

POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY

Informatyka, rok III



Adrian Leoniak, Konrad Ferbes

DOKUMENTACJA Z PRZEDMIOTU PODSTAWY TELEINFORMATYKI

Webscraper

Poznań, 2016

Spis treści

1. Wstęp.....	3
2. Opis projektu	5
2.1 Spider Nike	6
2.2 Spider Allegro	6
2.3 Aplikacja NikeSearch	7
3. Historia rozwoju aplikacji	7
3.1 Spider Nike.....	7
3.2 Spider Allegro	19
3.3 Aplikacja NikeSearch	21
3 Instrukcja obsługi aplikacji	23
4 Wykorzystane technologie	28
5 Harmonogram prac.....	29
6 Podsumowanie	30

1. Wstęp

Popularność wyszukiwarek internetowych rośnie z każdym rokiem. Stanowią one krwioobiegi współczesnego Internetu. Przeciętny użytkownik sieci zanim trafi na szukaną przez siebie treść niemal zawsze ucieka się do skorzystania z usług wyszukiwarek internetowych. Wiele osób korzysta z wyszukiwarek codziennie, nie posiadając żadnej wiedzy na temat algorytmów jakie definiują ich działanie. Bez nich poruszanie się w sieci byłoby wysoce niekomfortowe. Z drugiej strony, korzystając z wyszukiwarki, mamy zwykle do czynienia z problemem mnogości informacji. Wpisując w najpopularniejszą obecnie przeglądarkę Google (około 30 miliardów odwiedzin miesięcznie według serwisu similiarweb.com), na przykład, frazę „muzyka”, otrzymamy około 38 500 000 trafień. Żaden człowiek nie jest w stanie zgromadzić tych wyników i posortować ich według ważności. A więc to programy komputerowe są odpowiedzialne za układanie wyników w sensownej dla użytkownika kolejności.

Za pomocą specjalnych programów, tak zwanych web crawlerów, albo robotów indeksujących operatorzy wyszukiwarek (Bing, Google, Yahoo! i inni) przeszukują Internet. Każda strona ze wszystkimi słowami, zdjęciami i innymi elementami trafia do bazy danych wyszukiwarki. Indeks ten zawiera więc prawie wszystkie informacje z Internetu. Crawlery są potrzebne, żeby użytkownik mógł pozyskiwać przetworzone i wyselekcjonowane informacje według interesujących go kryteriów i zakwalifikować je według ich znaczenia. Im ważniejsza jest strona i im częściej jest aktualizowana, tym częściej powracają na nią wspomniane wcześniej roboty. To jednak nie jedyny cel, do jakiego można wykorzystać crawlery. W dzisiejszych czasach wiele innowacyjnych pomysłów w sieci powstaje w oparciu o przetwarzanie informacji pozyskanych z ogromnych baz danych.

Wydawać by się mogło, że ograniczenie możliwości eksploracji Internetu za pomocą robotów cofnęłoby jego rozwój. Status prawny czynności crawlowania jest jednak niejasny. Z jednej strony istnieją regulacje prawne, które ograniczają możliwość crawlowania, ale tylko w określonych przypadkach. Z drugiej strony firmy często próbują chronić swoje zasoby przed ich przeszukiwaniem za pomocą odpowiednich zapisów w regulaminach i posługiwania się plikami robots.txt.

Podstawową sprawą jest rozróżnienie samej czynności crawlowania (automatycznego przeszukiwania treści stron internetowych, indeksowania jej i pobierania) od ewentualnego późniejszego wykorzystywania pozyskanych w ten sposób treści.

Obecnie możliwość crawlowania na mocy przepisów prawa ograniczona jest w Unii Europejskiej w zasadzie tylko w odniesieniu do chronionych baz danych. W zależności od charakteru baz, różnie z prawnego punktu widzenia ocenia się crawlowanie ich zawartości.

Zdarza się jednak, że właściciel strony internetowej w regulaminie korzystania z serwisu zabrania crawlowania. W takim przypadku najczęściej posługuje plikiem robots.txt, to jest plikami zawierającymi instrukcje dla robotów indeksujących strony internetowe, ograniczającymi możliwość dokonywania takich działań. Nie jest to fizyczna blokada uniemożliwiająca automatyczne analizowanie zawartości strony, ale raczej informacja, że nie wolno tego robić. Wykrycie niestosowania się do tych zaleceń skutkować może permanentnym blokadą dostępu dla adresu IP używanego przez nasze urządzenie. Na rysunku 1 przedstawiona jest przykładowa zawartość pliku robots.txt dla witryny google.com:

```

User-agent: *
Disallow: /search
Allow: /search/about
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
Disallow: /?hl=*&
Allow: /?hl=*&gws_rd=ssl$
Disallow: /?hl=*&*&gws_rd=ssl
Allow: /?gws_rd=ssl$
Allow: /?pt1=true$
Disallow: /imgres
Disallow: /u/
Disallow: /preferences
Disallow: /setprefs
Disallow: /default
Disallow: /m?
Disallow: /m/
Allow: /m/finance
Disallow: /wml?
Disallow: /wml/?
Disallow: /wml/search?
Disallow: /xhtml?
Disallow: /xhtml/?
Disallow: /xhtml/search?
Disallow: /xml?
Disallow: /imode?
Disallow: /imode/?

```

Rysunek 1 Zawartość pliku robots.txt witryny google.com

Objaśnienia:

User-Agent: * - poniższe zasady obowiązują wszystkie pająki.

Disallow:/ - określa, do jakich ścieżek lub zasobów pająki mają zabroniony dostęp.

2. Opis projektu

Celem niniejszego projektu było stworzenie aplikacji pozwalającej na przeglądanie oferty butów dostępnych na stronie store.nike.com, jednocześnie wyszukując dostępne alternatywy w postaci aukcji w serwisie Allegro. Zadanie to polegało na utworzeniu w środowisku Scrapy web crawlerów dla obydwu wymienionych wcześniej serwisów, z uwzględnieniem struktur plików HTML reprezentujących strony oraz warunków crawlowania określonych w plikach robot.txt. Zindeksowane informacje zapisane zostały w formacie .json. Przeglądanie informacji uzyskanych przez utworzone pająki i wydawanie poleceń kolejnych crawlowań umożliwia zaimplementowana dodatkowo

2.1 Spider Nike

[illegible]

2.2 Spider Allegro

6

```
user-agent: *
disallow: /showcase.php/
disallow: /SendMailToFriend.php/
disallow: /SendMailToUser.php$
disallow: /myaccount/
disallow: /add_to_watchlist.php$
disallow: /report_item.php$
disallow: /NewItem/
disallow: /showitem/description/legacy/
sitemap: http://allegro.pl/sitemap/sitemap_index_pl.xml
```

Rysunek 3 Plik robots.txt adresu allegro

Do obsługi sitemap służy specjalna klasa pająka dostępna w Scrapy – SitemapSpider.

2. 3 Aplikacja NikeSearch

Aplikacja pozwala na przeglądanie informacji uzyskanych przez pająki i zgromadzonych w plikach JSON oraz wydawanie poleceń kolejnych crawlowań. Pozwala na wyszukanie modeli butów dostępnych na oficjalnej stronie sklepu Nike (www.store.nike.com) według ich nazwy, a po wybraniu konkretnego modelu dopasowanie do niego najbardziej pasujących aukcji na allegro. Dodatkowo, aplikacja pozwala na bezpośrednie przejście za pośrednictwem przeglądarki internetowej do strony zawierającej dane buty - zarówno do strony Nike, jak i powiązanej z nią aukcją na Allegro.

