МИНОБРНАУКИ РОСИИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра информатики и программирования

Анализ методов парсинга сайтов

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

студента 4 курса 441 группы

направления 02.03.03 математическое обеспечение и администрирование информационных систем

Факультет компьютерных наук и информационных технологий

Турченкова Павла Александровича

Научный руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, уч. степень, уч. звание подпись, дата инициалы, фамилия

Консультант

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, уч. степень, уч. звание подпись, дата инициалы, фамилия

Заф. кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, уч. степень, уч. звание подпись, дата инициалы, фамилия

Саратов 2019

**Содержание**

[Введение 4](#_Toc9714936)

[1 Отбор библиотек для парсинга 5](#_Toc9714937)

[2 Реализация на языке Java 7](#_Toc9714938)

[2.1 Описание библиотеки Jsoup 7](#_Toc9714939)

[2.2 Описание инструмента Selenium 9](#_Toc9714940)

[2.3 Описание библиотеки Unirest 11](#_Toc9714941)

[3 Реализация на языке C# 18](#_Toc9714942)

[3.1 Описание библиотеки HtmlAgilityPack 18](#_Toc9714943)

[3.2 Описание библиотеки Fizzler 19](#_Toc9714944)

[3.3 Описание библиотеки AngleSharp 20](#_Toc9714945)

[3.4 Описание библиотеки CsQuery 22](#_Toc9714946)

[3.5 Описание библиотеки RestSharp 23](#_Toc9714947)

[3.6 Описание инструмента Selenium 25](#_Toc9714948)

[4 Реализация на языке Python 28](#_Toc9714949)

[4.1 Описание библиотеки BeautifulSoup 28](#_Toc9714950)

[4.2 Описание фреймворка Scrapy 30](#_Toc9714951)

[4.3 Описание библиотеки Request 33](#_Toc9714952)

[4.4 Описание инструмента Selenium 35](#_Toc9714953)

[5 Сравнение скорости реализаций 38](#_Toc9714954)

[5.1 Сравнение реализаций парсера для сайта MegaBonus 38](#_Toc9714955)

[5.2 Сравнение реализаций парсера для сайта LetyShops 40](#_Toc9714956)

[6 Алгоритмы парсинга сайтов 42](#_Toc9714957)

[6.1 Алгоритм парсинга сайта MegaBonus 42](#_Toc9714958)

[6.2 Алгоритм парсинга сайта LetyShops 48](#_Toc9714959)

[6.3 Решение проблем при парсинге сайтов 54](#_Toc9714960)

[7 Описание визуализации результатов 56](#_Toc9714961)

[7.1 Использование MVC 56](#_Toc9714962)

[7.2 Вывод в Excel файл 57](#_Toc9714963)

[7.3 Вывод в CSV файл 59](#_Toc9714964)

[Заключение 61](#_Toc9714965)

[Список использованной литературы 62](#_Toc9714966)

[Приложение А. Реализация парсера с помощью JSoup 66](#_Toc9714967)

[Приложение Б. Реализация парсера с помощью Selenium Java 69](#_Toc9714968)

[Приложение В. Реализация парсера с помощью Unirest для сайта Kopikot.ru 72](#_Toc9714969)

[Приложение Г. Реализация парсера с помощью HtmlAgilityPack 75](#_Toc9714970)

[Приложение Д. Реализация парсера с помощью Fizzler 79](#_Toc9714971)

[Приложение Ж. Реализация парсера с помощью AngleSharp 81](#_Toc9714972)

[Приложение И. Реализация парсера с помощью CsQuery 84](#_Toc9714973)

[Приложение К. Реализация парсера с помощью RestSharp для сайта Kopikot.ru 86](#_Toc9714974)

[Приложение Л. Реализация парсера с помощью Selenium C# 89](#_Toc9714975)

[Приложение М. Реализация парсера с помощью BeautifulSoup 93](#_Toc9714976)

[Приложение Н. Реализация парсера с помощью Scrapy 95](#_Toc9714977)

[Приложение П. Реализация парсера с помощью Requests для сайта Kopikot.ru 98](#_Toc9714978)

[Приложение Р. Реализация парсера с помощью Selenium Python и BeautifulSoup 100](#_Toc9714979)

# Введение

Интернет постоянно расширяется. Каждый день появляется всё больше сайтов, на которых размещена различная информация. Человеку с каждым днём труднее найти информацию самостоятельно среди сайтов, поэтому роль парсинга сайтов становится всё значительнее.

Парсинг – это метод получения контента, который состоит в том, что специально написанный алгоритм заходит на страницы сайта и собирает информацию в указанных тегах, либо обращается к серверу и берёт информацию, размещённую на сайте.

На сегодняшний день существует достаточное количество компаний, которые занимаются парсингом и анализом сайтов. Например, компания Meltwater, которая разрабатывает и продает программное обеспечение для мониторинга СМИ и бизнес-аналитики [1]. К списку компаний, которые разрабатывают программы для парсинга, можно отнести Google и Яндекс [2].

Целью данной работы является реализация программы для парсинга сайта на языках программирования Java, C# и Python, а также анализ приведённых реализаций. Для этого необходимо решить следующие задачи: отобрать библиотеки для парсинга, реализовать программы, используя библиотеки, сравнить реализации по скорости выполнения парсинга и дать оценку.

# Отбор библиотек для парсинга

Для парсинга сайтов были отобраны специальные библиотеки для парсинга для трёх языков программирования: Java, C# и Python. Приведённые ниже библиотеки были отобраны из-за удобства в использовании и работоспособности с приведёнными для примеров сайтов.

Для языка программирования Java использованы следующие библиотеки:

1. Jsoup;
2. Unirest.

Для языка программирования C# использованы следующие библиотеки:

1. HtmlAgilityPack;
2. Fizzler;
3. AngleSharp;
4. CsQuery;
5. RestSharp.

Для языка программирования Python использованы следующие средства:

1. библиотека BeatifulSoup;
2. библиотека Request;
3. фреймворк Scrapy.

Помимо библиотек был использован Selenium – это набор различных бесплатных программных инструментов, каждый из которых имеет свой подход к поддержке автоматизации тестирования. Весь набор инструментов обеспечивает широкий набор функций тестирования, специально предназначенных для тестирования веб-приложений всех типов, что позволяет использовать множество опций для определения местоположения элементов пользовательского интерфейса. Одной из ключевых функций Selenium является поддержка выполнения тестов на нескольких платформах браузера.

В случае с парсингом сайтов Selenium предоставляет возможность полностью имитировать поведение пользователя на сайте, то есть переход по ссылкам, использование активных объектов, заполнение некоторых полей и так далее.

# Реализация на языке Java

В ходе работы над проектом были реализованы следующие способы парсинга контента сайтов на языке Java: Jsoup, Selenium и Unirest.

## Описание библиотеки Jsoup

Jsoup – это библиотека для анализа документа HTML. Jsoup предоставляет API для получения и манипулирования данными взятых либо с помощью URL, либо из HTML. У этой библиотеки есть методы похожие на DOM, CSS и JQuery для того, чтобы получать и обрабатывать данные [3].

Для парсинга сайта с помощью библиотеки Jsoup были использованы следующие методы:

* connect(String url) – в качестве входного аргумента требует строку в виде URL. Создает новый объект типа Connection для заданного URL, и используется для получения и анализа HTML-страницы;
* get() – выполняет GET запрос и анализирует результат. Возвращает объект Document;
* post() – выполняет POST запрос и анализирует результат. Возвращает объект Document;
* userAgent(String userAgent) – в качестве входного аргумента требует строку в виде агента пользователя, который выглядит следующим образом: «Mozilla/5.0 (Windows NT 6.2; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1667.0 Safari/537.36». Данный метод устанавливает заголовок запроса агента пользователя;
* parse(String html) – анализирует код HTML и возвращает объект Document;
* getElementsByClass(String className) – в качестве входного аргумента требует строку в виде наименования класса. Находит элементы, которые имеют этот данный класс и возвращает их. Тип возвращаемого значения Elements;
* getElementsByTag(String tagName) – в качестве входного аргумента требует строку в виде наименования html тэга. Находит элементы, которые имеют этот данный тэг и возвращает их. Тип возвращаемого значения Elements;
* attr(String key) – в качестве входного аргумента требует строку в виде наименования атрибута тэга. Возвращает содержимое атрибута;
* text() – возвращает текст внутри основного тэга и дочерних элементов;
* select(String cssQuery) – в качестве входного аргумента требует строку в виде css селектора. Возвращает элементы, которые соответствуют селектору.
* selectFirst(String cssQuery) – в качестве входного аргумента требует строку в виде css селектора. Возвращает первое соответствие заданному селектору. Аналогичный результат будет при select(String cssQuery).first(), но этот вариант менее эффективный, так как первый метод будет искать все совпадения, что будет затратно по времени, и только потом второй метод вернёт первый элемент [4][5][6].

При построении логики парсинга сайта использовалась панель разработчика в браузере Google Chrome, чтобы не писать css селекторы или XPath самостоятельно. Использование возможностей панели разработчика в браузере Google Chrome изображено на рисунке 1. Полный код программы приведён в Приложении А.

Плюсы библиотеки Jsoup:

* является интуитивно понятной библиотекой, которой легко пользоваться;
* содержит множество методов, которые облегчают парсинг сайтов;
* присутствует имитация пользователя через userAgent;
* имеется возможность взаимодействия с cookies.

Минусы библиотеки Jsoup:

* с помощью библиотеки Jsoup можно парсить только статические страницы, то есть нельзя работать с динамически подключаемым контентом;
* Jsoup не поддерживает XPath. Но css селекторы являются альтернативой;
* среди методов, которые возвращают элементы по классу, тэгам и так далее нет методов для возвращения только первого элемента. Всегда происходит полный обход html. Приходится пользоваться методом first().

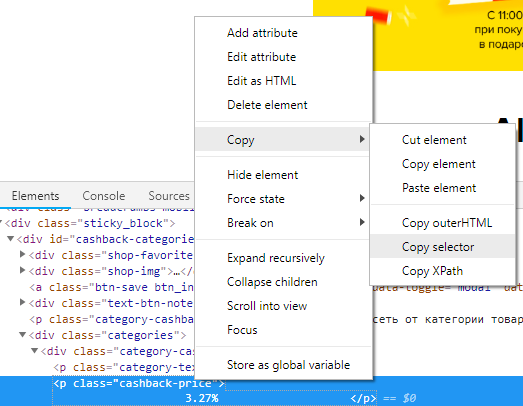


Рисунок 1 – использование возможностей панели разработчика в Google Chrome

## Описание инструмента Selenium

Для работы с Selenium нужно скачать вебдрайвер и подключить его через setProperty(String key, String value), где key – имя драйвера, то есть «webdriver.chrome.driver», а value – путь к самому драйверу. В данном проекте использовался chromedriver.exe. Также нужно подключить библиотеку selenium для взаимодействия с браузером.

Для парсинга сайта с помощью Selenium были использованы следующие методы:

* get(String URL) – в качестве входного аргумента требует строку в виде URL. С помощью этого метода происходит навигация по URL;
* navigate().to(String URL) - в качестве входного аргумента требует строку в виде URL. С помощью этого метода происходит навигация по URL. Разница этого метода и метода get(String URL) небольшая, но она есть: используя navigate(), можно использовать такие методы, как back(), forward(), refresh();
* findElement(By by) – в качестве параметра принимает класс By, который задает критерий поиска. Возвращает элемент типа WebElement, который соответствует заданному критерию;
* findElements(By by) - в качестве параметра принимает класс By, который задает критерий поиска. Возвращает список всех элементов типа WebElement, которые соответствуют заданному критерию;
* By – представляет из себя локатор, который уникально идентифицирует элемент страницы. Selenium предоставляет несколько способов использования локаторов для поиска элементов:

1. By.id – в качестве локатора используется атрибут id элемента страницы;
2. By.name – в качестве локатора используется атрибут name элемента страницы;
3. By.xpath – используется для поиска элемента по XPath выражению;
4. By.tagName – поиск по имени HTML тега;
5. By.className – поиск по классу элемента;
6. By.cssSelector – используется для поиска элемента по css селектору;
7. By.linkText – поиск ссылки с указанным текстом. Текст ссылки должен быть точным совпадением указанного текста;
8. By.partialLinkText – поиск ссылки по части с указанным текстом.

* getAttribute(String s) – в качестве параметра принимает строку в виде наименования атрибута. Возвращает текстовое представление найденного атрибута. Для получения текста внутри тэга нужно взять атрибут от «innerHTML».

Помимо методов для поиска элементов используются такие методы, как click(), который позволяет взаимодействовать с активными объектами такими как кнопки и ссылки и метод isDisplayed (), который проверяет отображение элемента [7][8][9]. Полный код программы приведён в Приложении Б.

При построении логики парсинга сайта использовалась панель разработчика в браузере Google Chrome, чтобы не писать css селекторы или XPath самостоятельно, как в примере с Jsoup. Использование возможностей панели разработчика в браузере Google Chrome изображено на рисунке 1.

Плюсы Selenium:

* имитация пользователя;
* методы для поиска как одного элемента, так и множества;
* поиск может осуществляться множеством локаторов.

Минусы Selenium:

* использует реальный браузер для работы, из-за чего скорость падает;
* необходимо скачивать дополнительные файлы.

## Описание библиотеки Unirest

Unirest – это набор облегченных HTTP-библиотек, доступных на нескольких языках, созданный и поддерживаемый Mashape, который также поддерживает API-интерфейс с открытым исходным кодом Gateway Kong [10].

Заголовки и параметры передаются с помощью API header() и fields(). Запрос выполняется при вызове метода asJson(). Кроме этого метода есть другие варианты, например asBinary(), asString() и asObject().

Чтобы передать несколько заголовков или полей, можно либо создать Map и передать их в .headers (<String, Object>) и .fields (<String, String>) соответственно, либо постепенно добавлять через .header(<String, Object>) и .field(<String, Object>) поле за полем [11].

Следующий код показывает, как с помощью запросов программа получает json с сайта [Kopikot.ru](https://www.kopikot.ru/), который в последствии обрабатывается.

Пример с использование Map:

1. HttpResponse<String> response = null;
2. Map<String, String> header = new HashMap<>();
3. header.put("User-Agent", "PostmanRuntime/7.11.0");
4. header.put("Accept", "\*/\*");
5. header.put("Cache-Control", "no-cache");
6. header.put("Postman-Token", "ac6a192d-72f5-4238-9321-55c1308e9846,09ec761f-7c53-4696-bf5a-dc44918e73c5");
7. header.put("Host", "d289b99uqa0t82.cloudfront.net");
8. header.put("accept-encoding", "gzip, deflate");
9. header.put("Connection", "keep-alive");
10. header.put("cache-control", "no-cache");
11. for (int i = 0; i <= 2000; i += 100) {
12. String url = "https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_" + i + "\_order\_popularity.json";
13. try {
14. response = Unirest.get(url)
15. .headers(header)
16. .asString();
17. } catch (UnirestException e) {
18. log.error(e + " : " + i);
19. break;
20. }

Пример без использования Map:

1. HttpResponse<String> response = null;
2. for (int i = 0; i <= 2000; i += 100) {
3. String url = "https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_" + i + "\_order\_popularity.json";
4. try {
5. response = Unirest.get(url)
6. .header("User-Agent", "PostmanRuntime/7.11.0")
7. .header("Accept", "\*/\*")
8. .header("Cache-Control", "no-cache")
9. .header("Postman-Token", "ac6a192d-72f5-4238-9321-55c1308e9846,09ec761f-7c53-4696-bf5a-dc44918e73c5")
10. .header("Host", "d289b99uqa0t82.cloudfront.net")
11. .header("accept-encoding", "gzip, deflate")
12. .header("Connection", "keep-alive")
13. .header("cache-control", "no-cache")
14. .asString();
15. } catch (UnirestException e) {
16. log.error(e + " : " + i);
17. break;
18. }
19. HttpResponse<String> finalResponse = response;
20. if (finalResponse != null) {
21. futures.add(pool.submit(() -> parseElements(finalResponse.getBody())));
22. }
23. }

Полный код программы приведён в Приложении В.

