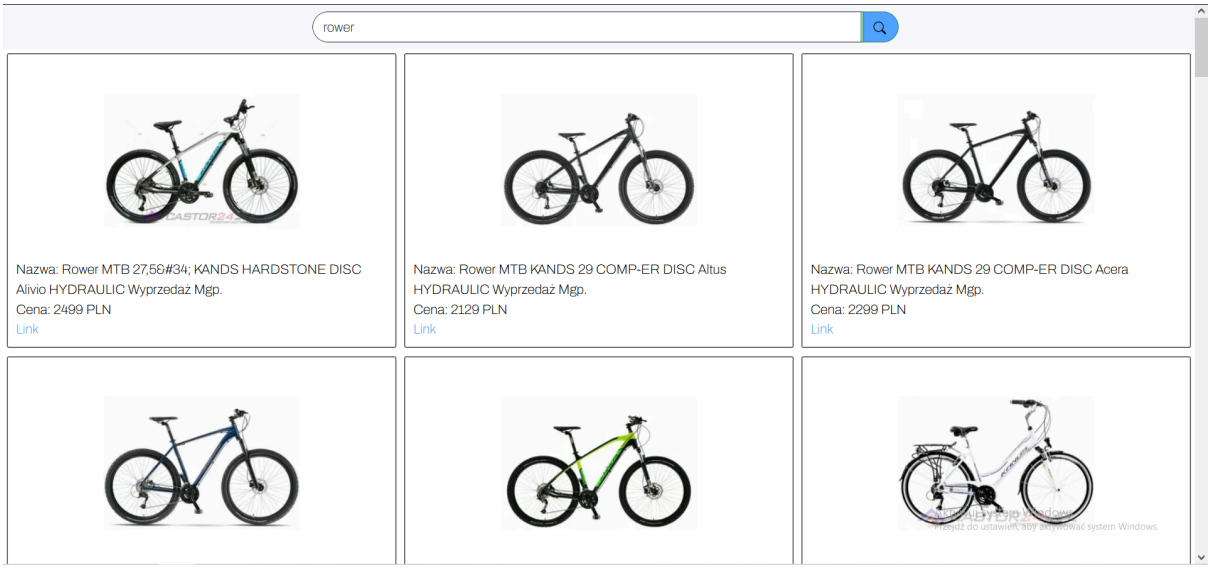
```
Scraping scheduled for product type: jogurt naturalny
Scraping scheduled for product type: papier
Scraping scheduled for product type: yerba
Scraping scheduled for product type: okno
Scraping scheduled for product type: tuba
Scraping scheduled for product type: moneta
Scraping scheduled for product type: biurko
Scraping completed for product type: długopis
Scraping completed for product type: kawa
Scraping completed for product type: ławka
Scraping completed for product type: teczka
Scraping completed for product type: kabel
Scraping completed for product type: piłka nożna
Scraping completed for product type: hak
Scraping completed for product type: kot
Scraping completed for product type: kula
Scraping completed for product type: yerba mate
```

Planowanie następnego cyklu scrapowania i jego wykonanie.