3. Historia rozwoju aplikacji

3.1 Spider Nike

Historia postępów w pracach nad modulem umożliwiającym web-scraping, czyli pozyskanie danych, w tym przypadku dotyczących produktów – butów udostępnianych na brytyjskiej wersji strony www.store.nike.com.

W miarę możliwości zespół starał się wykorzystywać się powłokę scrapy w celu przetestowania poprawności i prób dotarcia do pożądaných danych. Jednak ze względu na problemy związane z pisanem bardziej złożonych funkcji, kodu wykorzystującego pętle, próby były przeprowadzane w sposób następujący:

modyfikacja pająka, uruchomienie pająka i zapis danych do pliku json (używając komendy wiersza poleceń - scrapy crawl nazwapająka -o nazwapliku.json).

Początkowo w wyniku braku obeznania z framerowkiem, strukturą strony oraz formułowaniem odpowiednich selektorów ścieżek xpath zostały dokonane próby ekstrakcji jakichkolwiek danych.

Tak więc początkowo została wybrana zarówno zła ścieżka początkowa:

```
"http://store.nike.com/"
```

, jak i zakres dozwolonych ścieżek:

```
"http://store.nike.com/"
```

, a następnie korzystając z funkcji (próbując się dostać do struktur zapisanych w formie znaczników html) :

```
def parse(self, response):
    self.logger.info('Uzyskano odpowiedz od serwera %s.',
response.url)
    for sel in response.xpath('//ul/li'):
        item = NikeItem()
        item['name'] = sel.xpath('normalize-
space(a/text())').extract()
        item['link'] = sel.xpath('normalize-
space(a/@href)').extract()
        item['description'] = sel.xpath('normalize-
space(a/@data-subnav-label)').extract()
        yield item
```

uzyskując następujące rezultaty:

```
{"link": ["http://www.nike.com/us/en_us/"], "name": ["United
States"], "description": [""]},
{"link": ["http://www.nike.com/xl/es_la/"], "name":
["Am\u00e9rica Latina"], "description": [""]},
{"link": [""], "name": [""], "description": [""]},
```



```
{"link": ["http://www.nike.com/ca/en_gb/"], "name":  
["ENGLISH"], "description": [""]},
```

Zespół starał się odnaleźć we frameworku. Jako główny cel zostało obrane otrzymanie danych dotyczących adresów poszczególnych przedmiotów.

Kolejne próby przybliżyły zespół do rezultatu. Wprowadzono pomniejszy cel, a mianowicie postarano się zdobyć listę kategorii przedmiotów i ich adresów z następującym skutkiem:

```
{"link": ["http://store.nike.com/pl/pl_pl/pw/kobiety-bieganie-  
buty/7ptZ8yzZoi3"], "name": ["\r\n  
Bieganie\r\n",  
"description": ["\r\n",  
"\r\n"],  
{"link": ["http://store.nike.com/pl/pl_pl/pw/kobiety-  
%C4%87wiczenia-i-trening-buty/7ptZ9hkZoi3"], "name": ["\r\n  
\u0106wiczenia i trening\r\n",  
"\r\n"],  
{"link": ["http://store.nike.com/pl/pl_pl/pw/kobiety-tenis-  
buty/7ptZ8r0Zoi3"], "name": ["\r\n  
Tenis\r\n",  
"description": ["\r\n",  
"\r\n"],  
{"link": ["http://store.nike.com/pl/pl_pl/pw/kobiety-golf-  
buty/7ptZahaZoi3"], "name": ["\r\n  
Golf\r\n",  
"description": ["\r\n",  
"\r\n"]},
```

Problemem, prócz pobierania niewłaściwych danych stały się znaki formatujące tekst wynikające ze ścieżek selektora xpath w postaci a/text(). Problem został rozwiązany dzięki wprowadzeniu do formuły funkcji normalize-space.

```
{
  "link": ["http://store.nike.com/gb/en_gb/pw/mens-lifestyle-shoes/7puZ8yzZoi3"],
  "name": ["Lifestyle"],
  "description": ["Lifestyle"]
},
{
  "link": ["http://store.nike.com/gb/en_gb/pw/mens-running-shoes/7puZ8yzZoi3"],
  "name": ["Running"],
  "description": ["Running"]
},
{
  "link": ["http://store.nike.com/gb/en_gb/pw/mens-football-shoes/7puZ896Zoi3"],
  "name": ["Football"],
  "description": ["Football"]
},
{
  "link": ["http://store.nike.com/gb/en_gb/pw/mens-basketball-shoes/7puZ8r1Zoi3"],
  "name": ["Basketball"],
  "description": ["Basketball"]
},
```

Po wykonaniu pomniejszego celu umożliwiające początkowe zaznajomienie się z ekstrakcją danych zespół zwrócił uwagę na strukturę strony html, na której znajdowały się wszystkie produkty z kategorii obuwie. Zmieniono więc ścieżkę startową oraz dozwolne adresy na

```
"http://store.nike.com/gb/en_gb/pw/mens-shoes/7puZoi3"
```

Początkowo uzyskano dostęp do całej struktury gridwall:

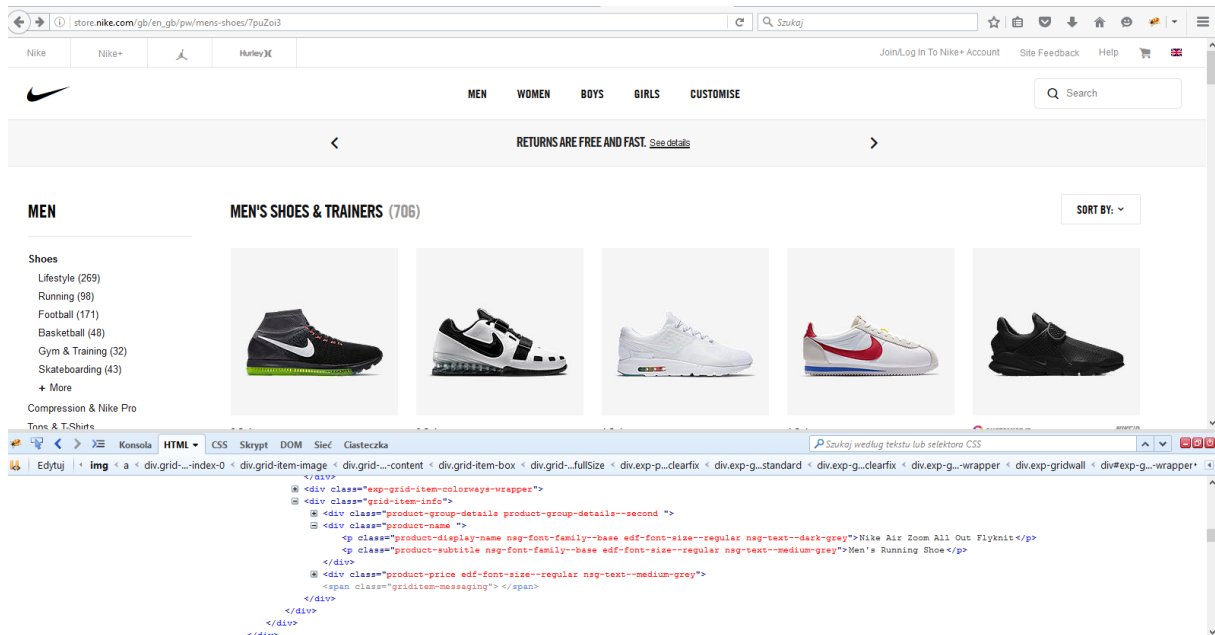
```

NukeSpider.py x gridwallDatajson x
sections products spriteSheet
[{"nextPageDataService":"http://store.nike.com/html-services/gridwallData?country=GB&lang_locale=en_GB&gridwallPath=meng-shoes/7puZoI3:pn=4",
  "seoBlurb":"<div class=device-type-desktop nike-cq-old-template nike-cq-pes-holder-page nike-cq-lang-en nike-cq-en GB nike-cq-tesla.xmlml" data-experience-id="one-ni
  "banners":["{"url":"http://origin-content.nike.com/content/nike/en_GB/one-nike/store/pes-holders/notifications/we_notification_freedelivery.partial-request.xml/variant/GB
  [{"text":"text","url":"http://store.nike.com/gb/en/gb/pw/meng-shoes/7puZoI3","sortField":"","sortDirection":null,"selected":true},"{"text":"Newest","url":"http://store.nike.com
  "spriteSheet":"http://images.nike.com/is/image/DotCom/pwp_sheet26NIKE_FWP36:6img0=749771_102","fullSize":true,"numberOfColors":1,"showNumberOfColors":true,"itemIndex":1
  "pdpUrl":"http://store.nike.com/gb/en/gb/product/424141662?cid=42159&pid=4241416627&pid=424141662",
  "spriteSheet":"http://ugc.nikeid.com/is/image/nike/pwp_sheet_legacy7&NIKE_FWP36:6img0=ugc/424141662.tif",
  "fullSize":false,"numberOfColors":1,"showNumberOfColors":true,"itemIndex":126,"colorways":null,"showColorways":false,
  "showReviews":true,"showPrice":true,"localPrice":"&spound=180","overridesLocalPrice":null,"employeePrice":"&spound=108",
  "overridesEmployeePrice":"&spound=180","ratingCount":null,"rating":null,"teamName":null,"maxAge":null,"textBadge":null,
  "gridType":"standard","segmentNumber":"","customizationType":"","customizable":true},"{"title":"Nike Hypervegnom Phatal II SG-PRO",
  "subtitle":"Men's Soft-Ground Football Boots","nikeid":false,"nikeidPremiumImage":false,"nfl":false,"jersey":false,
  "playerNamesAvailable":false,"widthsAvailable":false,"inStock":true,"comingSoon":false,"preorder":false,
  "pdpUrl":"http://store.nike.com/gb/en/gb/pd/hypervengom-phatal-ii-sg-pro-football-boots/pid-10949108/pgid-11648852",
  "spriteSheet":"http://images.nike.com/is/image/DotCom/pwp_sheet23NIKE_FWP36:6img0=747491_108",
  "fullSize":true,"numberOfColors":1,"showNumberOfColors":true,"itemIndex":129,"colorways":null,"showColorways":false,"showReviews":true,"showPrice":true,
  "localPrice":"&spound=145","overridesLocalPrice":null,"employeePrice":"&spound=87","overridesEmployeePrice":"&spound=145","ratingCount":null,
  "teamName":null,"maxAge":null,"textBadge":null,"gridType":"standard","segmentNumber":"","customizationType":"","customizable":false},
  "title":"Nike Zoom Kobe Icon Jacquard Premium","subtitle":"Men's Shoe","nikeid":false,"nikeidPremiumImage":false,"nfl":false,"jersey":false,
  "playerNamesAvailable":false,"widthsAvailable":false,"inStock":true,"comingSoon":false,"preorder":false,
  "pdpUrl":"http://store.nike.com/gb/en/gb/pd/zoom-kobe-icon-jacquard-shoe/pid-10949315/pgid-11262676",
  "spriteSheet":"http://images.nike.com/is/image/DotCom/pwp_sheet27NIKE_FWP36:6img0=832836_001","fullSize":true,"numberOfColors":1,
  "showNumberOfColors":true,"itemIndex":130,"colorways":null,"showColorways":false,"showReviews":true,"showPrice":true,
  "localPrice":"&spound=150","overridesLocalPrice":null,"employeePrice":"&spound=90","overridesEmployeePrice":"&spound=150",
  "ratingCount":null,"rating":null,"teamName":null,"maxAge":null,"textBadge":null,"gridType":"standard","segmentNumber":"","customizationType":"","customizable":false},"{"title":"Kobe XI Elite",
  "subtitle":"Men's Basketball Shoe","nikeid":false,

```

Rysunek 4- Zawartość pliku gridwallData.json

Zespół jednak następnie skupił uwagę na bardziej konkretnej klasie struktury html przechowującą informację o konkretnym obiekcie.



Rysunek 5- Struktura HTML ze szczególnym uwzględnieniem pól związanych z obiektem przedstawiona w dodatku Firebug.

W tym celu korzystano próbowano użyć takich struktur pająka jak

```
divs = response.xpath('//div')
for p in divs.xpath('./p'):
    print p.extract() /
```

, a następnie

```
produkty = response.xpath('./div[@class="grid-item-info"]/p')
print produkty

for produkt in produkty:
    item = obiekt()
    item['name'] = produkt.xpath('normalize-space(a[@class =
"./product-name/text()])').extract()
    item['price'] = produkt.xpath('normalize-space(a[@class =
"./prices/text()])').extract()
    yield item
```

Niepowodzenia w przypadku prób dostania się do informacji, używając powyższych funkcji, skłoniły zespół do zmiany podejścia. Zdecydowano się pobierać informację o adresie strony prezentującej konkretny przedmiot ze sklepu, a następnie pobranie danych takich jak nazwa oraz cena.

W tym celu skorzystano z powłoki scrapy i testowano między innymi następujący selektor:

```
response.xpath('normalize-space(//div[@class="grid-item-image-wrapper sprite-sheet sprite-index-0"]/a/@href)')
```

Liczne próby sformułowania odpowiedniego selektora nie przynosiły pożądanych rezultatów w wynikach prezentowanych przez powłokę. Spróbowano więc utworzyć bardziej złożoną strukturę już wewnątrz pająka.

```
for href in response.xpath('..//div[@class="grid-item-image-wrapper sprite-sheet sprite-index-0"]'):
    links = href.xpath('normalize-space(a/@href)').extract()
    self.logger.info
    for link in links:
        print link
```

W przypadku użycia struktury oraz zamiast wypisania “link” zastosowano yield link (oraz niepotrzebnie nadmiarowo operacji yield podlegało również pole name obiektu) oraz rezultat działania pająka zapisano do pliku json, otrzymano następujący rezultat

```
{"link": ["http://store.nike.com/gb/en_gb/pd/pro-core-compression-shirt/pid-709273/pgid-10257600"], "name": []},
{"link": ["http://store.nike.com/gb/en_gb/pd/flyknit-lunar-3-running-shoe/pid-10191260/pgid-11261785"], "name": []},
{"link": ["http://store.nike.com/gb/en_gb/product/flyknit-lunar-3-id/?piid=39456&pbid=966090970#?pbid=966090970"], "name": []},
{"link": ["http://store.nike.com/gb/en_gb/product/flyknit-
```