Для упрощения работы с запросами использовалось бесплатное приложение Postman. Используя это приложение, можно отправлять запросы на сайты с нужными заголовками и полями. В примере выше можно заметить такие заголовки, как User-Agent, Cache-Control и прочие. Некоторые заголовки Postman ставит автоматически.

Как можно заметить из примера сверху, URL, который передаётся в запрос отличается от URL самого сайта. Такие различия объясняются тем, запрос должен передаваться не по адресу самого сайта, а другому, который будет указан в панели разработчика. При отправке запроса на адрес самого магазина в ответе будет не html код страницы, а javaScript страницы.

Чтобы узнать URL, по которому передаются данные, нужно воспользоваться панелью разработчика и найти XHR – XMLHttpRequest, который отвечает за добавление данных на страницу. В случае с сайтом [Kopikot.ru](https://www.kopikot.ru/), запрос будет отправлен при нажатии кнопки «Больше». После этого в панели разработчика в разделе Network будет нужный XHR. Результат изображён на рисунке 2.

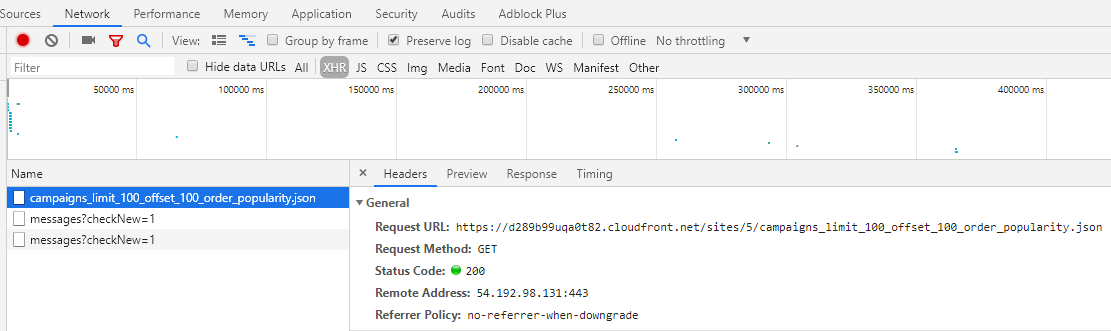


Рисунок 2 – XHR для 100 магазинов с сайта [Kopikot.ru](https://www.kopikot.ru/)

Как видно на рисунке 2, в разделе Headers указан Request URL. Так как при отображении магазинов было выбрано показывать по 100 магазинов, в URL видно это число: «offset\_100». При нажатии кнопки ещё раз в Network появляется ещё один XHR, но уже с числом 200. Кроме самого URL важно какой метод использует запрос: GET или POST. Посмотреть это можно в «Request Method» Результат виден на рисунке 3.

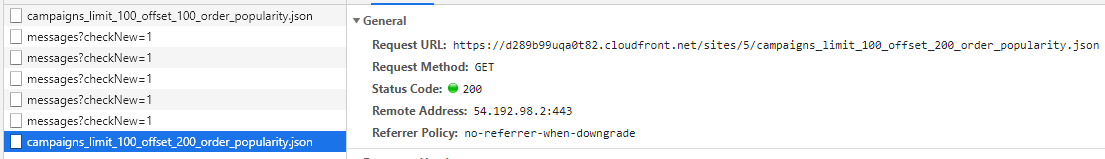


Рисунок 3 – XHR для 200 магазинов с сайта Kopikot.ru

Практически было выяснено, что таких XHR всего 14 штук: от 0 до 13.

Используя программу Postman, через этот URL отправляется запрос. В качестве ответа в разделе Body появляется JSON файл с нужной информацией. Результат запроса на рисунке 4.

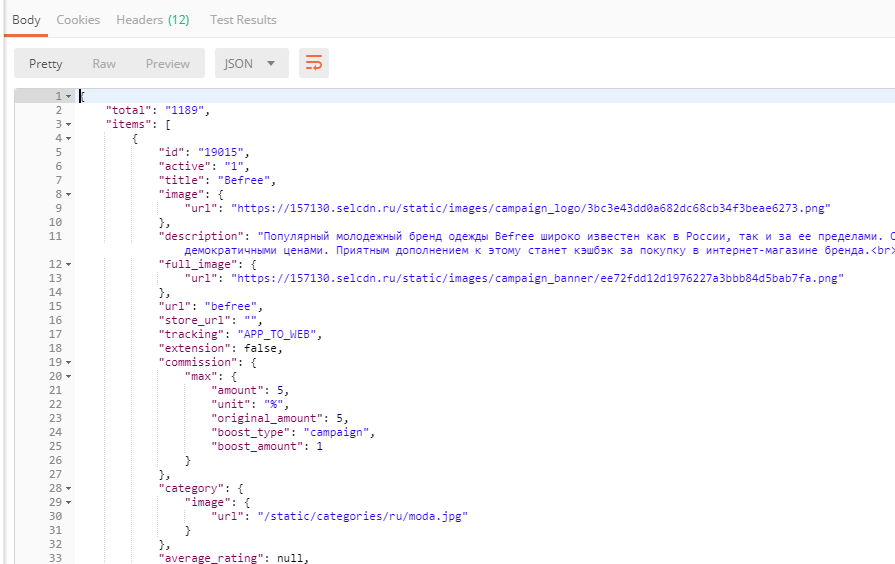


Рисунок 4 – результат запроса к URL через Postman

При просмотре какие запросы происходят на сервере важно обращать внимание на ту информацию, которая показана в Headers. Пример изображён на рисунке 5.

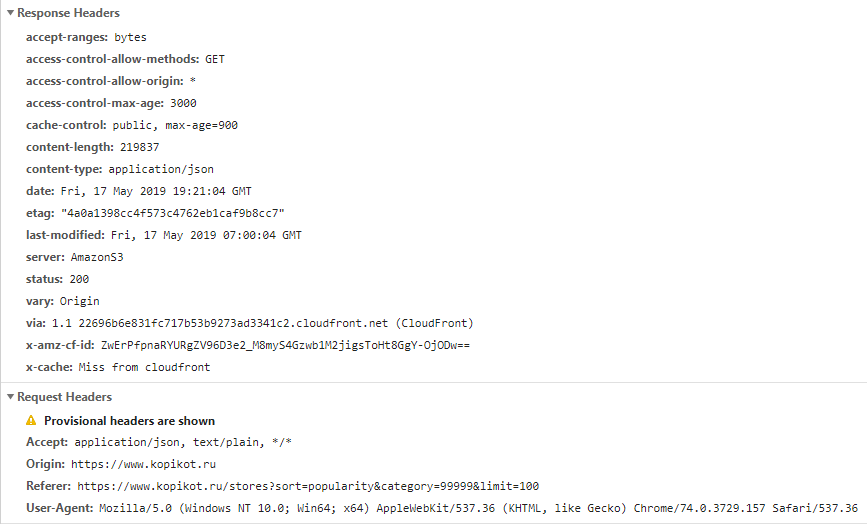


Рисунок 5 – Headers сайта Kopikot.ru

На этом рисунке можно увидеть, например, в каком формате будет ответ на запрос – поле content-type.

Также имеются случаи, когда при запросе передаются некоторые параметры. Для примера приведён Headers сайта [Cash4Brands.ru](https://cash4brands.ru/) на рисунке 6 и рисунке 7.

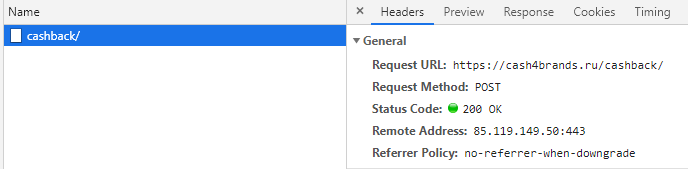


Рисунок 6 – Headers сайта Cash4Brands.ru часть 1

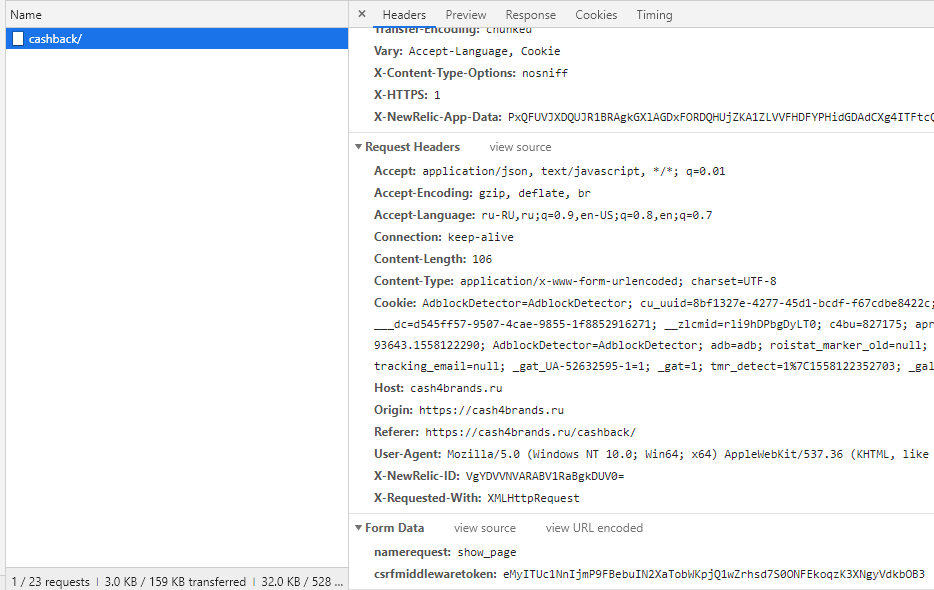


Рисунок 7 – Headers сайта Cash4Brands.ru часть 2

На рисунке 7 в «Form Data» можно заметить 2 параметра, которые передаются: namerequest и csrfmiddlewaretoken. В данном случае нужен лишь первый параметр, так как второй отвечает за cookies. У поля с именем csrfmiddlewaretoken значением является хэш для идентификатора сессии и секретный ключ [12]. Поле namerequest и его значение show\_page надо добавить в Body. Пример приведён на рисунке 8.

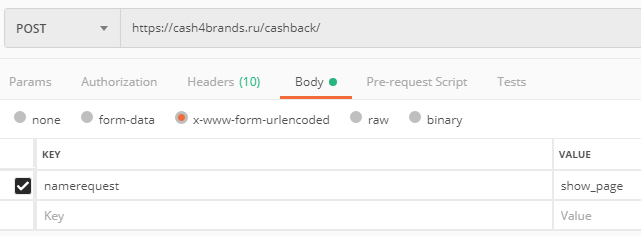


Рисунок 8 – параметры для запроса на Cash4Brands.ru

Результатом данного запроса является JSON, который в дальнейшем обрабатывается.

Программа Postman предоставляет возможность сгенерировать код для запроса. Для этого нужно нажать кнопку Code. В окне можно выбрать язык программирования и какие библиотеки использовать. Пример изображен на рисунке 9.

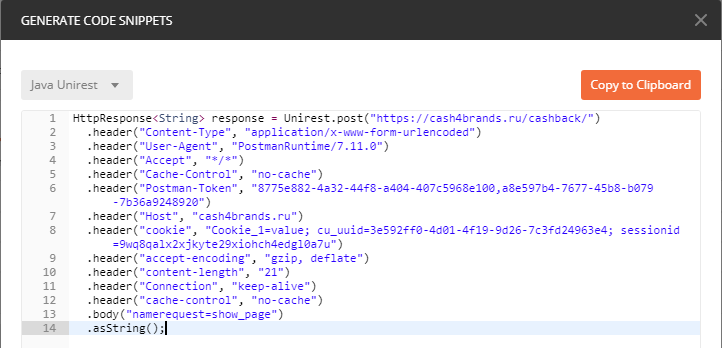


Рисунок 9 – генерация Unirest кода в Postman

После получения JSON с запроса, он обрабатывается с помощью парсера для JSON. Разбор JSON осуществляется с помощью библиотеки JsonPath. Для примера взят код для обработки данных с Kopikot.ru.

List<Object> items = JsonPath.read(response, "$..items[\*]");

В этой строчке в список items записываются все элементы, которые содержатся в тэге «items». Далее элементы этого списка передаются методам, которые извлекают из этих элементов нужную информацию.

String label = JsonPath.read(item, "$.commission.max.unit");

В этой строчке кода из метода, который извлекает валюту кэшбэка. В данному случае это процент. Через точку происходит обращение к последующим элементам для извлечения информации. Полный код программы приведён в Приложении В.

Плюсы Unirest:

* быстродействие.

Минусы Unirest:

* поиск нужных запросов.

# Реализация на языке C#

В ходе работы над проектом были реализованы следующие способы парсинга контента сайтов на языке C#: HtmlAgilityPack, Fizzler, AngleSharp, CsQuery, RestSharp, Selenium.

## Описание библиотеки HtmlAgilityPack

HtmlAgilityPack – это HTML-анализатор, написанный на C# для чтения и записи DOM, и поддерживает обычный XPATH или XSLT – eXtensible Stylesheet Language Transformations [13][14][15].

Для парсинга сайта с помощью парсера HtmlAgilityPack были использованы следующие методы:

* SelectNodes(string xpath) – в качестве входного аргумента требует строку в виде XPath выражения. Возвращает HtmlAgilityPack.HtmlNodeCollection, которая содержит коллекцию узлов, соответствующих запросу XPath выражения, или ноль, если ни один узел не соответствует XPath выражению. Метод выбирает список узлов, соответствующих XPath выражению;
* SelectSingleNode(string xpath) – в качестве входного аргумента требует строку в виде XPath выражения. Возвращает первый HtmlAgilityPack.HtmlNode, соответствующий запросу XPath или нулевой ссылке, если не найдено ни одного соответствующего узла. Метод выбирает первый HtmlNode, соответствующий XPath выражению;
* Load(string url) – в качестве входного аргумента требует строку в виде URL. Возвращает объект класса HtmlDocument, который в дальнейшем обрабатывается. Метод получает HTML-документ из интернет-ресурса;
* GetAttributeValue(string name, string def) – в качестве первого входного аргумента требует строку в виде имени атрибута, второго аргумента – строку в виде значения, которое будет возвращено если не будет найден атрибут. Возвращает строку в виде значения атрибута. Метод ищет совпадение по названию атрибута в старшем элементе. Если совпадений нет, то вернётся второй аргумент.

Помимо методов используется свойство InnerText для извлечения текста между начальным и конечным тегами объекта. Полный код программы приведён в Приложении Г.

Плюсы HtmlAgilityPack:

* работает с XPath выражениями.

Минусы:

* работает только с XPath выражениями.

## Описание библиотеки Fizzler

Fizzler – Библиотека .NET для выбора элементов из дерева узлов на основе css селектора. Реализация по умолчанию основана на HTMLAgilityPack и выбирает из документов HTML [16].

Для парсинга сайта с помощью библиотеки Fizzler были использованы следующие методы:

* Load(string url) – в качестве входного аргумента требует строку в виде URL. Возвращает объект класса HtmlDocument, который в дальнейшем обрабатывается. Метод получает HTML-документ из интернет-ресурса;
* QuerySelector(string selector) – в качестве входного аргумента требует строку в виде css селектора. Возвращает элемент типа HtmlNode. Метод осуществляет поиск элемента по селектору.
* QuerySelectorAll(string selector) – в качестве входного аргумента требует строку в виде css селектора. Возвращает список элементов типа HtmlNode. Метод осуществляет поиск элементов по селектору.
* GetAttributeValue(string name, string def) – в качестве первого входного аргумента требует строку в виде имени атрибута, второго аргумента – строку в виде значения, которое будет возвращено если не будет найден атрибут. Возвращает строку в виде значения атрибута. Метод ищет совпадение по названию атрибута в старшем элементе. Если совпадений нет, то вернётся второй аргумент.