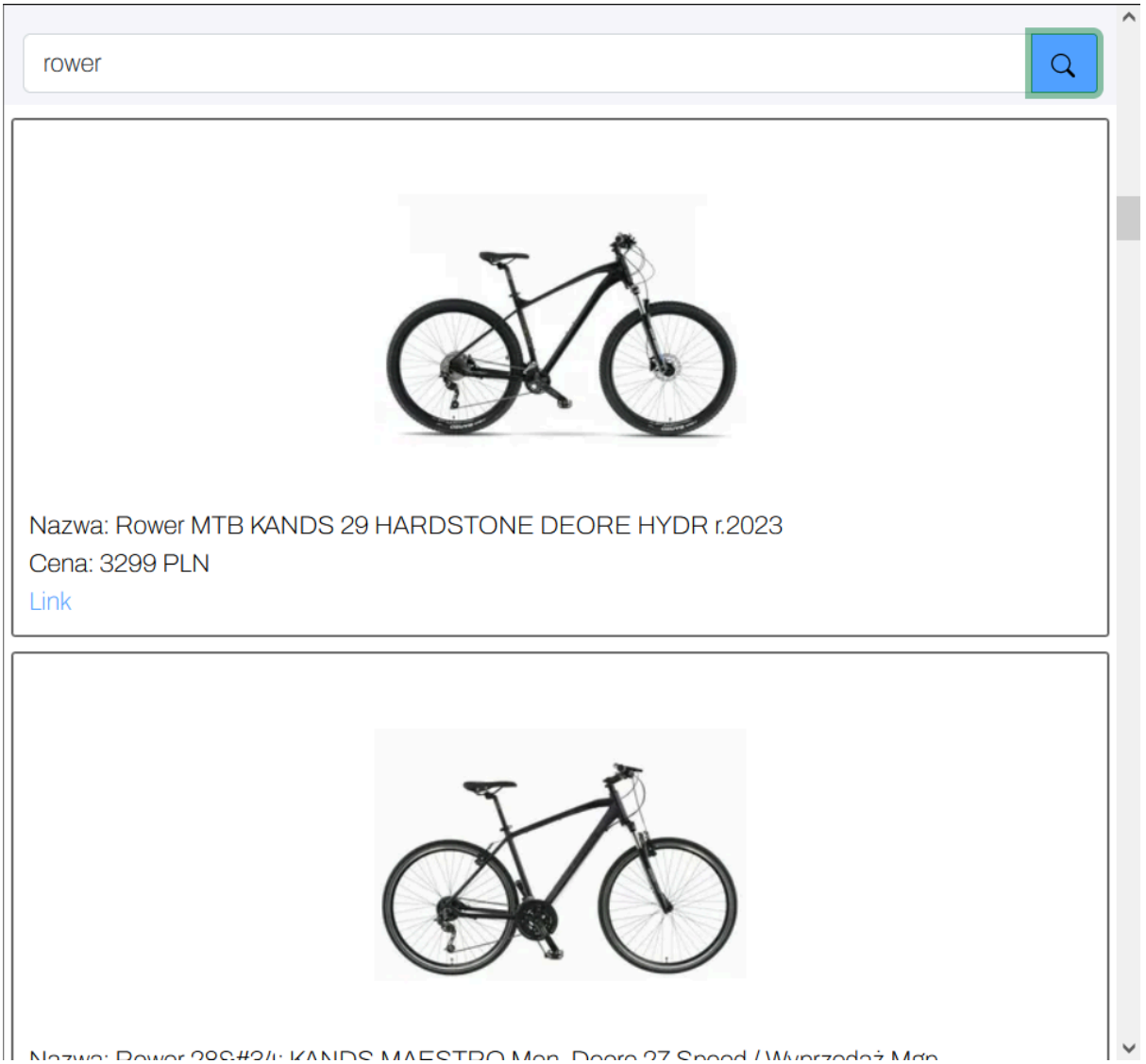
Autorzy

Patrycja Kubica Kamila Kmiecik Dawid Waluszek Oskar Walaszczyk

Strona główna po wejściu na stronę web localhost:3000/stronaglow.html



Wykonane wyszukiwanie produktu rower.



Responsywność strony.

16. Podsumowanie Wnioski

16.1. W ramach projektu konieczne było opracowanie scraper'a, z uwagi na brak ogólnodostępnego API dla platformy nokaut.pl.

16.2. HtmlAgilityPack zapewnia łatwe i wygodne parsowanie kodu HTML, umożliwiając dostęp do różnych elementów strony internetowej – linku, nazwy oraz linku url.

16.3. Wykorzystanie Xunit w C# i Node-Cron w Node.js umożliwiło nam automatyzację testów i sprawdzenia poprawności działania aplikacji.

16.4. Node.js oferuje bibliotekę mssql, która pozwala na operacje bazodanowe dla MsSQL.

16.5. Axios jest biblioteką Node.js pozwalającą na wysyłanie asynchronicznych zapytań http oraz ich zarządzaniem.

16.6. Express jest biblioteką Node.js, która pozwala na proste budowanie aplikacji webowych dla serwera.