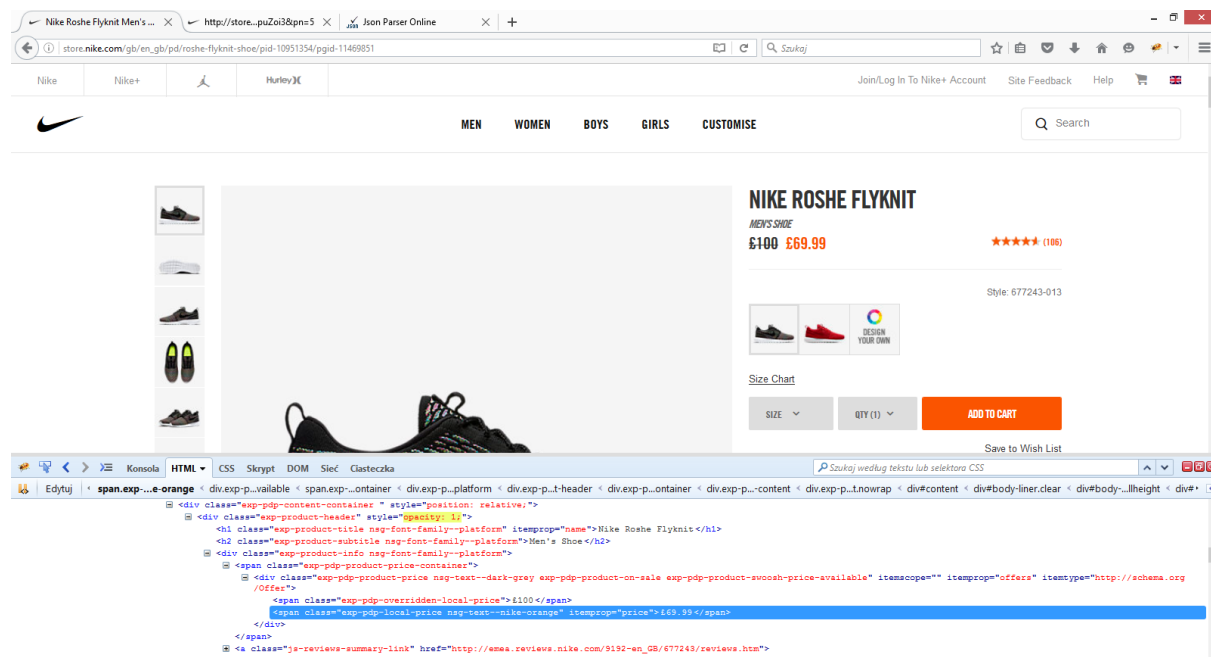
```
lunar-3-id/?piid=39455&pbid=151042551#?pbid=151042551"],
"name": [""],
},
```

Zespół mając dostęp do danych adresów, musiał nauczyć się budowania pająka w taki sposób, by podążał on za podanymi adresami, a następnie pobierał z nich dane. Znamiona takiego działania sprawiają, iż do funkcji pająka pomijając scraping dołączył crawling. Użyto w tym celu następującego podejścia. W przypadku wywołania pierwszej funkcji pająka (domyślnej) wykonany był kod przedstawiony powyżej. Jednak zamiast yieldu linków jako obiektów były one przekazywane jako argument wbudowanej funkcji frameworka scrapy.Request w postaci:

```
yield scrapy.Request(link, callback=self.parse_dir_contents)
```

Oznaczało to, iż w przypadku odpowiedzi zwrotnej na żądanie wysłane na dany adres pająk wywoływał funkcję parse_dir_contents, której celem było wyeksportowanie pożądaných danych o pojedynczym produkcie.

Strona ukazująca konkretny produkt prezentuje się następująco:



Kolejne próby uzyskania danych przynosiły coraz to lepsze rezultaty począwszy od :

```
{
  "price": ["\u00a328"],
  "name": ["Nike Pro Core - Compression"]},
  {"price": [""], "name": [""]},
  {"price": [""], "name": [""]},
  {"price": [""], "name": [""]},
  {"price": [""], "name": [""]},
  {"price": [""], "name": [""]},
  {"price": [""], "name": [""]},
  {"price": ["\u00a390"], "name": ["Nike Air Huarache"]},
  {"price": [""], "name": [""]}
}
```

,poprzez :

```
{
  "price": "\u00a370",
  "name": "Nike Cortez Basic 1972 QS"},
  {"price": "\u00a3120",
  "name": "Nike Air Force 1 Flyknit Low"},
  {"price": "\u00a3110",
  "name": "Nike Metcon 2"},
  {"price": "\u00a3130",
  "name": "Nike Free Train Force Flyknit"},
  {"price": "\u00a390",
  "name": "Nike SB Stefan Janoski Max"},
  {"price": "\u00a3110",
  "name": "Nike Air Max 90 Ultra BR"},
  {"price": "\u00a3110",
  "name": "Nike Air Huarache Ultra Breathe"},
}
```

, aż finalnie do uzyskania:

```
{
  "price": "\u00a3125",
  "link": "http://store.nike.com/gb/en_gb/pd/free-rn-motion-flyknit-running-shoe/pid-10952618/pgid-11266652",
  "name": "Nike Free RN Motion Flyknit"},
}
```

Uwagę zwrócił specyficzny tekst pojawiający się w polu “price”. Okazało się, iż pajak wraz z Pythonem danej wersji uświadczył problemów w interpretacji ASCII znaku

funta. Zdecydowano się zostawić dany format, który w późniejszej fazie projektu, aplikacji był odpowiednio parsowany i interpretowany jako funt.

W celu uzyskania powyższych rezultatów utworzono następująca konstrukcję definiującą obsługę konkretnego obiektu (czyli funkcję wywoływaną przez `scrapy.Request`)

```
def parse_dir_contents(self, response):
    #self.logger.info('Obiekt ze strony: ')
    #print response.url
    item = ItemApp() #NikeShoe
    names = response.xpath('normalize-
space(//h1/text())').extract()
    prices = response.xpath('normalize-
space(//span[@itemprop="price"]/text())').extract()
    #images = response.xpath('..//img[@class="hero-image-
container"] ')
    #images = response.xpath('normalize-space(//img//data
for name in names:
    for price in prices:
        name = name.lstrip('u\\')
        #price = price.lstrip('u\\'\\xa')

        if name and price:
            item['title'] = name
            item['price'] = price
            item['url'] = response.url
            item['image'] =
response.css('img').xpath('normalize-space(@data-src-
small)')[3].extract()
            yield item
```

Dodatkowo, w celu usunięcia niepożądanych znaków z pola name skorzystano ze wbudowanej funkcji języka Python `.lstrip`. Nazwy obiektów otrzymano ekstraktując

dane tekstowe ze znaczników nagłówków html, a ceny wyszukując klas span, których atrybut o nazwie itemprop był równy „price”. W celu uzyskania adresu obrazka skorzystano z kombinacji selektorów css oraz xpath. Znacznie ułatwiło to dostęp do pożądanej danej.

W wyniku korzystania z takiego podejścia do scrapingu (tzn. pobrania wielu odnośników, a następnie wysłanie osobnego żądania do każdego z nich) serwer Nike zaczął blokować, odrzucać większość żądań. W celu rozwiązania tego problemu wprowadzono opóźnienie wysyłania żądań dodając pole w pliku pająka settings.py

```
DOWNLOAD_DELAY=3
```

Finalnym rezultatem użycia pająka korzystającego z tych funkcji jest plik formatu json o następującej strukturze:

```
{"url": "http://store.nike.com/gb/en_gb/pd/air-max-zero-  
bettrue-shoe/pid-11189799/pgid-11276418", "price": "\u00a3115",  
"image":  
"http://images.nike.com/is/image/DotCom/PDP_HERO_S/NIKE-AIR-  
KIT-QS-789695_101_A_PREM.jpg", "title": "Nike Air Max Zero  
BeTrue"},
```

Z danych funkcji korzysta pająk NikeSpider.

Zespół dostrzegł jednak problem. Owe funkcje zwracały jedynie około 54 obiektów, podczas gdy dana kategoria na stronie posiada około 700 produktów. W rezultacie badań wynikło, iż dane na stronie generowane są dynamicznie z użyciem ,czy to Javascriptu ,czy też metodyk Ajax. Problem okazał się dość popularny w społeczności korzystającej z frameworka Scrapy. Wynikło, iż w ogólnym przekonaniu Scrapy sprawdza się tylko i wyłącznie do operowania na stronach statycznych. Po jednak głębszych poszukiwaniach udało się wypracować rozwiązanie problemu, które przerosło najśmielsze oczekiwania.