Помимо методов используется свойство InnerText для извлечения текста между начальным и конечным тегами объекта. Полный код программы приведён в Приложении Д.

Плюсы Fizzler:

* имеет все те же методы, что и HtmlAgilitypack;
* работает с css селекторами.

Минусы Fizzler:

* работает только с XPath выражениями и css селекторами.

## Описание библиотеки AngleSharp

AngleSharp – это библиотека .NET, которая позволяет анализировать гипертексты такие как HTML, SVG и MathML. XML также поддерживается библиотекой. Важным аспектом AngleSharp является то, что CSS также может быть проанализирован. Включенный анализатор построен на основе официальной спецификации W3C – World Wide Web Consortium. Также присутствуют стандартные функции DOM, такие как querySelector или querySelectorAll [17].

Для парсинга сайта с помощью библиотеки AngleSharp были использованы следующие методы:

* ParseDocument(string source) – в качестве входного аргумента требует строку в виде html. Метод парсит переданный аргумент и возвращает результат.
* GetElementsByClassName(string classNames) – в качестве входного аргумента требует строку в виде имени класса. Метод возвращает список уникальных элементов, которые имели класс с именем, которое было передано в виде строки.
* GetElementsByTagName(string tagName) – в качестве входного аргумента требует строку в виде имени тэга. Метод возвращает список уникальных элементов, которые имели тэг с именем, которое было передано в виде строки.
* GetAttribute(string name) – в качестве входного аргумента требует строку в виде имени атрибута. Метод возвращает значение искомого атрибута, либо null, если атрибут не найден.
* QuerySelector (string selectors) – в качестве входного аргумента требует строку в виде css селектора. Возвращает элемент типа IElement. Метод осуществляет поиск элемента по селектору.
* QuerySelectorAll(string selector) – в качестве входного аргумента требует строку в виде css селектора. Возвращает список элементов типа IElement. Метод осуществляет поиск элементов по селектору.

Помимо методов используется свойство TextContext для извлечения текста между начальным и конечным тегами объекта и свойство InnerHtml для извлечения html кода элемента.

Для получения HTML используется класс WebClient, с помощью которого происходит подключение к страницам сайта.

1. WebClient webClient = new WebClient
2. {
3. Encoding = Encoding.UTF8
4. };
5. string page = "https://letyshops.com/shops?page=" + i;
6. string html = webClient.DownloadString(page);
7. HtmlParser parser = new HtmlParser();
8. var result = parser.ParseDocument(html).GetElementsByClassName("b-teaser");

Полный код программы приведён в Приложении Ж.

Плюсы AngleSharp:

* при отсутствии элемента возвращает null, а не выбрасывает ошибку.
* работает с css селекторами

Минусы AngleShrap:

* выбор по локаторам возвращает только коллекцию

## Описание библиотеки CsQuery

CsQuery – это порт jQuery для .NET 4. Он реализует все css селекторы CSS2 и CSS3, все методы манипулирования DOM в jQuery и некоторые служебные методы [18].

Для парсинга сайта с помощью CsQuery были использованы следующие методы:

* CreateFromUrl(string url) – в качестве входного аргумента требует строку в виде URL. Возвращает HTML. Метод создаёт новое DOM дерево из файла html.
* GetAttribute(string name) – в качестве входного аргумента требует строку в виде имени атрибута. Метод ищет атрибут с заданным именем и возвращает его значение.
* CreateDocument(string html) – в качестве входного аргумента требует строку в виде html. Метод создаёт новое DOM дерево из переданного html.

Помимо методов используется свойство Attributes для создания коллекции из атрибутов текущего элемента и индексатор CQ this[string selector] { get; } для доступа к элементам через css селектор. Полный код программы приведён в Приложении И.

Плюсы CsQuery:

* есть возможность парсинга json;
* работает с css селекторами;
* подключение к сайту осуществляется без дополнительных библиотек.

Минусы CsQuery:

* работает только с css селекторами;
* для получения значения атрибута требуется создать список всех атрибутов.

## Описание библиотеки RestSharp

RestSharp – простой REST и HTTP API клиент для .NET. С помощью него осуществляется построение запросов на сервер [19].

RestSharp является аналогичной библиотекой для создания запросов, как Описание библиотеки Unirest в языке программирования Java и Request в языке программирования Python.

Сначала надо создать объект класса RestClient – client, передав в качестве параметра строку в виде URL. Далее создаётся запрос как объект класса RestRequest – request, передав в качестве параметра метод запроса, в данном случае POST. Заголовки и параметры передаются с помощью методов AddHeader() и AddParameter(). Запрос выполняется после того, как вызовется метод класса RestClient Execute(), в который передаётся request. Метод вернёт ответ с сервера в виде переменной типа IRestResponse, из которой в дальнейшем можно извлечь информацию.

Чтобы передать несколько заголовков можно создать Dictionary из значений, либо передавать их последовательно.

Следующий код показывает, как с помощью запросов программа получает json с сайта [Kopikot.ru](https://www.kopikot.ru/), который в последствии обрабатывается.

Пример с использование Dictionary:

1. string url = $"https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_{i}\_order\_popularity.json";
2. var client = new RestClient(url);
3. var request = new RestRequest(Method.GET);
4. Dictionary<String, String> headers = new Dictionary<string, string>()
5. {
6. { "cache-control", "no-cache"},
7. { "Connection", "keep-alive" },
8. { "accept-encoding", "gzip, deflate"},
9. { "Host", "d289b99uqa0t82.cloudfront.net"},
10. { "Postman-Token", "048aef15-143b-4f61-8c44-60467f64a33d,e85413f5-28a6-4878-b792-942c640071cc" },
11. { "Cache-Control", "no-cache"},
12. { "Accept", "\*/\*"},
13. { "User-Agent", "PostmanRuntime/7.11.0"},
14. };
15. foreach (var item in headers)
16. {
17. request.AddHeader(item.Key , item.Value);
18. }
19. IRestResponse response = client.Execute(request);

Пример без использования Map:

1. string url = $"https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_{i}\_order\_popularity.json";
2. var client = new RestClient(url);
3. var request = new RestRequest(Method.GET);
4. request.AddHeader("cache-control", "no-cache");
5. request.AddHeader("Connection", "keep-alive");
6. request.AddHeader("accept-encoding", "gzip, deflate");
7. request.AddHeader("Host", "d289b99uqa0t82.cloudfront.net");
8. request.AddHeader("Postman-Token", "048aef15-143b-4f61-8c44-60467f64a33d,e85413f5-28a6-4878-b792-942c640071cc");
9. request.AddHeader("Cache-Control", "no-cache");
10. request.AddHeader("Accept", "\*/\*");
11. request.AddHeader("User-Agent", "PostmanRuntime/7.11.0");
12. IRestResponse response = client.Execute(request);

Полный код программы приведён в Приложении К.

Также, как и с Unirest, для упрощения построения запроса используется бесплатная программа Postman. Использовались аналогичные параметры и заголовки. Для языка C# в Postman имеется только RestSharp. Пример кода из Postman для сайта Cash4Brands изображён на рисунке 10.

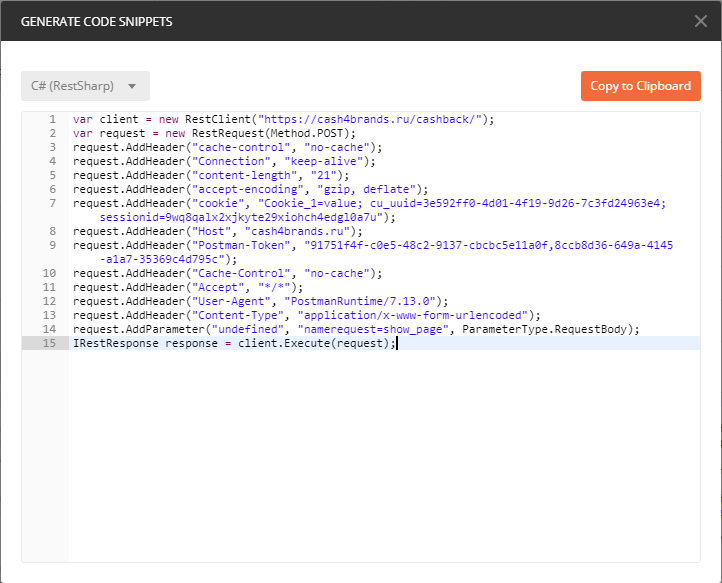


Рисунок 10 – генерация RestSharp кода в Postman

После получения JSON с запроса, он обрабатывается с помощью парсера для JSON. Разбор JSON осуществляется с помощью библиотеки Json.NET. Для примера взят код для обработки данных с Kopikot.ru [20].

Для начала работы с парсером JSON нужно создать переменную тпиа JObject, используя статический метод класса JObject Parse(string json), который принимает строку в виде json. Ниже приведён пример извлечения из ответа с сервера json.

JObject jsonParse = JObject.Parse(response.Content);

Переход по элементам осуществляется с помощью индексатора JToken this[object key]. В качестве переменной выступает строка, в которой содержится имя тэга. Ниже приведён примеры взятия всех дочерних элементов тэга и взятия отдельного элемента в виде валюты кэшбэка.

Пример 1:

1. var listOfItems = jsonParse["items"];

Пример 2:

1. token["commission"]["max"]["unit"].ToString();

Плюсы RestSharp:

* быстродействие.

Минусы RestSharp:

* поиск нужных запросов.

## Описание инструмента Selenium

Для работы с Selenium на платформе .Net не нужно скачивать вебдрайвер, как в случаях с Java и Python. Но нужно подключить библиотеку selenium для взаимодействия с браузером.

Для парсинга сайта с помощью Selenium были использованы следующие методы:

* Navigate().GoToUrl(String URL) - в качестве входного аргумента требует строку в виде URL. С помощью этого метода происходит навигация по URL. Используя Navigate(), можно использовать такие дополнительные методы, как Back(), Forward(), Refresh();
* FindElement(By by) – в качестве параметра принимает класс By, который задает критерий поиска. Возвращает элемент типа IWebElement, который соответствует заданному критерию;
* FindElements(By by) - в качестве параметра принимает класс By, который задает критерий поиска. Возвращает список всех элементов типа IWebElement, которые соответствуют заданному критерию;
* By – представляет из себя локатор, который уникально идентифицирует элемент страницы. Selenium предоставляет несколько способов использования локаторов для поиска элементов:

1. By.Id – в качестве локатора используется атрибут id элемента страницы;
2. By.Name – в качестве локатора используется атрибут name элемента страницы;
3. By.Xpath – используется для поиска элемента по XPath выражению;
4. By.TagName – поиск по имени HTML тега;
5. By.ClassName – поиск по классу элемента;
6. By.CssSelector – используется для поиска элемента по css селектору;
7. By.LinkText – поиск ссылки с указанным текстом. Текст ссылки должен быть точным совпадением указанного текста;
8. By. PartialLinkText – поиск ссылки по части с указанным текстом.

* GetAttribute(String s) –в качестве параметра принимает строку в виде наименования атрибута. Возвращает текстовое представление найденного атрибута. Для получения текста внутри тэга нужно взять атрибут от «innerHTML».

Помимо методов для поиска элементов используются такой метод, как Click(), который позволяет взаимодействовать с активными объектами такими как кнопки и ссылки и свойство Displayed, которое возвращает true если элемент на экране [7][8][9][21]. Полный код программы приведён в Приложении Л.

При построении логики парсинга сайта использовалась панель разработчика в браузере Google Chrome, чтобы не писать css селекторы или XPath самостоятельно, как в примере с Jsoup. Использование возможностей панели разработчика в браузере Google Chrome изображено на рисунке 1.

Плюсы Selenium:

* имитация пользователя;
* методы для поиска как одного элемента, так и множества;
* поиск может осуществляться множеством локаторов;
* не нужно скачивать дополнительные файлы, всё скачивается с помощью NuGet.

Минусы Selenium:

* использует реальный браузер для работы, из-за чего скорость падает.

# Реализация на языке Python

В ходе работы над проектом были реализованы следующие способы парсинга контента сайтов на языке Python: BeautifulSoup, Request, Scrapy.

## Описание библиотеки BeautifulSoup

Библиотека BeautifulSoup является самой популярной библиотекой для веб-скрапинга. Кроме того, эта библиотека является самой доступной для понимания и для работы.

Своё имя библиотека BeautifulSoup получила в честь одноимённого стихотворения Черепахи Квази из «Приключения Алисы в Стране чудес».

Так как BeutifulSoup не может самостоятельно подключаться к сайтам и считывать html, для работы с веб-скрапингом нужно из библиотеки urllib.request импортировать метод urlopen. Этот метод создает файлоподобный объект, который читается методом read(). Объект в виде html передаётся в конструктор BeautifulSoup для создания объекта класса, с которым можно будет в дальнейшем работать.

BeautifulSoup может использовать различные HTML-парсеры , каждый из которых имеет свои преимущества и недостатки. В данном примере будет использован парсер lxml в качестве анализатора [22][23].

Для парсинга сайта с помощью BeautifulSoup были использованы следующие методы:

1. findAll(name=None, attrs={}, recursive=True, text=None, limit=None,\*\*kwargs) – метод для поиска элементов. Он извлекает список объектов Tag, соответствующих заданному критерию.

Аргумент name является наиболее важным. Этот аргумент ограничивает набор имён тегов. Аргумент kwargs налагает ограничения на атрибуты тега. Также, как и с атрибутом name, можно передать различные значения. Аргумент text позволяет находить объекты Tag. Его значением может быть строка или регулярное выражение, или список, или словарь, или специальные значения: True и None, или вызываемый объект. Логический аргумент recursive, который по умолчанию равен True, сообщает BeautifulSoup о том, нужно ли обходить всё поддерево или искать лишь среди непосредственных потомков объекта Tag или объекта парсера.

Ниже приведён пример использования метода find\_all().

shops = soup.find\_all('div', class\_='b-teaser')

В результате в переменной shops будут элементы, которые соответствуют заданным критериям, то есть тэг «div» с классом «b-teaser».

1. find(name, attrs, recursive, text, \*\*kwargs) – метод для поиска элемента. Он возвращает только первый дочерний элемент этого тега, соответствующий заданному критерию.

Ниже приведён пример использования метода find().

label = shop.find('span', class\_='b-shop-teaser\_\_label ')

В результате в переменной label будет элемент, которые соответствует заданному критерию, то есть тэг «span» с классом «b-shop-teaser\_\_label ».

В качестве именованного аргумента нельзя использовать зарезервированные слова, такие как name, for и так далее. Чтобы решить эту проблему, можно использовать attrs, если необходимо наложить ограничения на атрибуты, имена которых совпадают с зарезервированными словами, либо добавлять нижнее подчёркивание, как приведено в примерах.

get(name=None) – в качестве входного аргумента требует строку в виде имени аргумента. Возвращает значение атрибута. Если такого атрибута нет, то выбрасывает ошибка: AttributeError: 'NoneType' object has no attribute 'get'. Это метод принадлежит библиотеке lxml.

Ниже приведён пример использования метода get().

1. image = shop.find('div', class\_='b-teaser\_\_cover').find('img').get('src')

В результате в переменной image будет ссылка на картинку магазина. Полный код программы приведён в Приложении М.

Плюсы BeautifulSoup:

* может использовать другие парсеры.

Минусы BeautifulSoup:

* не работает с XPath выражениями и css селекторами, но эта проблема решается добавлением lxml.

## Описание фреймворка Scrapy

Один из видов программ, которые написаны специально для скрапинга, называются Пауками. Для них существует отдельный фреймворк Scrapy.

Scrapy – один из наиболее популярных и производительных фреймворков Python для получения данных с веб-страниц, которая включает в себя большинство общих функциональных возможностей. Это значит, что многие функции уже заложены в Scrapy [24].

Фреймворк Scrapy можно установить в PyCharm через настройки.

Для создания проекта в консоли нужно прописать «scrapy startproject project\_name», где вместо project\_name нужно написать имя проекта. После этого в директории создастся проект с определённой файловой структурой следующего типа:

orphanage/

scrapy.cfg

orphanage/

\_\_init\_\_.py

items.py

pipelines.py

settings.py

spiders/

\_\_init\_\_.py

...

В данной файловой структуре имеются следующие элементы:

scrapy.cfg – настройки проекта;

orphanage/ – Python модуль проекта;

orphanage/items.py – классы, описывающие модель собираемых данных;

orphanage/pipelines.py – используется в основном для описания пользовательских форматов сохранения результатов парсинга;

orphanage/settings.py – пользовательские настройки паука;

orphanage/spiders/ – директория, в которой хранятся файлы с классами пауков. Каждого паука принято писать в отдельном файле.

В файл items.py описываются атрибуты, которые будут изыматься с сайтов. Например,

1. class Shop(scrapy.Item):
2. # define the fields for your item here like:
3. name = scrapy.Field()
4. url = scrapy.Field()
5. discount = scrapy.Field()
6. label = scrapy.Field()
7. image = scrapy.Field()

В этом примере описываются аргументы, которые будут соответствовать объектам, находящиеся на сайте.

В папке spyders хранится класс с самим пауком. Для примера, ниже приводится класс паука

1. class ArcadySpider(scrapy.Spider):
2. name = "arcady"
3. address = "https://letyshops.com/shops?page="
4. clear\_address = 'https://letyshops.com'
5. allowed\_domains = ['https://letyshops.com']
6. start\_urls = []
7. max\_page = get\_max\_page()
8. for i in range(1, max\_page + 1):
9. start\_urls.append(address + i.\_\_str\_\_())
10. rules = (
11. Rule(LinkExtractor(allow=('')), callback="parse", follow=False)
12. )
13. def parse(self, response):
14. shops = response.xpath('//div[@class="b-teaser"]')
15. for i, shop in enumerate(shops):
16. item = {
17. 'name': self.get\_name(shop, i),
18. 'discount': self.get\_discount(shop, i),
19. 'label': self.get\_label(shop, i),
20. 'image': self.get\_image(shop, i),
21. 'url': self.get\_url(shop, i)
22. }
23. items.append(item)
24. return items

Основным методом является parse(). Он отвечает за обращение к нужным url и вызов определённых методов. Также parse() является методом по умолчанию.

В переменную start\_urls заносятся URL’ы, которые будут обходиться пауком.

Переменная rules – список правил обхода ресурса. Правило передаёт ссылки из start\_urls методу parse. Аргумент callback отвечает за метод, которому будут переданы данные. Аргумент follow отвечает за извлечение ссылок [25].

Запуск осуществляется с помощью консольной команды «scrapy crawl [имя паука], либо с помощью создания объекта класса CrawlerProcess, с последующим вызовом метода crawl(), в который нужно передать имя класса, в котором реализован паук. После этого запустить методом start(). Ниже представлен пример кода.

1. process = CrawlerProcess()
2. process.crawl(ArcadySpider)
3. process.start()

Для того, чтобы данные записывались в нужный файл и с нужным форматом надо дописать специальные команды: «scrapy crawl [имя паука] -o scraped\_data.csv -t csv». В этом примере данные будут сохраняться в файле с именем «scraped\_data.csv» в формате csv.

В Scrapy используется собственный механизм извлечения данных, который основан так же, как и lxml на libxml2, из HTML-документов – селекторы или selectors. Фактически, селекторы – это отдельные классы, при создании экземпляров которых, на вход передаётся объект класса Response, представляющий собой ответ сервера [26][27][28]. Полный код программы приведён в Приложении Н.

Плюсы Scrapy:

* имеет множество настроек, например, переход по ссылкам и ограничение домена;
* работает с XPath выражениями и css селекторами;
* вывод результатов в файлы.

Минусы Scrapy:

* создание приложения из консоли;
* не может искать файлы по локаторам.

## Описание библиотеки Request

Библиотека Request – библиотека для выполнения запросов к серверу и обработки ответов на языке программирования Python. Данная библиотека является основной для веб-скрапинга страниц сайтов. Пользуясь данной библиотекой, можно получить содержимое страницы в виде html-кода для дальнейшего веб-скрапинга [29].

Request является аналогичной библиотекой для создания запросов, как Описание библиотеки Unirest в языке программирования Java и RestSharp в языке программирования C#

Ниже приведён пример запроса с помощью Request на Python.

1. def get\_json(self, i):
2. url= "https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_" + str(
   1. + "\_order\_popularity.json"
3. payload = ""
4. headers = {
5. 'User-Agent': "PostmanRuntime/7.11.0",
6. 'Accept': "\*/\*",
7. 'Cache-Control': "no-cache",
   1. 'Postman-Token': "b6eeb7b4-63dd-454e-b213-6d2d62b74946,1e851911-db3b-4406-88de-5ffc9ecbaa5d",
8. 'Host': "d289b99uqa0t82.cloudfront.net",
9. 'accept-encoding': "gzip, deflate",
10. 'Connection': "keep-alive",
11. 'cache-control': "no-cache"
12. }
13. response = requests.request("GET", url, data=payload, headers=headers)
14. data = json.loads(response.text)
15. items = data["items"]
16. return items

Заголовки в HTTP-запрос добавляются по типу словаря в headers. В payload хранятся параметры, которые перечисляются через знак амперсанд «&», либо, как и headers, то есть в виде словаря. Пример с параметрами через амперсанд изображён на рисунке 11.

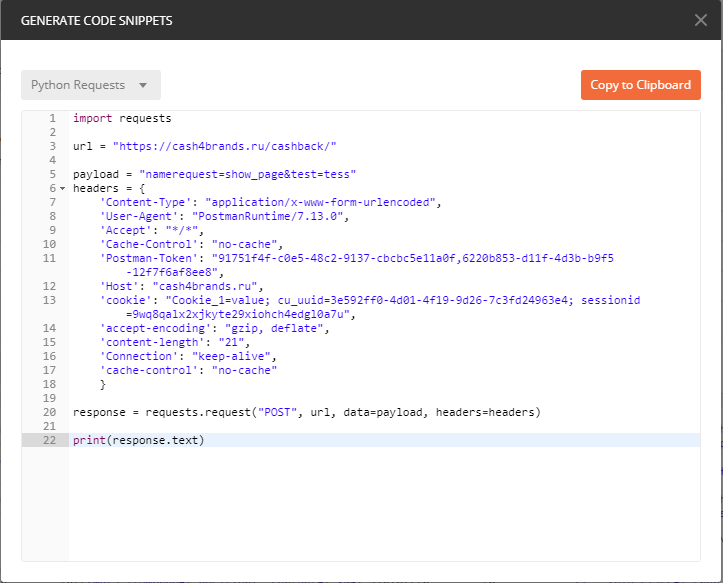


Рисунок 11 – генерация Request кода в Postman

Для отправки запроса нужно вызвать метод request(), в который передаются имя запроса – «GET», адрес сайта – url, параметры – payload и заголовки – headers.

После получения запроса, он обрабатывается с помощью встроенной библиотеки json. Ниже приведён пример работы библиотеки для парсинга json.

1. data = json.loads(response.text)
2. items = data["items"]

В примере видно, что json загружается с помощью метода loads, в который передаётся результат запроса в виде текста. Чтобы получить данные, нужно обратиться к тэгу, как к ключу. Аналогичные действия надо сделать, используя библиотеку Json NET [30]. Полный код программы приведён в Приложении П.

Плюсы Request:

* быстродействие;
* удобное добавление заголовков и параметров.

Минус Request:

* поиск нужных запросов.

## Описание инструмента Selenium

Для работы с Selenium нужно скачать вебдрайвер. При создании объекта для работы с Selenium, в качестве аргумента конструктора нужно указать путь к вебдрайверу. Ниже приведён пример.

\_\_file\_path\_to\_Chrome = "E:\Документы\PyCharmProject\Parsing-on-python\chromedriver\_win32\chromedriver.exe"

driver = webdriver.Chrome(self.\_\_file\_path\_to\_Chrome)

В данном проекте использовался chromedriver.exe. Также нужно подключить библиотеку selenium для взаимодействия с браузером.

Для парсинга сайта с помощью Selenium были использованы следующие методы:

* get(url) – в качестве входного аргумента требует строку в виде URL. С помощью этого метода происходит навигация по URL;
* find\_element\_by\_class\_name(className) – в качестве параметра требует строку в виде имени класса. Возвращает список всех элементов, которые соответствуют заданному критерию;
* get\_attribute(attr) – в качестве параметра требует строку в виде имени аргумента. Возвращает значение аргумента, если он найден, либо выбрасывает ошибку. Для получения текста внутри тэга нужно взять атрибут от «innerHTML».

Помимо использованных методов, есть все те методы, которые поддерживает selenium, то есть поиск элементов по локаторам. При этом можно выбрать метод с использованием By, либо без него. Полный код программы приведён в Приложении Р.

При построении логики парсинга сайта использовалась панель разработчика в браузере Google Chrome, чтобы не писать css селекторы или XPath самостоятельно, как в примере с Jsoup. Использование возможностей панели разработчика в браузере Google Chrome изображено на рисунке 1.

Так как Selenium использует реальный браузер, удобнее было бы распараллелить программу для более быстрого парсинга. Для этого на Python есть библиотека multiprocessing. Но возникает проблема, связанная с типом данных найденных элементов. Они являются несереализуемыми из-за чего библиотека multiprocessing не может с ними работать. Для решения данной проблемы пришлось до распараллеливания найти нужные элементы с помощью методов Selenium, но дальше передать их библиотеке BeautifulSoup. При попытке распараллеливания полученных и переконвентированных в другой формат элементов снова возникает ошибка – превышена максимальная глубина рекурсии. Эта ошибка появляется из-за того, что в данных обнаружились некоторые ссылки, которые использовала библиотека multiprocessing. Из-за этой ошибки дальнейшее распараллеливание было не возможным. Есть два способа решить данную проблему.

Первый способ заключается в том, что надо повысить максимальную глубину рекурсии с помощью метода sys.setrecursionlimit, который принимает на вход число. Это число будет максимально допустимой глубиной рекурсии. Но при больших данных этот способ не решает проблему.

Второй способ заключается в том, что надо привести все элементы к строке, а затем передавать их методу, где они снова будут приведены к начальному формату.

Плюсы Selenium:

* имитация пользователя;
* методы для поиска как одного элемента, так и множества;
* поиск может осуществляться множеством локаторов.

Минусы Selenium:

* использует реальный браузер для работы, из-за чего скорость падает;
* необходимо скачивать дополнительные файлы;
* типы объектов являются несереализуемыми.

# Сравнение скорости реализаций

В главе описывается сравнение скорости выполнения приведённых выше библиотек, фреймворков и инструментов. Сравниваются распараллеленные версии, так как некоторые нераспараллеленные версии работают очень долго.

Сравниваться методы парсинга будут по двум сайтам: MegaBonus и LetyShops.

Для сайта MegaBonus сравниваются реализации Selenium на трёх языках. Так как с Selenium на Python возникла проблема, сравниваться будут реализации на Selenium, совмещенные с ещё одной библиотекой. Для версии на Java и C# будут сравнения с использованием только Selenium.

Для сайта LetyShops сравниваются реализации других приведённых библиотек и фреймворков на трёх языках.

## Сравнение реализаций парсера для сайта MegaBonus

На рисунке 12 изображены замеры времени выполнения работы парсеров на Java и C#, используя только Selenium.

Рисунок 12 – время выполнения Selenium в секундах

На рисунке 13 изображены замеры времени выполнения работы парсеров на Java, C# и Python, используя Selenium и стороннюю библиотеку. Так как на C# много вариантов из чего выбирать, среди них также было сделано сравнение по скорости выполнения. Результаты скорости выполнения парсеров среди библиотек C# на рисунке 14.

Рисунок 13 – время выполнения Selenium и другой библиотеки в секундах

Рисунок 14 – время выполнения парсеров на C# в секундах

Из результатов, приведённых выше, можно сделать вывод, что для использования Selenium вместе с другой библиотекой подходят все приведённые, так как разница между ними незначительная в рамках парсинга.

Если использовать только Selenium, то лучше использовать версию на Java, так как версия на C# работает намного медленнее.

## Сравнение реализаций парсера для сайта LetyShops

На рисунке 15 изображены замеры времени выполнения работы парсеров на Python.

Рисунок 15 – время выполнения парсеров на Python в секундах

На рисунке 16 изображены замеры времени выполнения работы парсеров на Java.

Рисунок 16 – время выполнения парсеров на Java в секундах

На рисунке 14 изображены замеры времени выполнения работы парсеров на C#.

Рассматривая скорость выполнения парсеров, можно сделать вывод о библиотеках для парсинга. Для C# быстрее всего работает Fizzler. Для двух других языков программирования не имеет смысла сравнивать с другими, так как для них выбрана только одна библиотека, которая работает с html кодом.

Среди библиотек и фреймворков, которые работают с запросами, быстрее всех является RestSharp и Unirest.

# Алгоритмы парсинга сайтов

## Алгоритм парсинга сайта MegaBonus

В этой главе описывается алгоритм парсинга сайта MegaBonus. В результате действия алгоритма в списке будут находиться магазины с кэшбэками.

По ходу разбора алгоритма будут описаны нюансы при парсинге данного сайта.

Для разбора алгоритма будет использоваться пример на языке программирования C#, используя инструменты для тестирования Selenium.

Первым шагом для построения алгоритма парсинга сайта является ознакомление с самим сайтом. Сначала надо перейти на страницу, которая отображает предложенные магазины. В данном случае это https://megabonus.com/feed. Страница сайта изображена на рисунке 17. При просмотре данной страницы следует обратить внимание на саму структуру сайта, то есть на расположение магазинов, а также на те элементы, которые позволяют получиться следующие магазины.

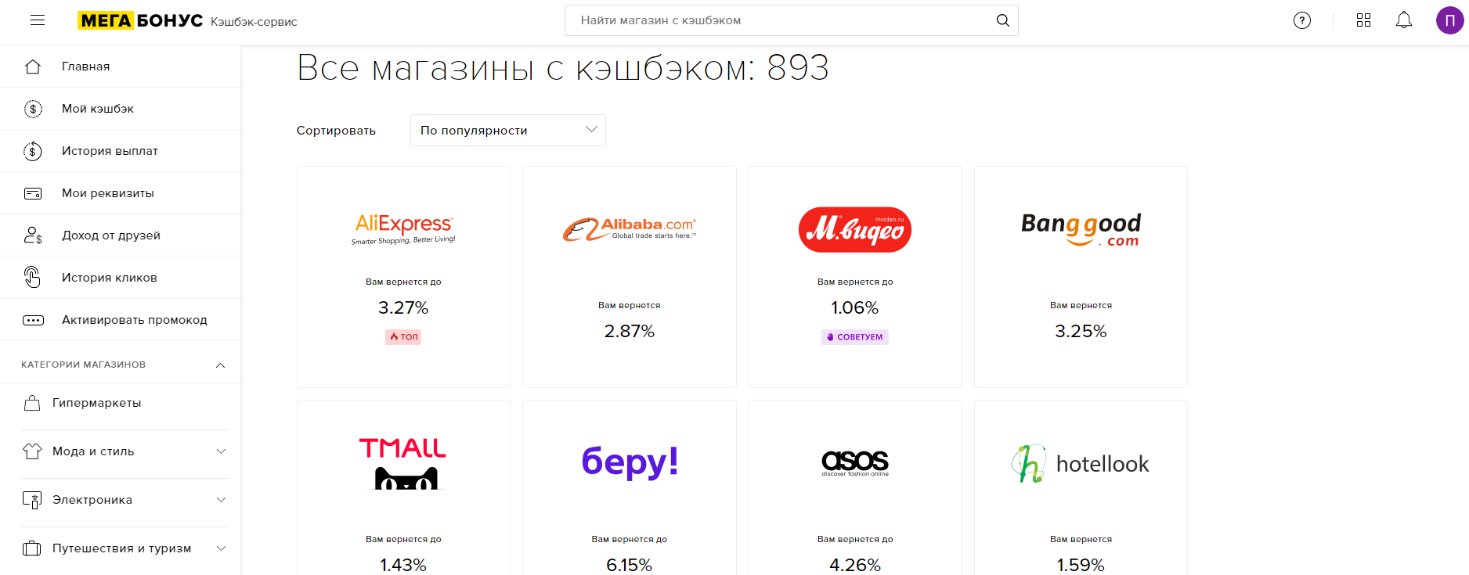


Рисунок 17 – страница с магазинами сайта MegaBonus

После всех магазинов расположен элемент, который позволяет выводить следующие магазины на страницу сайта. Для просмотра структуры страницы в виде html используется панель разработчика. С помощью этой панели осуществляется поиск элементов.