Obserwując zachowanie się strony, można było zauważyć, iż dane ładowane są dopiero w momencie użycia paska przesuwania. W celu uzyskania dokładniejszych informacji skorzystano z dodatku Firebug do przeglądarki internetowej Firefox, a

następnie prześledzono sieciową część komunikacji klienta ze stroną przy użyciu XHR(XMLHttpRequest), czyli żądań wykonywanych już po załadowaniu się strony internetowej, w trakcie interakcji z użytkownikiem.

URL	Status	Domena	Rozmiar	Zdalny IP	Os czasu
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	9,6 KB		733ms
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	9,7 KB		629ms
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	10,3 KB		653ms
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	8,7 KB		862ms
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	7,3 KB		285ms
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	9,2 KB		11ms
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	8,6 KB		47ms
GET gridwallData?country=GB&lang=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X	200 OK (BFCache)	store.nike.com	6,0 KB		1,5s

Rysunek 6 - Zapis komunikacji sieciowej z serwerem Nike w przypadku użycia paska przesuwania.

Odkryto, iż w momencie użycia paska przesuwania generowana jest metoda GET oraz ścieżce `gridwallData?country=GB&lang_locale=en_GB&gridwallPath=mens-shoes/7puZoi3&pn=X`, gdzie X to numer zależny od miejsca, w którym znajduje się klient(0,12). Okazało się również, iż odpowiedzią serwera na dane żądanie jest plik json zawierający wszystkie dane o produktach. W celu analizy tak ogromnych plików json skorzystano ze strony <http://json.parser.online.fr/>.

The screenshot shows the 'Json Parser Online' interface. The left pane displays the raw JSON response, which is a large object containing metadata and a 'products' array. The right pane shows the parsed JSON structure, where the 'products' array is expanded, revealing details for multiple shoe items, such as 'Basketball Shoe' and 'Jordan 6'. The interface includes a search bar and various navigation options.

Rysunek 7 - Przedstawienie korzystania z usługi znajdującej się na stronie <http://json.parser.online.fr/>

Należało więc zmienić podejście odnośnie działania pająka. Po licznych próbach finalna funkcja podstawowa pająka prezentowała się następująco:

```
def parse(self, response):
    for i in range(0,12):
```

```

        yield FormRequest(url="http://store.nike.com/html-
services/gridwallData?country=GB&lang_locale=en_GB&gridwallPat
h=mens-shoes/7puZoi3&pn=" +
str(i),method="GET",callback=self.parser2)

```

Analogicznie jak funkcja poprzedniego pająka rezultatem tej funkcji było wywołanie kolejnej funkcji, jednak w tym przypadku przekazywana do niej odpowiedź była w formacie json. Powyższa struktura imituje akcje korzystania z paska przesuwania. Wysyła więc żądanie GET oraz określoną ścieżkę, będąc świadomą ile takich żądań strona obsługuje.

Wywoływana funkcja parser2 odpowiada za załadowanie odpowiedzi jako unicodu, a następnie pobranie interesujących nas informacji i sparsowanie ich do nowego dokumentu json.

```

def parser2(self, response):
    jsonresponse = json.loads(response.body_as_unicode())
    #jsonresponse = json.loads(response.body_as_unicode())
    #yield jsonresponse
    for sel in jsonresponse['sections']:
        for sel2 in sel['products']:
            item = ItemApp()
            item['price'] = sel2['localPrice']
            item['title'] = sel2['title']
            item['url'] = sel2['pdpUrl']
            item['image'] = sel2['spriteSheet']
            yield item

```

W drodze generowania powyższego kodu nie uchroniono się od licznych trudności związanych z dostępem do odpowiednich pól pliku json. Ich przyczyną był wysoki poziom złożoności pól oraz rozmiar całego pliku. Przykładowe błędne próby dostępu do pól przed skorzystaniem z pętli.

```

item['price'] =
jsonresponse['sections'][0]['products'][0]['localPrice']

```

oraz

```
item['price'] =  
jsonresponse['sections'][0]['products'][0]['colorways'][0]['lo  
calPrice']
```

Z finalnych wersji funkcji `parse` oraz `parser2` korzysta pająk `NikeScroll`.

Rezultatem re definicji podejścia do scrapowania strony Nike, było uzyskiwanie ogromnych plików json będącymi odpowiedziami na kilka żądań. Sprawilo to, iż cały proces pobrania ogromnej ilości danych zajmuje mniej niż minutę oraz wyłączyło konieczność wprowadzenia ograniczeń ilości żądań w pliku `settings.py`.

Okazało się jednak, iż odpowiedzi zwrotne na żądania GET generują się dopiero w momencie użycia paska przesuwania tak więc początkowe przedmioty(około 54) nie były uwzględnione w tych odpowiedziach. Stąd też zapotrzebowanie na użycie obu pająków w celu uzyskania wszystkich danych. Korzystanie z obu pająków ukazuje ogromny kontrast. Podczas gdy `NikeSpider` uzyskuje informacje o około 54 obiektach i wymaga to przybliżonego czasu 5 minut, `NikeScroll` ekstraktuje 650 obiektów w czasie mniejszym niż 1 minuta.

3.2 Spider Allegro

Klasa `SitemapSpider` pozwala na crawlowanie strony poprzez wykrycie adresu URL, który przechowuje plik `sitemapy`.

Wspiera ona zarówno zagnieżdżone `sitemapy`, jak i adresy URL zawarte w pliku `robots.txt`.

Na etapie implementacji zespół wyklarował wspólne dla obu serwisów, najistotniejsze informacje potrzebne do porównywania butów:

- Nazwa modelu buta/nazwa aukcji,
- Cena,
- Adres URL buta/aukcji,
- Adres URL zdjęcia butów.

Implementacja SitemapSpider pobierającego powyższe dane przedstawia się następująco:

```
import scrapy

from scrapy.spiders import SitemapSpider
from scrapy.selector import Selector
from scrapy.exceptions import DropItem

from allegro.items import AllegroItem

class AllegroSpider(SitemapSpider):
    name = 'allegro-spider'
    allowed_domains = ['allegro.pl']
    sitemap_urls = ['http://allegro.pl/robots.txt']
    sitemap_rules = [
        ('sportowe-nike', 'parse_nike'),
    ]
    def parse_nike(self, response):
        yield {
            'price': response.selector.xpath('//meta[@itemprop="price"]/@content').extract(),
            'title': response.selector.xpath('//meta[@itemprop="name"]/@content').extract(),
            'image': response.selector.xpath('//meta[@itemprop="image"]/@content').extract(),
            'url': response.selector.xpath('//meta[@itemprop="url"]/@content').extract()
        }
```

Program będzie przeszukiwał strony zawierające słowo „nike” w adresie URL. Pobieranie konkretnych informacji z poszczególnych stron odbywa się z użyciem języka ścieżek XML – Xpath.

Problemem jaki pojawiał się podczas kolejnych testów, było to, że w przypadku aukcji, które były już wygasłe, bądź unieważnione Allegro Spider pobierał dane zawierające puste pola – wpisy takich butów, nie tylko utrudniały implementację aplikacji zestawiającej ze sobą te dane, ale także wpisy te były bezwartościowe dla potencjalnego użytkownika aplikacji. Problem ten został usunięty poprzez implementację specjalnego filtra, który odrzucał wpisy zawierające niekompletne dane. Został on umieszczony w pliku pipelines.py w projekcie pająka i przedstawiał się następująco:

```

class PricePipeline(object):

    def process_item(self, item, spider):

        # to test if only "job_id" is empty,
        # change to:
        # if not(item["job_id"]):
        if not(all(item.values())):
            raise DropItem()
        else:
            return item

```

Innym z problemów było to, że aukcje nie zawsze zawierały buty (sporadycznie zdarzają się także ubrania Nike). Po stronie pająka jest to problem nie do przejścia. Ewentualne zawężenie zakresu poszukiwań mogłoby zmniejszyć faktyczną ilość butów, gdyż użytkownicy serwisu Allegro nie przywiązują dbałości do odpowiedniego nazewnictwa wystawianych przez siebie aukcji.