Чтобы добавить все магазины для дальнейшей обработки, нужно активировать приведённый выше элемент до тех пор, пока он не исчезнет с экрана. Для этого нужно найти элемент, который имеет класс «see-more» и активировать его с помощью инструментов Selenium. Пример кода приведён ниже.

1. var button = driver.FindElement(By.ClassName("see-more"));
2. while (button.Displayed)
3. {
4. try
5. {
6. button.Click();
7. button = driver.FindElement(By.ClassName("see-more"));
8. }
9. catch (Exception e)
10. {
11. logger.Info("Кнопка 'Показать ещё' не была найдена. Поиск продолжится " + e);
12. }
13. }

Как видно из примера, после нахождения элемента следует цикл while с проверкой на то, что элемент находится на экране. Внутри блока while вставлен обработчик ошибки, так как если элемент отсутствует, то выбрасывается ошибка. Элемент может не находиться если магазины закончились, либо если сайт не успел прогрузиться, а элемент ещё ищется. Такая обработка событий помогает избежать второй случай.

Если элемент на экране, то он активируется, после чего добавляются магазины на страницу. После активации элемент снова ищется, и так пока не будут показаны все магазины.

Магазины находятся в списке с тэгом «ul» с классом «cacheback-block-list». Элементы данного списка содержат информацию о магазинах в тегах «li». С помощью метода происходит обращение к элементу по классу. После нахождения общего списка, из него извлекаются элементы с информацией. На рисунке 18 изображён выбор конкретного магазина. Пример кода приведён ниже.

1. var ul = driver.FindElement(By.ClassName("cacheback-block-list"));
2. var webElements = ul.FindElements(By.TagName("li"));

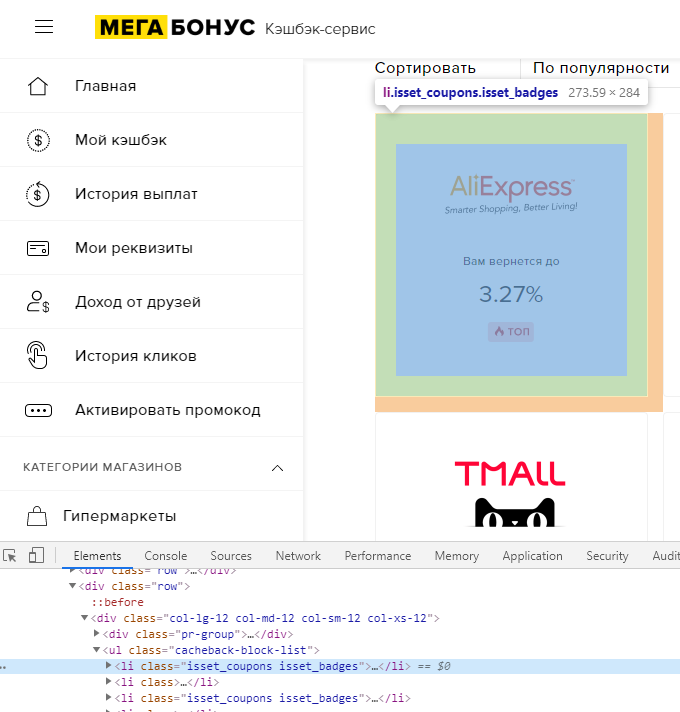


Рисунок 18 - выбор элемента с магазином на MegaBonus

Далее элементы списка webElements передаются на вход методу ParseElements(), который извлекает нужную информацию. Пример кода приведён ниже.

1. private Shop ParseElements(IWebElement item)
2. {
3. var name = GetName(item);
4. var fullDiscount = GetFullDiscount(item);
5. var discount = GetDiscount(fullDiscount);
6. var label = GetLabel(fullDiscount);
7. var image = GetImage(item);
8. var url = GetPage(item);
9. if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(url)))
10. {
11. return new Shop(name, discount, label, image, url);
12. }
13. return null;
14. }

Как видно из примера, для каждого элемента есть свой метод, в котором содержится логика для извлечения информации.

Метод GetName() извлекает из элемента имя, которое находится в элементе с классом «holder-more» в тэге «a». С помощью метода FindElement() и локаторов By осуществляется поиск элемента, соответствующего заданным критериям. С помощью метода GetAttribute() извлекается текст внутри тэга. Так как в извлечённом тексте не будет самого имени, а, например, будет «Подробнее про кэшбэк в Aliexpress», используется регулярное выражение для извлечения имени. Пример кода приведён ниже.

1. private String GetName(IWebElement element)
2. {
3. Regex regex = new Regex("Подробнее про кэшбэк в ([\\w\\s\\d\\W]+)");
4. String name = "";
5. try
6. {
7. name = element.FindElement(By.ClassName("holder-more")).FindElement(By.TagName("a")).GetAttribute("innerHTML");
8. }
9. catch (Exception e)
10. {
11. if (e is NullReferenceException || e is NoSuchElementException)
12. {
13. logger.Error("Произошла ошибка: " + e);
14. return null;
15. }
16. }
17. Match matcher = regex.Match(name);
18. if (matcher.Success)
19. {
20. return matcher.Groups[1].Value;
21. }
22. return null;
23. }

Метод GetFullDiscount () извлекает из элемента значение кэшбэка вместе с валютой, которое находится в дочернем элементе с тэгом «strong» элемента с тэгом «div» и классом «your-percentage». Текст извлекается с помощью свойства Text. Пример кода приведён ниже.

1. private String GetFullDiscount(IWebElement element)
2. {
3. String fullDiscount = "";
4. try
5. {
6. fullDiscount = element.FindElement(By.CssSelector("div.your-percentage > strong")).Text;
7. }
8. catch (Exception e)
9. {
10. if (e is NullReferenceException || e is NoSuchElementException)
11. {
12. logger.Error("Произошла ошибка: " + e);
13. return null;
14. }
15. }
16. return fullDiscount;
17. }

Метод GetDiscount() с помощью регулярного выражения извлекает из строки значение кэшбэка. После извлечения значения кэшбэка, его надо привести к типу double. Если в строке разделителем целой и дробной части является точка, то язык программирования C# выдаёт ошибку, в которой написано, что строка имеет неправильный формат. Для решения этой проблемы нужно заменить точку запятой с помощью метода replace(). Полный код приведён ниже.

1. private Double GetDiscount(String fullDiscount)
2. {
3. if (fullDiscount == null)
4. {
5. logger.Error("Пустой fullDiscount");
6. return Double.NaN;
7. }
8. Regex regex = new Regex("\\d+[.|,]\*\\d\*");
9. String discount = "";
10. Match matcher = regex.Match(fullDiscount);
11. if (matcher.Success)
12. {
13. discount = matcher.Value;
14. }
15. if (Double.TryParse(discount.Replace('.', ','), out double result))
16. {
17. return result;
18. }
19. return Double.NaN;
20. }

Метод GetLabel() с помощью регулярного выражения извлекает из строки валюту кэшбэка. Регулярное выражение содержит валюты, которые могут встретиться во время парсинга. Пример кода приведён ниже.

1. private String GetLabel(String fullDiscount)
2. {
3. if (fullDiscount == null)
4. {
5. logger.Error("Пустой fullDiscount");
6. return null;
7. }
8. Regex regex = new Regex("[$%€]|руб|(р.)|cent|р|Р|RUB|USD|EUR|SEK|UAH|INR|BRL|GBP|CHF|PLN");
9. Match matcher = regex.Match(fullDiscount);
10. if (matcher.Success)
11. {
12. return matcher.Value;
13. }
14. return null;
15. }

Метод GetImage() извлекает из элемента ссылку на картинку, которая находится в элементе с тэгом «img» в атрибуте «src». С помощью метода GetAttribute() извлекается значение атрибута. Пример кода приведен ниже.

1. private String GetImage(IWebElement element)
2. {
3. String image = "";
4. try
5. {
6. image = element.FindElement(By.TagName("img")).GetAttribute("src");
7. }
8. catch (Exception e)
9. {
10. if (e is NullReferenceException || e is NoSuchElementException)
11. {
12. logger.Error("Произошла ошибка: " + e);
13. return null;
14. }
15. }
16. return image;

Метод GetPage() извлекает из элемента ссылку на страницу магазина, которая находится в значении атрибута «href» дочернего элементе с тэгом «a» элемента с тэгом «div» и классом «holder-img». С помощью метода GetAttribute() извлекается значение атрибута. Пример кода приведен ниже.

1. private String GetPage(IWebElement element)
2. {
3. String page = "";
4. try
5. {
6. page = element.FindElement(By.CssSelector("div.holder-img > a")).GetAttribute("href");
7. }
8. catch (Exception e)
9. {
10. if (e is NullReferenceException || e is NoSuchElementException)
11. {
12. logger.Error("Произошла ошибка: " + e);
13. return null;
14. }
15. }
16. return page;
17. }

После получения элементов происходит проверка этих элементов с помощью условного оператора if. Если элементы не пустые и правильные, то они передаются в конструктор для создания объекта класса Shop, который в последствии возвращается. Полный код программы приведён в Приложении Л.

## Алгоритм парсинга сайта LetyShops

В этой главе описывается алгоритм парсинга сайта LetyShops. В результате действия алгоритма в списке будут находиться магазины с кэшбэками.

По ходу разбора алгоритма будут описаны нюансы при парсинге данного сайта.

Для разбора алгоритма будет использоваться пример на языке программирования C#, используя библиотеку Fizzler.

Первым шагом для построения алгоритма парсинга сайта является ознакомление с самим сайтом. Сначала надо перейти на страницу, которая отображает предложенные магазины. В данном случае это <https://letyshops.com/shops>. Страница сайта изображена на рисунке 19. При просмотре данной страницы следует обратить внимание на саму структуру сайта, то есть на расположение магазинов, а также на те элементы, которые позволяют получиться следующие магазины.

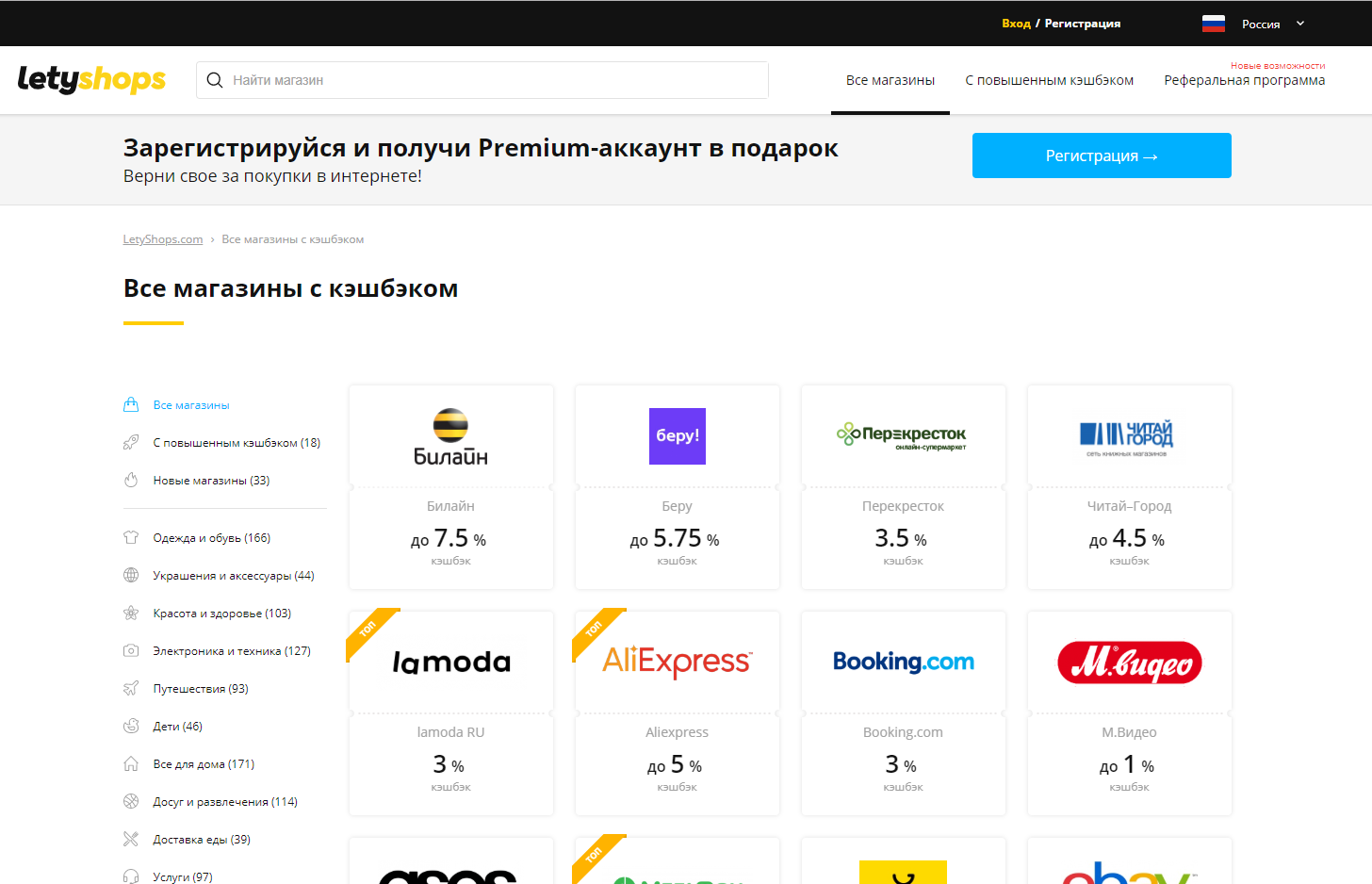


Рисунок 19 – страница с магазинами сайта LetyShops

После всех магазинов расположен элемент, который позволяет переходить по страницам магазина. Элемент изображен на рисунке 20.



Рисунок 20 – элемент страницы для перехода по страницам

После перехода на последующие страницы, URL меняется так, что в него добавляются параметры в виде номера страницы – https://letyshops.com/shops?page=2. Это облегчает создание алгоритма, так как для перехода по сайту потребуется лишь менять один параметр.

Для просмотра структуры страницы в виде html используется панель разработчика. С помощью этой панели осуществляется поиск элементов. На рисунке 21 изображен выбор конкретного магазина.

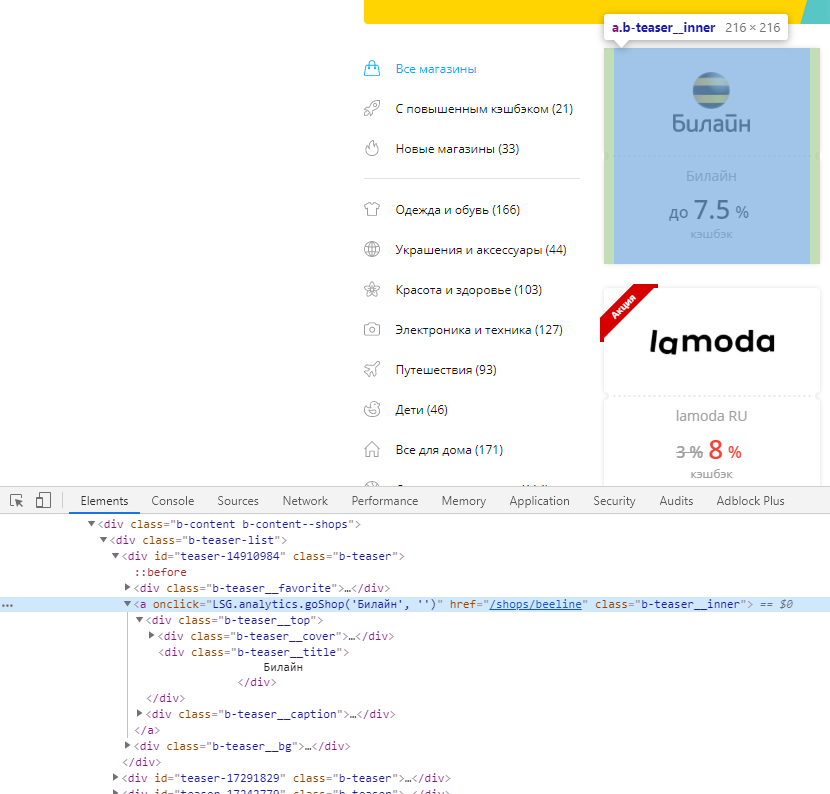


Рисунок 21 – выбор элемента с магазином на LetyShops

Из предыдущего рисунка видно, что элемент находится в тэге «а» с классом «b-teaser\_\_inner». Внутри этого элемента содержится вся нужная информация. То есть нужно извлечь все элементы с приведённым ранее классом. Ниже представлен код для извлечения элементов с одной страницы.

1. var listOfShops = html.QuerySelectorAll("div.b-teaser > a.b-teaser\_\_inner");

Используя css селектор, в html коде находятся все элементы, которые подходят под заданный критерий и записываются в список listOfShops.

Далее с помощью foreach осуществляется проход по списку, и каждый элемент передаётся на вход методам в качестве аргумента. После получения данных из методов, происходит проверка, а затем добавление в список. Ниже представлен код.

1. foreach (var item in listOfShops)
2. {
3. String name = GetName(item);
4. Double discount = GetDiscount(item);
5. String label = GetLabel(item);
6. String url = GetUrl(item);
7. String image = GetImage(item);
8. if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(url)))
9. {
10. Shops.Add(new Shop(name, discount, label, image, url));
11. }
12. }

Как видно из примера, для каждого элемента есть свой метод, в котором содержится логика для извлечения информации.

Метод GetName() извлекает из элемента имя, которое находится в элементе «div» с классом «b-teaser\_\_title». С помощью css селектора происходит перемещение к этому элементу, а с помощью свойства InnerText извлекается текст тэга. Метод Trim() нужен для удаления лишних пробелов. Пример кода приведён ниже.

1. private String GetName(HtmlNode html)
2. {
3. return html.QuerySelector("div.b-teaser\_\_title").InnerText.Trim();
4. }

Метод GetDiscount извлекает из элемента значение кэшбэка, которое может находиться в двух местах в зависимости от того старое значение или новое. Если значение кэшбэка не менялось и не помечалось красным, то оно находится в тэге «span» с классом «b-shop-teaser\_\_cash». Если значение кэшбэка новое и оно помечено красным, как на рисунке 22, то путь к элементу будет уже иным. Элемент находится в тэге «span» с классом «b-shop-teaser\_\_new-cash». Далее с помощью css селектора происходит перемещение к этому элементу, а с помощью свойства InnerText извлекается текст тэга. Метод Trim() нужен для удаления лишних пробелов. После извлечения значения кэшбэка, его надо привести к типу double. Если в строке разделителем целой и дробной части является точка, то язык программирования C# выдаёт ошибку, в которой написано, что строка имеет неправильный формат. Для решения этой проблемы нужно заменить точку запятой с помощью метода replace(). Пример кода приведён ниже.

1. private Double GetDiscount(HtmlNode html)
2. {
3. string discount = "";
4. try
5. {
6. discount = html.QuerySelector("div.b-teaser\_\_caption > div.b-teaser\_\_cashback-rate > div > div > span.b-shop-teaser\_\_cash").InnerText.Trim();
7. }
8. catch (NullReferenceException e)
9. {
10. logger.Info("Вторая попытка взять discount " + e);
11. discount = html.QuerySelector("div.b-teaser\_\_caption > div.b-teaser\_\_cashback-rate > div > div > span.b-shop-teaser\_\_new-cash").InnerText.Trim(); ;
12. }
13. if (Double.TryParse(discount.Replace('.', ','), out double result))
14. {
15. return result;
16. }
17. return Double.NaN;
18. }

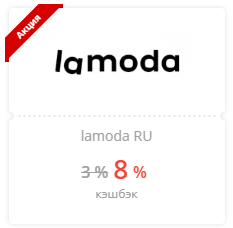


Рисунок 22 – новое значение кэшбэка

Метод GetLabel() извлекает из элемента валюту. На сайте есть три вида кэшбэка по расположению в html коде: первый содержит в себе только значение и валюту – тэг «span» с классом «b-shop-teaser\_\_label»; второй содержит в себе слово «до», который расположен в тэге «span» с классом «b-shop-teaser\_\_label», а потом значение и валюту, которая расположена в том же тэге с таким же классом; третий – кэшбэк красного цвета, как изображено на рисунке 22, находится в тэге «span» с классом «b-shop-teaser\_\_label b-shop-teaser\_\_label--red». Для того, чтобы не писать множество проверок было решено с помощью QuerySelectorAll найти все элементы, которые подходили под следующий критерий: тэг «span» с классом «b-shop-teaser\_\_label», и из коллекции брать последний элемент. Далее с помощью css селектора происходит перемещение к этому элементу, а с помощью свойства InnerText извлекается текст тэга. Метод Trim() нужен для удаления лишних пробелов. Пример кода приведён ниже.

1. private String GetLabel(HtmlNode html)
2. {
3. var labelList = html.QuerySelectorAll("div.b-teaser\_\_caption > div.b-teaser\_\_cashback-rate > div > div > span.b-shop-teaser\_\_label");
4. return labelList.Last().InnerText.Trim();
5. }

Метод GetUrl() извлекает из элемента страницу магазина на сайте. Ссылка находится в тэге «a» с классом «b-teaser\_\_inner» в атрибуте «href». Так как передаваемый методу элемент как раз совпадают критерию, с помощью метода GetAttributeValue() извлекается значение заданного атрибута. Так как ссылка не полная, в начало добавляется основной URL сайта. Пример кода приведён ниже.

1. private String GetUrl(HtmlNode html)
2. {
3. return addressOfSite + html.GetAttributeValue("href", "");
4. }

Метод GetImage() извлекает из элемента ссылку на картинку магазина на сайте. Ссылка находится в тэге «img», который является дочерним элементом тэга «div» с классом «b-teaser\_\_cover». С помощью css селектора происходит переход к элементу. Далее с помощью метода GetAttributeValue() извлекается значение заданного атрибута. Пример кода приведён ниже.

1. private String GetImage(HtmlNode html)
2. {
3. return html.QuerySelector("div.b-teaser\_\_top > div.b-teaser\_\_cover > img").GetAttributeValue("src", "");
4. }

После получения элементов происходит проверка этих элементов с помощью условного оператора if. Если элементы не пустые и правильные, то они передаются в конструктор для создания объекта класса Shop, который в последствии добавляется в список, где содержатся все магазины.

Все вышеописанные методы обрабатывают одну страницу сайта. Чтобы обработать следующие страницы, потребуется знать точное количество страниц, а потом передавать в параметр числа. Для того, чтобы узнать максимальное количество страниц, есть метод GetMaxPage().

Метод GetMaxPage() извлекает из элемента номер последней страницы сайта. Номер находится тэге «a», который хранится в пятом дочернем элементе списка со всеми номерами страниц. Далее с помощью css селектора происходит перемещение к этому элементу, а с помощью свойства InnerText извлекается текст тэга. Метод Trim() нужен для удаления лишних пробелов. Пример кода приведён ниже.

1. private Int32 GetMaxPage()
2. {
3. var htmlWeb = new HtmlWeb
4. {
5. OverrideEncoding = Encoding.UTF8
6. };
7. var document = htmlWeb.Load("https://letyshops.com/shops?page=1");
8. var html = document.DocumentNode;
9. string maxPage = html.QuerySelector("div.b-content.b-content--shops > ul > li:nth-child(5) > a").InnerText.Trim();
10. if (Int32.TryParse(maxPage, out int result))
11. {
12. return result;
13. }
14. return 30;
15. }

После определения максимального количества страниц, программа может полностью обойти страницы с магазинами на сайте и записать нужную информацию. Пример кода приведён ниже.

1. var maxPage = GetMaxPage();
2. Parallel.For(1, maxPage + 1, ParseElements);

Полный код программы приведён в Приложении Д.

## Решение проблем при парсинге сайтов

При парсинге сайтов могут возникнуть различные проблемы, которые можно решить каким-либо способом. В этой главе будет описание решений некоторых проблем, которые были встречены.

При парсинге сайте программа делает частые запросы к одному и тому же ресурсу. Из-за этого некоторые сайты могут быть защищены с помощью блокировки IP адреса. Для решения данной проблемы можно добавить время ожидания, например, после нажатия на кнопку должно пройти не меньше пяти секунд. Недостаток этого метода в том, что вычислять время приходится на практике. Другим решением является использование прокси-серверов. Например, Selenium может использовать прокси. Пример кода приведён ниже.

1. Proxy proxy = new Proxy();
2. proxy.setHttpProxy(PROXY).setFtpProxy(PROXY).setSslProxy(PROXY).setSslProxy(PROXY);
3. DesiredCapabilities capabilities = DesiredCapabilities.firefox();
4. capabilities.setCapability(CapabilityType.PROXY, proxy);
5. WebDriver driver = new FirefoxDriver(capabilities)

В данном примере используется браузер Firefox, но он также работает для Chrome. В качестве PROXY должна быть строка с IP прокси-сервера. Недостатком этого метода является постоянный поиск новых проски-серверов.

При парсинге сайта, у которого отсутствует или истекла SSL – Secure Sockets Layer лицензия, может возникнуть проблема при подключении. Для решения данной проблемы можно создать хранилище доверенных SSL сертификатов, либо задать специальные параметры для игнорирования SSL сертификатов [35].

При парсинге сайта может потребоваться войти в аккаунт, чтобы получиться доступ к более подробной информации. Для решения этой проблемы можно обратиться к специальному запросу, который отвечает за логирование, либо можно использовать Selenium.

При парсинге сайта следует обращать внимание на то, как выглядит информация после того, как она была извлечена с сайта. Данные могут быть иной кодировки. Данная проблема решается сменой кодировки.

# Описание визуализации результатов

В этой главе описано как визуализировались полученные после парсинга данные.

После получения данных, информация о магазинах добавляется в базу данных MySQL. Используя базу данных, можно перенести информацию в любой файл. Для примера были реализованы три метода визуализации: с помощью MVC, в Excel файл и в CSV файл.

Файлы типа Excel и CSV являются одними из стандартных файлов для записи и чтения данных, поэтому они были использованы.

## Использование MVC

Для отображения собранной информации был использован паттерн MVC, который используется для разработки веб-приложений.

На языке программирования Java использовался фреймворк Spring. Для доступа к данным использовался ORM Hibernate. Результат изображён на рисунке 23.

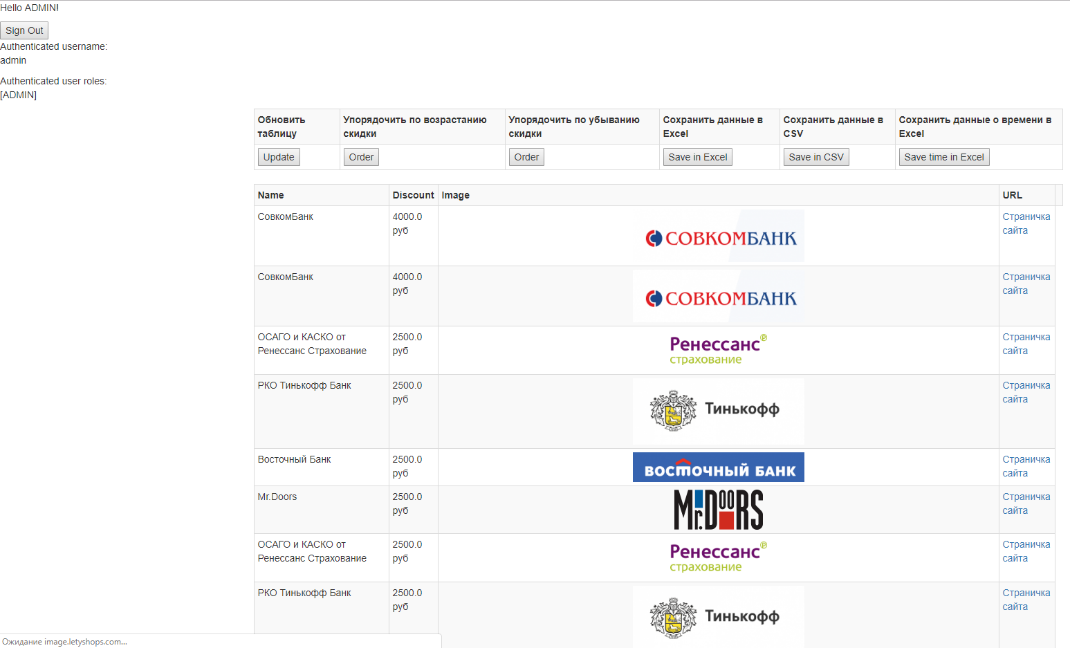


Рисунок 23 – вывод результата на страницу сайта, используя Spring

На языке программирования C# использовалась платформа для разработки веб-приложений – ASP.NET. Для доступа к данным использовался ORM Entity Framework. Результат изображён на рисунке 24.



Рисунок 24 – вывод результата на страницу сайта, используя ASP.NET

На языке программирования Python использовался фреймворк Django. Для доступа к данным использовался внутренний ORM фреймворка. Результат изображён на рисунке 25.



Рисунок 25 – вывод результата на страницу сайта, используя Django

## Вывод в Excel файл

Excel является удобным форматом для чтения данных. Для каждого языка программирования реализована возможность создания Excel файла с данными из базы данных.

Для записи данных в Excel файл на языке программирования Java использовалась библиотека Apache POI [31]. Результат изображён на рисунке 26.

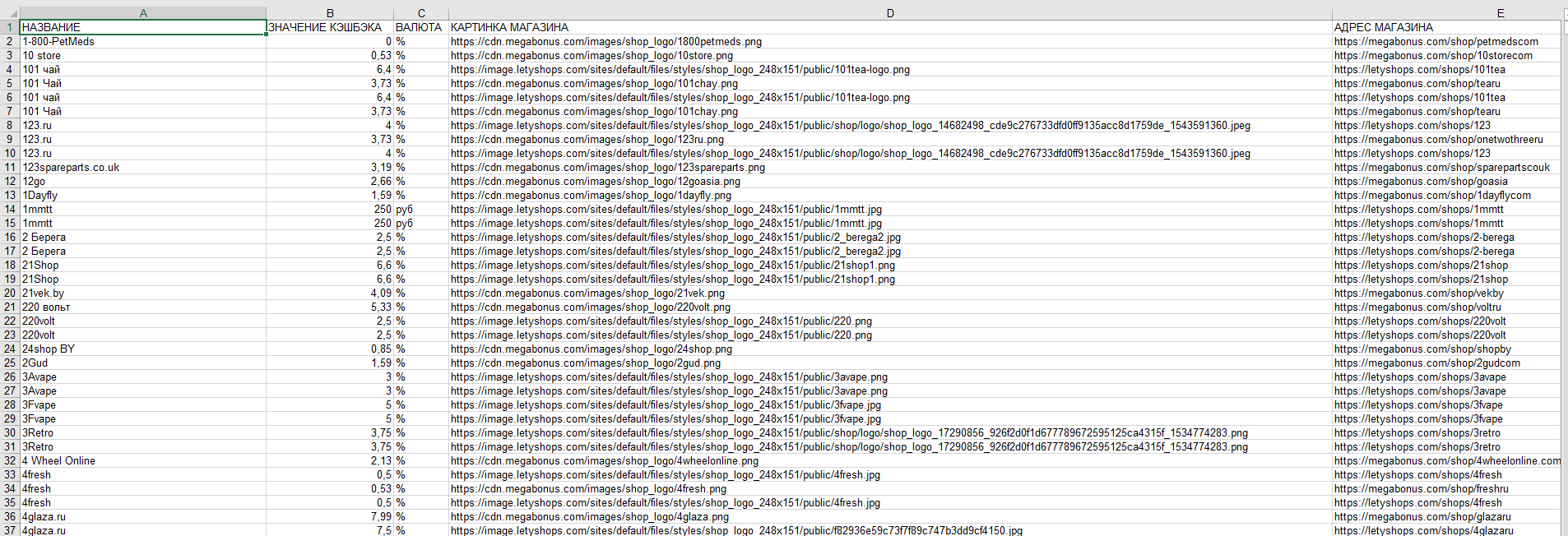


Рисунок 26 – Excel файл на Java

Для записи данных в Excel на языке программирования C# использовалась библиотека OfficeOpenXml [32]. Результат изображён на рисунке 27.