3.3 Aplikacja NikeSearch

W celu przedstawienia list butów z obydwu serwisów w sposób graficzny, dane dostępne w plikach .json są początkowo deserializowane z użyciem biblioteki Newtonsoft.Json:

```

2 references
public static IList<ProduktOb> Deserialize(string path)
{
    using (StreamReader r = new StreamReader(System.IO.Path.GetFullPath(path)))
    {
        string json = r.ReadToEnd();
        ProduktOb productObj = new ProduktOb();
        IList<ProduktOb> validProdcuts;
        validProdcuts = JsonConvert.DeserializeObject<IList<ProduktOb>>(json);
        return validProdcuts;
    }
}

```

Deserializowane dane zapisywane są do pól zmiennych typu Object zawartych w klasie ProjektOb.

Aby umieścić zdeserializowane dane w kontrolerze WPF typu ListBox należy utworzyć kolekcję obserwowalną (ObservableCollection). Implementacja tych kolekcji przedstawia się następująco:

```

2 references
public class ProduktyAllegro : ObservableCollection<Produkt>
{
    1 reference
    public ProduktyAllegro()
    {
        IList<Program.ProduktOb> produkty = Program.Deserialize(Program.currentAllegroJson);
        foreach (var produkt in produkty)
        {
            Add(new Produkt(produkt.url.ToString().Split('')[1], produkt.price.ToString().Split('')[1], produkt.image.ToString().Split('')[1],
                produkt.title.ToString().Split('')[1]));
        }
    }
}
3 references
public class ProduktyNike : ObservableCollection<Produkt>
{
    2 references
    public ProduktyNike()
    {
        IList<Program.ProduktOb> produkty = Program.Deserialize(Program.currentNikeJson);
        foreach (var produkt in produkty)
        {
            string cenastr = produkt.price.ToString().Split(';')[1].Replace(".", ",");
            double cena = 5.58 * Convert.ToDouble(cenastr);
            cenastr=Math.Round(cena,2).ToString();
            Add(new Produkt(produkt.url.ToString(), cenastr, produkt.image.ToString(), produkt.title.ToString()));
        }
    }
}

```

W przypadku kolekcji produktów Nike zauważyć można, że pole dotyczące ceny ulega konwersji z funtów na złotówki, aby użytkownik otrzymywał bardziej przejrzyste porównanie cen.

Metoda szukająca podobieństw w nazwie pomiędzy modelami butów z oficjalnej strony Nike i aukcjami Allegro jest prostym algorytmem, który od nazwy docelowej modelu buta firmy Nike odejmuje kolejno litery do momenty niż łańcuch tekstowy nie będzie zawierał się w nazwie aukcji na Allegro:

```

while (listBox.Items.Count==0 && comparer!="Nike "&&comparer!="Nike" && comparer!="Hurley")
{
    Console.WriteLine("zero elementow");
    listBox.Items.Filter = delegate (object obj)
    {
        Produkt pr = (Produkt)obj;
        string str = pr.title.ToString();
        if (String.IsNullOrEmpty(str)) return false;
        int index = str.IndexOf(comparer, StringComparison.CurrentCultureIgnoreCase);
        return (index > -1);
    };
    comparer = comparer.Remove(comparer.Length - 1);
    Console.WriteLine(comparer);
}

```

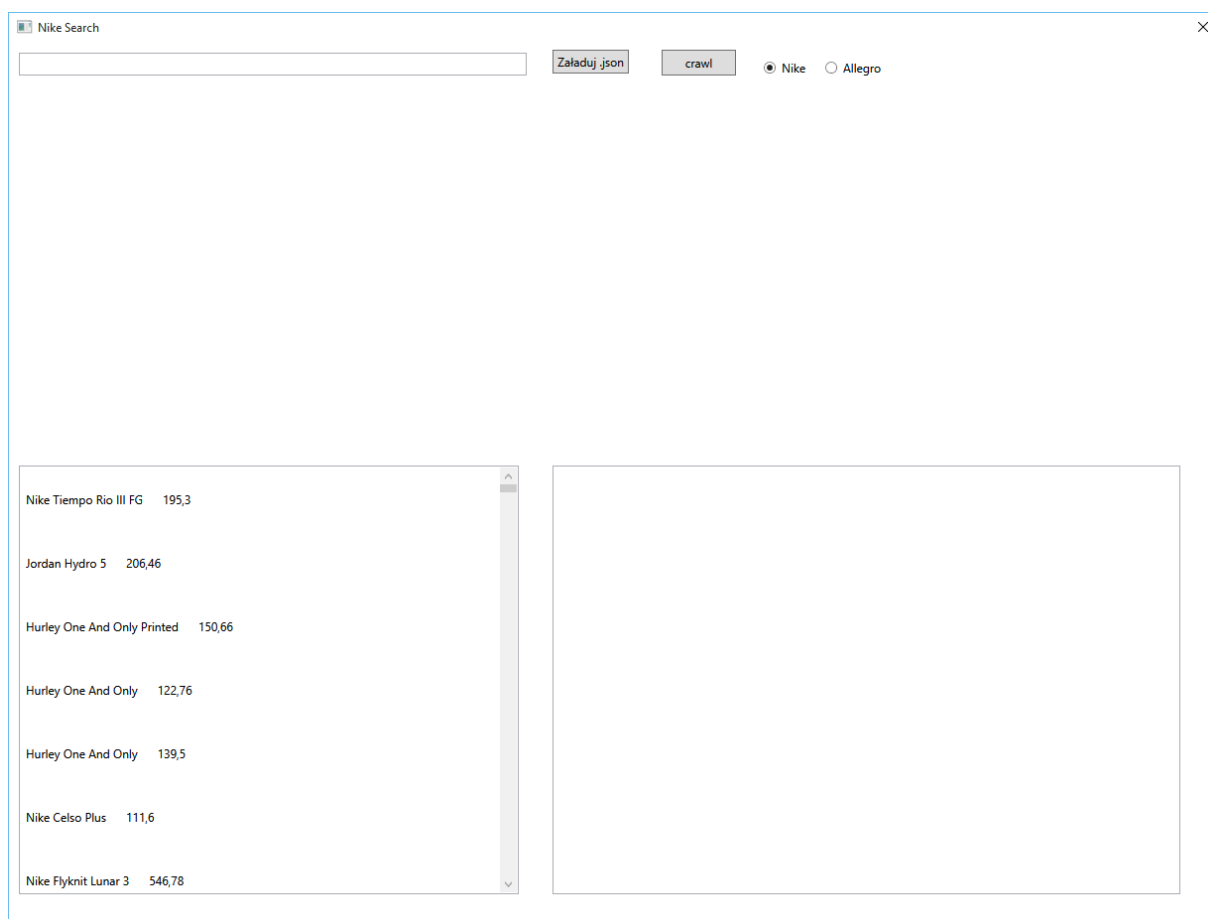
Rozwiązanie takie zwraca wiarygodne wyniki, gdyż według analizy zespołu nazewnictwo butów Nike stworzone jest według zasady „od ogółu do szczegółu”

NIKE [Nazwa Modelu] [Numer wersji modelu] [Dodatkowa użyta technologia][Kolor][Dodatkowe informacje: rozmiarówka, ID]

Dodatkowo w warunku wyszukiwania pomijane są słowa „Nike” i „Hurley”. Pierwsze ze słów zwracało by zbyt ogólne wyniki, drugie jest nazwą firmy surferskiej, którą wykupił Nike, w związku z czym nie dotyczy realizowanego zadania.