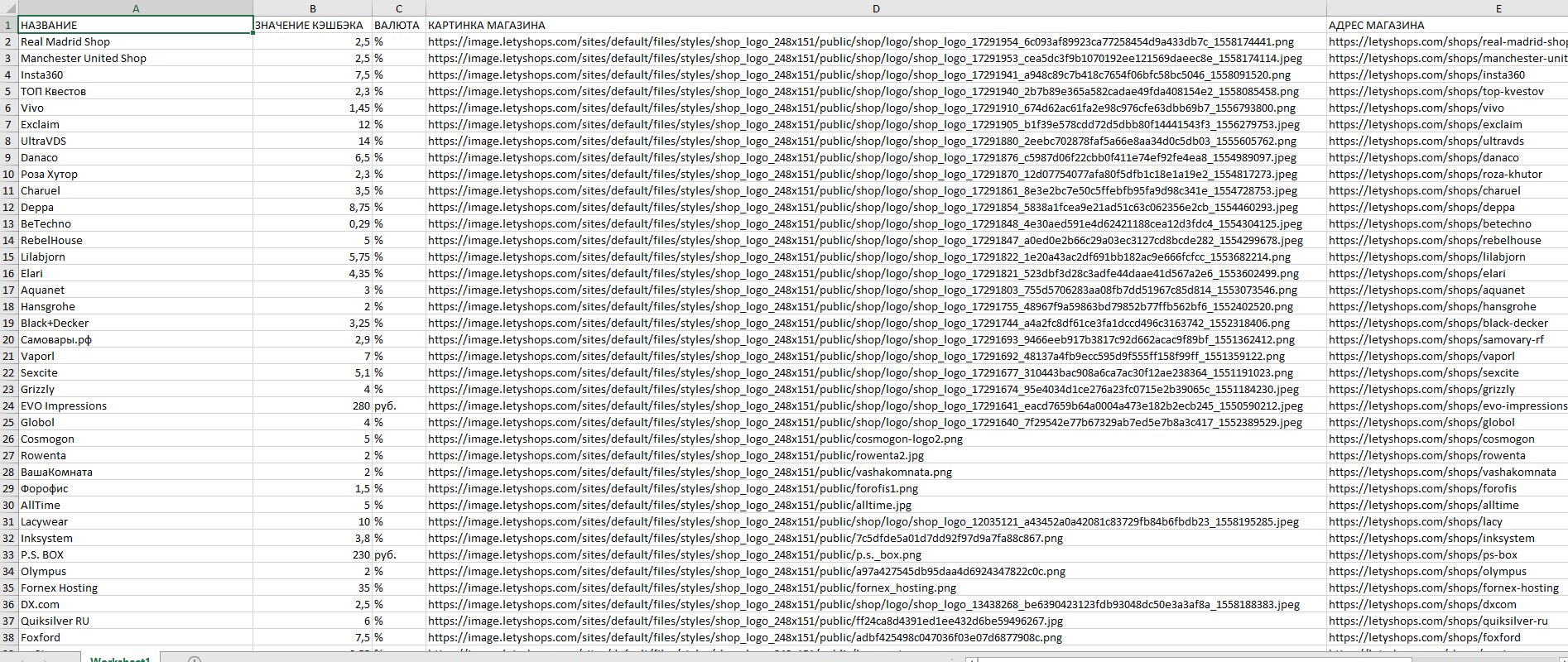


Рисунок 27 – Excel файл на C#

Для записи данных в Excel на языке программирования Django использовалась библиотека openpyxl [33]. Результат изображён на рисунке 28

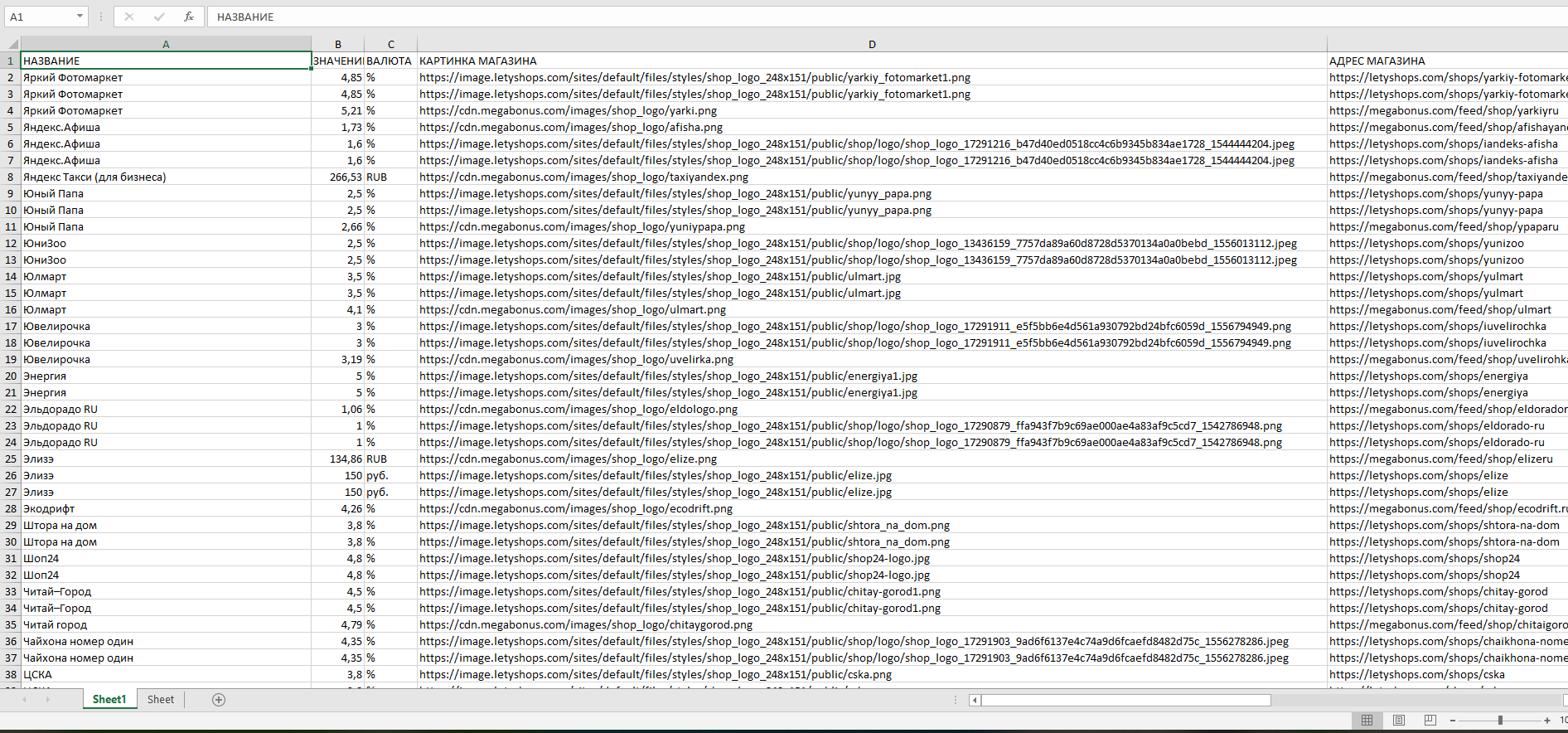


Рисунок 28 – Excel файл на Python

## Вывод в CSV файл

CSV является достаточно распространённым форматом для чтения данных, который можно считать почти любой программой. Для каждого языка программирования реализована возможность создания CSV файла с данными из базы данных.

Для записи данных в CSV файл на языке программирования Java использовалась библиотека opencsv [34]. Результат изображён на рисунке 29.

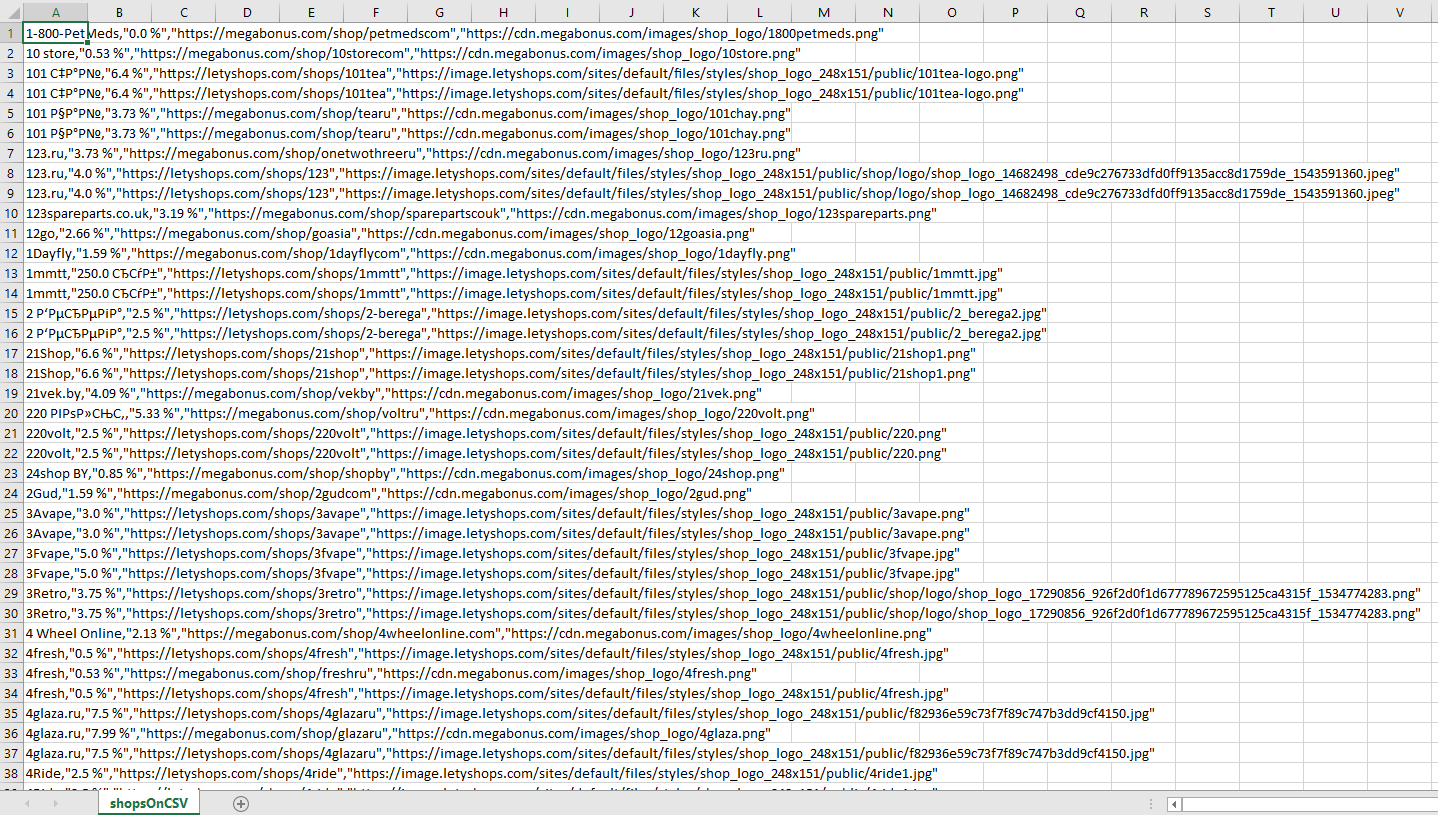


Рисунок 29 – CSV файл на Java

Для записи данных в CSV файл на языке программирования C# использовались стандартные средства в виде StreamWriter. Результат изображён на рисунке 30.

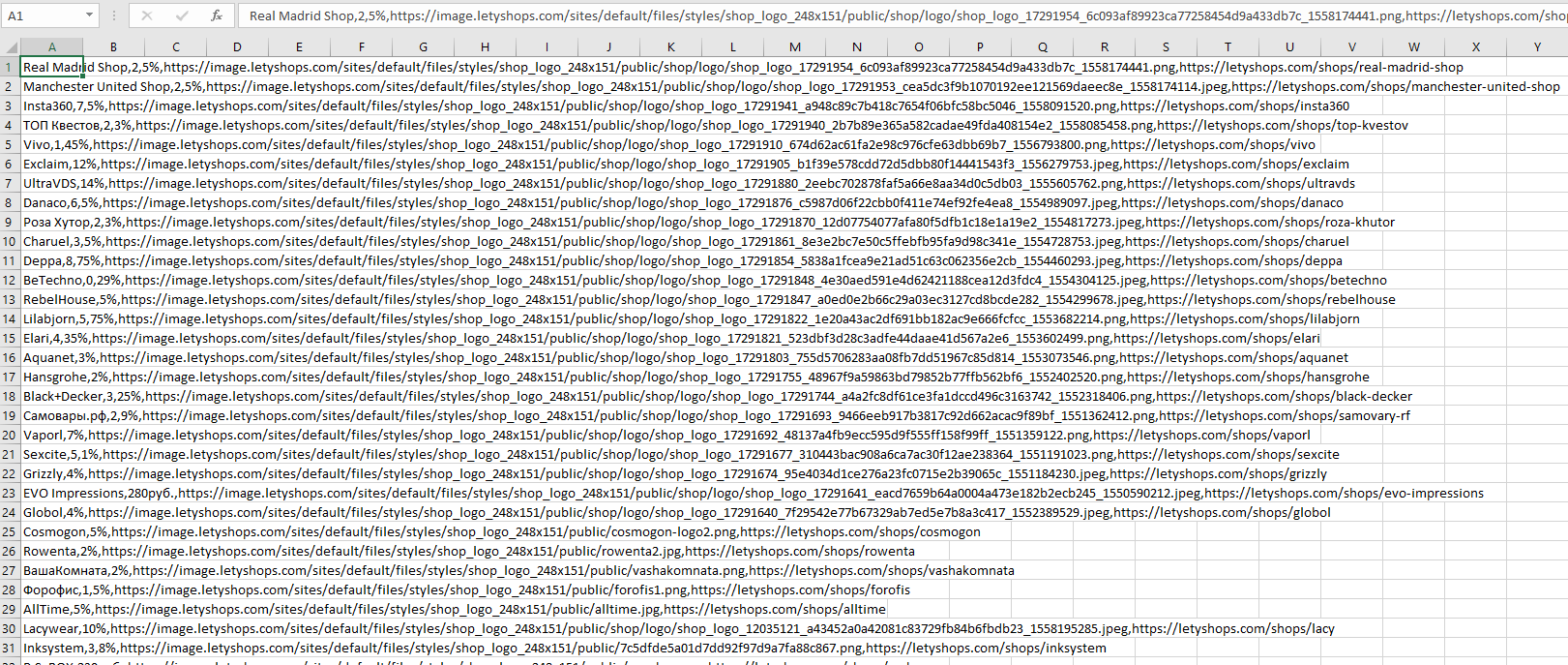


Рисунок 30 – CSV файл на C#

Для записи данных в CSV файл на языке программирования Java использовалась встроенная библиотека csv. Результат изображён на рисунке 31.