3 Instrukcja obsługi aplikacji

Po uruchomieniu aplikacji naszym oczom ukaże się następujące okno:




W lewym dolnym rogu widoczna jest lista wszystkich butów, dostępnych na stronie store.nike.com. Obok nazwy, widoczna jest cena butów w przeliczeniu na złotówki. W celu uzyskania dokładniejszego poglądu na buty, należy kliknąć na odpowiadającą nam pozycję:

Zaladuj json

crawl

☒ Nike
☐ Allegro

Nike Tiempo Rio III FG



Cena: 195,3 zł

[Link do Nike](#)

Nike Tiempo Rio III FG	195,3
Jordan Hydro 5	206,46
Hurley One And Only Printed	150,66
Hurley One And Only	122,76
Hurley One And Only	139,5
Nike Celso Plus	111,6
Nike Flyknit Lunar 3	546,78

BUTY SPORTOWE NIKE TIEMPO RIO III IC - 40 do 47,5 139,0

Ponad listą ukaże się pobrany obraz buta, jego nazwa, cena oraz hiperłącze, które przeniesie nas bezpośrednio do strony buta w witrynie nike.com:

Nike

Nike+

Hurley

Join/Log in To Nike+ Account

Site Feedback

Help



MEN

WOMEN

BOYS

GIRLS

CUSTOMISE

NIKE TIEMPO RIO III FG

NIKE'S FIRM-GROUND FOOTBALL BOOT

£35

Rate this product

Clear Jade/White/Black

Style: 819235-307

Size Chart

SIZE

QTY (1)

ADD TO CART

Save to Wish List

FREE DELIVERY Applies to orders of £50 or more. Returns are always free. [See details](#)


SHARE

Po wybraniu buta, na drugiej liście pojawiają się proponowane aukcje znalezione na Allegro. Zaznaczenie tej aukcji da analogiczny efekt, jak w przypadku listy Nike:

Nike Search

Zaladuj .json
craw!
☒ Nike
☐ Allegro

Nike Tiempo Rio III FG




Cena: 195,3 zł

[Link do Nike](#)

Nike Tiempo Rio III FG	195,3
Jordan Hydro 5	206,46
Hurley One And Only Printed	150,66
Hurley One And Only	122,76
Hurley One And Only	139,5
Nike Celso Plus	111,6
Nike Flyknit Lunar 3	546,78

BUTY SPORTOWE NIKE TIEMPO RIO III IC - 40 do 47,5



Cena: 139.0 zł

[Link do Allegro](#)

BUTY SPORTOWE NIKE TIEMPO RIO III IC - 40 do 47,5	139.0
---	-------

Aby wyszukać konkretnej nazwy buta, należy wpisać go w pole tekstowe w lewym górnym rogu. Nie trzeba zatwierdzać wyszukiwania – filtr listy automatycznie dopasowuje wyniki do wpisanej przez nas treści:

Nike Search

Załaduj .json

crawl

☒ Nike
☐ Allegro

Nike Air Max 90 Ultra Moire

Cena: 613,8 zł

[Link do Nike](#)

Nike Air Max BW Ultra Knit Jacquard Premium	725,4
Nike Flyknit Air Max	800,67
Nike Air Max 90 Premium iD	753,3
Nike Air Max 90 Ultra Moire	613,8
Nike Air Max 90 iD	697,5
Nike Air Max Zero BeTrue	641,7
Nike Air Max Zero iD	864,9

BUTY MĘSKIE SPORTOWE NIKE AIR MAX 90 LTR r 43

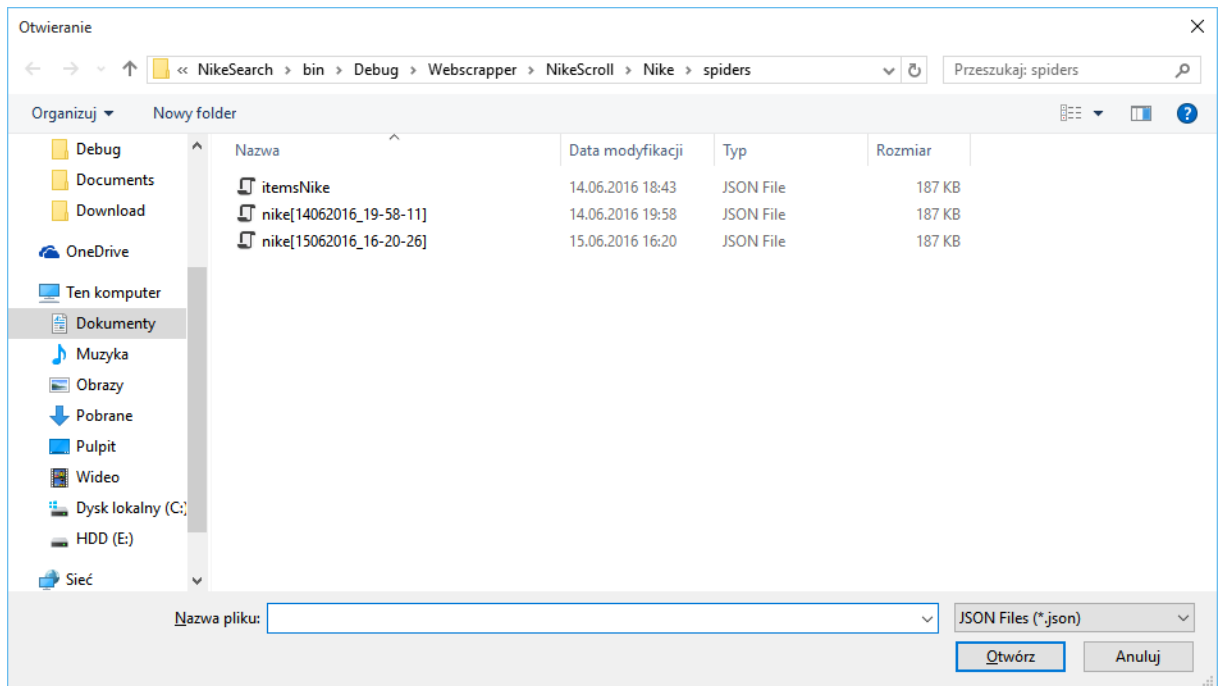
Cena: 449.99 zł

[Link do Allegro](#)

BUTY MĘSKIE SPORTOWE NIKE AIR MAX 90 LTR r 43	449.99
BUTY MĘSKIE SPORTOWE NIKE AIR MAX 90 LTR r 46	449.99
BUTY SPORTOWE NIKE AIR MAX 90 MESKIE OREO 36-45	180.00
BUTY MĘSKIE SPORTOWE NIKE AIR MAX 90 LTR r 44,5	449.99
BUTY MĘSKIE SPORTOWE NIKE AIR MAX 90 LTR r 44	449.99
Buty Damskie Sportowe Nike Air Max 90 MESH GS 36.5	379.00
Buty Damskie Sportowe Nike Air Max 90 MESH GS 40	379.0

Przyciski RadioButton w górnej prawej części ekranu służą do wybrania, listy chcemy wykonać operacje dostępne pod przyciskami:

- Załaduj .json – pozwala na wybranie pliku .json, który chcemy użyć do przeglądania w programie NikeSearch. Po wybraniu odpowiedniego pliku załaduje się on automatycznie w programie, bez konieczności jego restartu



- Crawl – pozwala na uruchomienie pajaków po wybraniu tej opcji uruchomione zostanie okno konsoli. Plik .json będzie nazwany automatycznie według daty i godziny crawlowania. Poniższy przykład w przypadku naciśnięciu przycisku „Crawl” dla serwisu Allegro:

```
C:\WINDOWS\SYSTEM32\cmd.exe

'scrappy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
'scrappy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
'scrappy.downloadermiddlewares.redirect.RedirectMiddleware',
'scrappy.downloadermiddlewares.cookies.CookiesMiddleware',
'scrappy.downloadermiddlewares.chunked.ChunkedTransferMiddleware',
'scrappy.downloadermiddlewares.stats.DownloaderStats']
2016-06-17 23:05:52 [scrappy] INFO: Enabled spider middlewares:
['scrappy.spidermiddlewares.httperror.HttpErrorMiddleware',
'scrappy.spidermiddlewares.offsite.OffsiteMiddleware',
'scrappy.spidermiddlewares.referer.RefererMiddleware',
'scrappy.spidermiddlewares.urllength.UrlLengthMiddleware',
'scrappy.spidermiddlewares.depth.DepthMiddleware']
2016-06-17 23:05:52 [scrappy] INFO: Enabled item pipelines:
['allegro.pipelines.PricePipeline']
2016-06-17 23:05:52 [scrappy] INFO: Spider opened
2016-06-17 23:05:52 [scrappy] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 i
tems (at 0 items/min)
2016-06-17 23:05:52 [scrappy] DEBUG: Telnet console listening on 127.0.0.1:6023
2016-06-17 23:05:52 [scrappy] DEBUG: Crawled (200) <GET http://allegro.pl/robots.
txt> (referer: None)
2016-06-17 23:05:52 [scrappy] DEBUG: Crawled (200) <GET http://allegro.pl/robots.
txt> (referer: None)
2016-06-17 23:05:52 [scrappy] DEBUG: Crawled (200) <GET http://allegro.pl/sitemap
/sitemap_index_pl.xml> (referer: http://allegro.pl/robots.txt)
```

4 Wykorzystane technologie

Wykorzystane technologie, oprogramowanie wraz z uzasadnieniem.

- Framework Scrapy – najbardziej popularne oraz rozbudowane narzędzie umożliwiające tworzenie web-scrapery, oraz web-crawlers, posiadające największą społeczność użytkowników wśród technologii tego typu. Dodatkowo pełne wsparcie twórców w postaci licznych poradników i przykładów użycia
- Python 2.7 Anaconda – wykorzystywany framework został stworzony przy użyciu Pythona stąd też potrzeba użycia tego języka. Jest on jednocześnie jednym z najpotężniejszych języków umożliwiających pobieranie danych ze stron internetowych.
- PyCharm – najbardziej rozbudowano środowisko (IDE) języka Python, polecane przez prowadzącego zajęcia projektowe, dostępne dla zespołu dzięki

modułowi eProgramy systemu Politechniki Poznańskiej udostępniającego licencje zakupione przez uczelnię.

- Json – przejrzysty, nowoczesny format zapisu danych. W pełni wystarczający na potrzeby projektu. Dodatkową zaletą jest fakt, iż uzyskane odpowiedzi w przypadku stosowania metod dynamicznych na stronach internetowych są właśnie w formacie json, co ułatwia parsowanie.
- C# - w celu stworzenia przykładowej aplikacji korzystającej z pozyskanych danych zespół wykorzystał język, w którym tworzeniu oprogramowania posiada największe doświadczenie.
- Visual Studio 2015 – środowisko pozwalające na tworzenie oprogramowania w języku C#, dostępne dzięki wykupionym i udostępnionym przez Politechnikę Poznańską licencjom, a także bardzo dobrze znane zespołowi.
- WPF (Windows Presentation Foundation) – narzędzie umożliwiające tworzenie form, okienek znacznie bardziej rozbudowane i nowoczesne od Windows Forms, a także dobrze znane członkom zespołu .

Podsumowując, wybór języka, w którym tworzone pająki był uwarunkowany wyborem frameworka o najlepszej reputacji, wsparciu oraz możliwościach. Natomiast wybór języka i technologii użytych do stworzenia aplikacji korzystającej z danych podyktowany był doświadczeniem zespołu projektowego.

5 Harmonogram prac

Poniżej przedstawiono listę ważniejszych zadań jakie należało zrealizować w ramach pracy nad projektem:

1. Prace organizacyjne
2. Utworzenie projektu aplikacji – Konrad Ferbes, Adrian Leoniak
3. Utworzenie Pajaka Nike – Adrian Leoniak
4. Utworzenie Pajaka Allegro – Konrad Ferbes
5. Stworzenie aplikacji przetwarzającej zebrane dane – Konrad Ferbes, Adrian Leoniak

6. Testy / poprawki - Konrad Ferbes, Adrian Leoniak
7. Utworzenie dokumentacji systemu - Konrad Ferbes, Adrian Leoniak

6 Podsumowanie

Projekt zakończył się pełnym sukcesem w jego głównym zakresie – pobieraniu danych ze stron internetowych. Dodatkowo zadowalającym materiałem ukazującym przykładowe wykorzystanie zdobytych informacji okazała się aplikacja. Jej ewentualne braki wynikają ze zbyt niedokładnego algorytmu wyszukiującego przedmioty z jednego sklepu w drugim oraz z natury serwisu allegro, za którego ofertę produktów odpowiedzialni są zarówno zwykli użytkownicy, jak i sklepy, które nie stosują precyzyjnego i wspólnego nazewnictwa aukcji.

W czasie prac napotkano wiele problemów, jednak dzięki pomocy wyszukanej w sieci, licznym dociekaniom oraz próbom przeprowadzonym zarówno w powłoce frameworka, jak i stosując pająki, udało się wszystkie rozwiązać.

Projekt okazał się wymagający w o wiele szerszej tematyce, niż wstępnie przewidywano. Od poznania takich podstaw jak struktury statycznych stron internetowych, plików w formacie json, ścieżek xpath po zaawansowane nowoczesne technologie takie jak dynamicznie generowana zawartość stron internetowych, które to posiadają nieliczne rozwiązania problemów dotyczące frameworka Scrapy.

Dodatkowo zespół miał okazję zaznajomić się z językiem Python (w wersji Anaconda2), niezwykle potężnym dodatkiem Firebug do przeglądarki FireFox, a także profesjonalnym środowiskiem PyCharm. W przypadku problemów z tzw. Infinity-scrolllem, czyli akcjami paska przesuwania generującymi dynamiczne żądania poznano komunikaty XHR oraz w procesie poszukiwania rozwiązań pobieżnie zapoznano się z ogromnym narzędziem Selenium (konkretnie z implementacją jego fragmentów w frameworku Scrapy). Napotkane błędy z odrzucaniem żądań przez serwer doświadczyły zespół w kwestii podstawowych zabezpieczeń serwerów przed przeciążeniami oraz prawidłowej konfiguracji pająków.

Rezultatem działań zespołu jest w pełni działająca aplikacja umożliwiająca uruchamianie trzech pająków (dwóch związanych ze stroną Nike oraz jednym

związanym z serwisem Allegro), import danych z plików json, wyszukiwanie produktów (konkretnie obuwia marki Nike) w serwisach Nike oraz Allegro, a także próby odnajdowania produktów jednego sklepu w drugim serwisie. Pająki identyfikują się serwerom, informując o autorach oraz celu działania. Przestrzegają one zasad ustalonych przez twórców serwisów zapisanych w plikach robots.txt, a także w wymagających tego sytuacjach ograniczają ilość wysyłanych przez siebie żądań. Aplikacja w sposób przejrzysty przedstawia podstawowe informacje o produktach, ich nazwę, cenę, obrazek, a także hiperłącze pozwalające na przejście do strony sklepu, serwisu dotyczącej wybranego produktu.