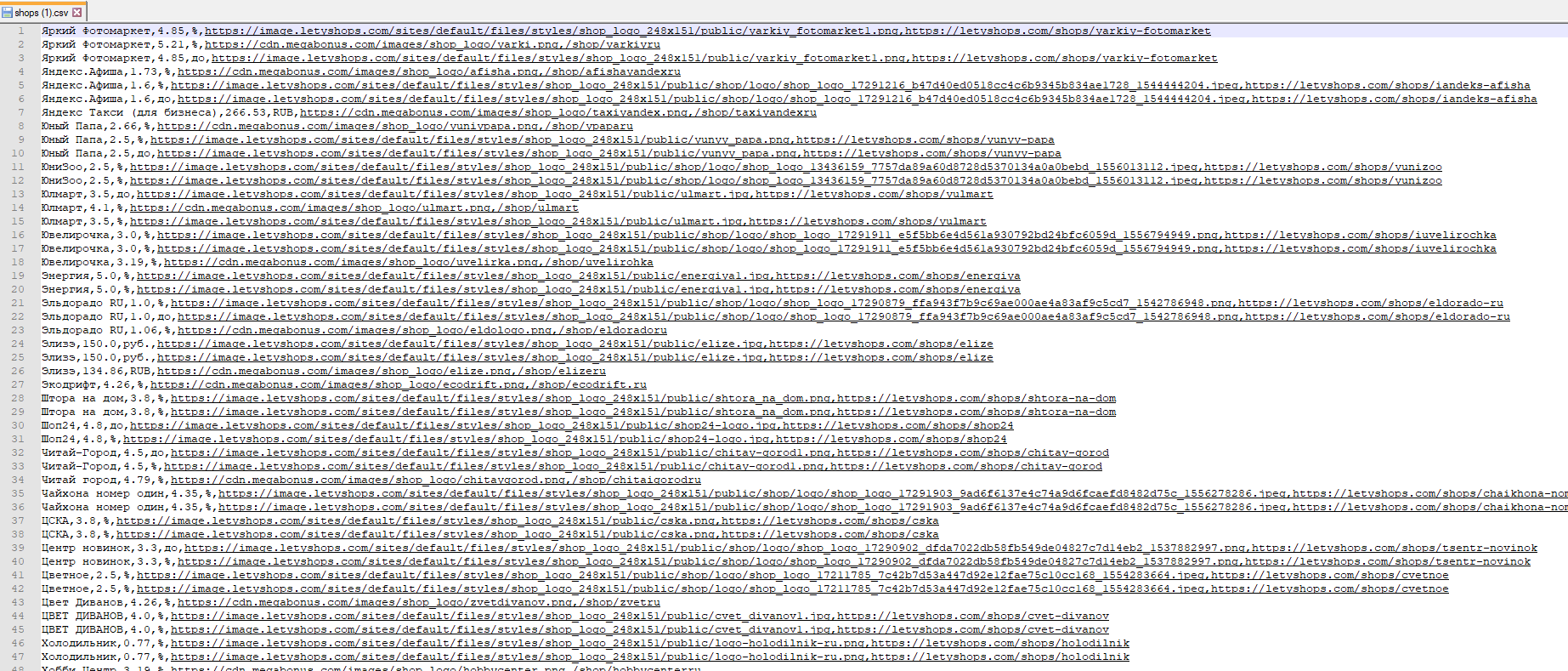


Рисунок 31 – CSV файл на Python

# Заключение

В данной работе были рассмотрены библиотеки для парсинга сайтов на трёх языках программирования и их основной функционал. Помимо этого, были написаны программы, которые, используя библиотеки для парсинга сайтов, собирают и систематизируют данные. Также произведён анализ скорости выполнения различных реализаций библиотек.

В ходе анализа скорости выполнения были сделаны выводы, что для использования Selenium вместе с другой библиотекой подходят все версии Selenium вместе с Jsoup на Java, либо с Fizzler на C#, либо с BeautifulSoup на Python. Если использовать только Selenium, то лучше использовать версию на Java.

Рассматривая скорость выполнения парсеров, которые реализуют библиотеки для париснга, можно сделать вывод, что быстрее всего работают библиотека Fizzler на языке программирования C# и библиотека Jsoup на языке программирования Java.

Среди библиотек и фреймворков, которые работают с запросами, быстрее является RestSharp и Unirest.

В ходе работы были описаны способы решения проблем, возникающие при написании программы и самом парсинге сайтов и способы визуализации данных для лучшего ознакомления с ними.

# Список использованной литературы

1. Meltwater [Электронный ресурс]: сайт. URL: https://www.meltwater.com/?ucs (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
2. Парсинг сайтов: как с точки зрения закона выглядит один из самых полезных ИТ- инструментов по миру (и в России)? [Электронный ресурс]: сайт. URL: https://habr.com/ru/post/340302/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
3. Использование Java JSoup для анализа кода HTML [Электронный ресурс]: сайт. URL: https://o7planning.org/ru/10399/jsoup-java-html-parser-tutorial (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
4. Interface Connection [Электронный ресурс]: документация. URL: https://jsoup.org/apidocs/org/jsoup/Connection.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
5. jsoup: Java HTML Parser [Электронный ресурс]: сайт. URL: https://jsoup.org/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
6. Class Element [Электронный ресурс]: документация. URL: https://jsoup.org/apidocs/org/jsoup/nodes/Element.html#select-java.lang.String- (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
7. How To Navigate A URL Using Selenium Web Driver [Электронный ресурс]: сайт. URL: https://www.oodlestechnologies.com/blogs/How-To-Navigate-A-URL-Using-Selenium-Web-Driver/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
8. Find Element and FindElements in Selenium WebDriver [Электронный ресурс]: сайт. URL: https://www.guru99.com/find-element-selenium.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
9. Selenium WebDriver: поиск элементов на странице [Электронный ресурс]: сайт. URL: http://internetka.in.ua/selenium-webdriver-findelement-by/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
10. Unirest for Java [Электронный ресурс]: сайт. URL: http://unirest.io/java.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
11. Руководство по Unirest [Электронный ресурс]: сайт. URL: https://www.codeflow.site/ru/article/unirest (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
12. Защита от CSRF [Электронный ресурс]: сайт. URL: https://djbook.ru/ch14s05.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
13. Распарсить HTML в .NET и выжить: анализ и сравнение библиотек [Электронный ресурс]: сайт. URL: https://habr.com/ru/post/273807/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
14. Html Agility Pack – удобный .NET парсер HTML [Электронный ресурс]: сайт. URL: https://habr.com/ru/post/112325/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
15. HAP – Documentations [Электронный ресурс]: документация. URL: https://html-agility-pack.net/documentation (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
16. fizzler [Электронный ресурс]: документация. URL: https://code.google.com/archive/p/fizzler/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
17. AngleSharp [Электронный ресурс]: репозиторий. URL: https://github.com/AngleSharp/AngleSharp (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
18. CsQuery [Электронный ресурс]: репозиторий. URL: https://github.com/jamietre/CsQuery (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
19. RestSharp [Электронный ресурс]: сайт. URL: http://restsharp.org/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
20. Json.NET [Электронный ресурс]: сайт. URL: https://www.newtonsoft.com/json (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
21. Introduction [Электронный ресурс]: документация. URL: https://www.seleniumhq.org/docs/01\_introducing\_selenium.jsp (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
22. Beautiful Soup Documentation [Электронный ресурс]: документация. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
23. BeautifulSoup Parser [Электронный ресурс]: сайт. URL: https://lxml.de/elementsoup.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
24. WEB SCRAPING С ПОМОЩЬЮ SCRAPY И PYTHON 3 [Электронный ресурс]: сайт. URL: https://www.8host.com/blog/web-scraping-s-pomoshhyu-scrapy-i-python-3/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
25. Подборка примеров Scrapy LinkExtractor, Rules для последующих практикумов [Электронный ресурс]: сайт. URL: http://pythonr.blogspot.com/2015/01/scrapy-linkextractor-rules.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
26. Scrapy Tutorial [Электронный ресурс]: документация. URL: https://doc.scrapy.org/en/latest/intro/tutorial.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
27. Краулер своими руками. Часть 1 [Электронный ресурс]: сайт. URL: http://pi-code.blogspot.com/2008/12/1.html (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
28. Библиотека для парсинга сайта Scrapy [Электронный ресурс]: сайт. URL: https://tyvik.ru/posts/parsing-scrapy/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
29. Quickstart [Электронный ресурс]: сайт. URL: https://2.python-requests.org//en/master/user/quickstart/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
30. Python: модуль requests [Электронный ресурс]: сайт. URL: https://rtfm.co.ua/python-modul-requests/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
31. Работа с Excel в Java через Apache POI [Электронный ресурс]: сайт. URL: https://tproger.ru/translations/how-to-read-write-excel-file-java-poi-example/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
32. Create Excel Files in C# [Электронный ресурс]: сайт. URL: https://www.codebyamir.com/blog/create-excel-files-in-c-sharp (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
33. A Python library to read/write Excel 2010 xlsx/xlsm files [Электронный ресурс]: сайт. URL: https://openpyxl.readthedocs.io/en/stable/ (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
34. Opencsv Users Guide [Электронный ресурс]: документация. URL: http://opencsv.sourceforge.net/#writing (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
35. How to connect via HTTPS using Jsoup? [Электронный ресурс]: документация. URL: https://stackoverflow.com/questions/7744075/how-to-connect-via-https-using-jsoup (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.

# Приложение А. Реализация парсера с помощью JSoup

import java.util.regex.Pattern;

import java.util.stream.Collectors;

import java.util.stream.Stream;

@Component

public class LetyShopsParser implements ParserInterface {

private static final Logger log = Logger.getLogger(LetyShopsParser.class);

private final Pattern patternForDiscount;

private final Pattern patternForLabel;

private final ExecutorService pool;

private final CompletionService<List<Shop>> completionService;

@Value("${parsing.site.letyshops}")

private String addressOfSite;

public LetyShopsParser() {

int THREADS = Runtime.getRuntime().availableProcessors();

this.pool = Executors.newFixedThreadPool(THREADS);

completionService = new ExecutorCompletionService<>(this.pool);

patternForLabel = Pattern.compile("[$%€]|руб|(р.)|cent");

patternForDiscount = Pattern.compile("\\d+[.|,]\*\\d\*");

}

@Timer

@Override

public List<Shop> parsing() {

BasicConfigurator.configure();

int maxPage = getMaxPage();

List<Future<List<Shop>>> futures = new LinkedList<>();

List<Shop> result;

for (int i = 1; i <= maxPage; i++) {

final int finalI = i;

futures.add(completionService.submit(() -> parsElements(finalI)));

}

result = futures.stream().flatMap(getFutureStream()).collect(Collectors.toList());

return result;

}

@PreDestroy

private void destroy() {

pool.shutdown();

}

private Function<Future<List<Shop>>, Stream<? extends Shop>> getFutureStream() {

return it -> {

try {

return it.get().stream();

} catch (InterruptedException | ExecutionException e) {

log.error(e);

return Stream.empty();

}

};

}

private List<Shop> parsElements(int i) throws IOException {

Document document = Jsoup.connect("https://letyshops.com/shops?page=" + i).get();

Elements items = document.getElementsByClass("b-teaser\_\_inner");

List<Shop> pageResult = new LinkedList<>();

for (Element item : items) {

String name = getName(item);

String pagesOnTheSite = getPagesOnTheSite(item);

Double discount = getDiscount(item);

String label = getLabel(item);

String image = getImage(item);

if (name != null & image != null & (discount != Double.NaN & discount != 0) & label != null) {

LetyShops letyShops = new LetyShops(name, discount, label, pagesOnTheSite, image);

pageResult.add(letyShops);

}

}

return pageResult;

}

private String getName(Element item) {

return item.getElementsByClass("b-teaser\_\_title").text();

}

private String getImage(Element item) {

return item.getElementsByClass("b-teaser\_\_cover").first().getElementsByTag("img").attr("src");

}

private String getLabel(Element item) {

String label = item.getElementsByClass("b-shop-teaser\_\_label").text();

Matcher matcher = patternForLabel.matcher(label);

if (matcher.find()) {

return label.substring(matcher.start(), matcher.end());

}

return "";

}

private String getPagesOnTheSite(Element item) {

return addressOfSite + item.attr("href");

}

private Double getDiscount(Element item) {

String discount = item.getElementsByClass("b-shop-teaser\_\_cash-value-row").first().text();

String trueDiscount = "";

Matcher matcher = patternForDiscount.matcher(discount);

while (matcher.find()) {

trueDiscount = matcher.group();

}

try {

return Double.parseDouble(trueDiscount.replace(',','.'));

} catch (NumberFormatException e) {

log.error(e);

return Double.NaN;

}

}

private int getMaxPage() {

try {

Elements elements = Jsoup.connect("https://letyshops.com/shops?page=1")

.get()

.getElementsByClass("b-pagination\_\_item");

return Integer.parseInt(elements.get(elements.size() - 2).text());

} catch (IOException e) {

log.error(e);

return 0;

}

}

}

# Приложение Б. Реализация парсера с помощью Selenium Java

package com.turchenkov.parsing.parsingmethods;

import com.turchenkov.parsing.customannotation.Timer;

import com.turchenkov.parsing.domains.shop.MegaBonus;

import com.turchenkov.parsing.domains.shop.Shop;

import org.apache.log4j.BasicConfigurator;

import org.apache.log4j.Logger;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.springframework.stereotype.Component;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutionException;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.Future;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

@Component

public class WebDriverMegaBonusParser implements ParserInterface {

private static final Logger log = Logger.getLogger(WebDriverMegaBonusParser.class);

private final Pattern patternForName = Pattern.compile("Подробнее про кэшбэк в ([\\w\\s\\d\\W]+)");

private final Pattern patternForDiscount = Pattern.compile("\\d+[.|,]\*\\d\*");

private final Pattern patternForLabel = Pattern.compile("[$%€]|руб|(р.)|cent|р|Р");

private final String addressForParsing = "https://megabonus.com/feed";

private final String addressOfSite = "https://megabonus.com";

private final ExecutorService pool;

public WebDriverMegaBonusParser() {

pool = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());

}

@Timer

@Override

public List<Shop> parsing() {

BasicConfigurator.configure();

try {

System.setProperty("webdriver.chrome.driver", "chromedriver\_win32/chromedriver.exe");

} catch (Exception e) {

log.error(e);

}

log.info(WebDriverMegaBonusParser.class.getSimpleName() + " is working");

List<Shop> result = new ArrayList<>();

List<Future<Shop>> futures = new ArrayList<>();

WebDriver driver = new ChromeDriver();

driver.get(addressForParsing);

WebElement button = driver.findElement(By.className("see-more"));

while (button.isDisplayed()) {

try {

button.click();

button = driver.findElement(By.className("see-more"));

} catch (Exception e) {

log.error(e);

}

}

WebElement listOfElements = driver.findElement(By.className("cacheback-block-list"));

List<WebElement> elements = listOfElements.findElements(By.tagName("li"));

driver.close();

for (WebElement element : elements) {

futures.add(pool.submit(() -> parseElements(element)));

}

for (Future<Shop> future : futures) {

try {

Shop shop = future.get();

if (shop != null) {

result.add(shop);

}

} catch (InterruptedException | ExecutionException e) {

log.error(e);

}

}

return result;

}

private Shop parseElements(WebElement element) {

String name = getName(element);

String fullDiscount = getFullDiscount(element);

Double discount = getDiscount(fullDiscount);

String image = getImage(element);

String label = getLabel(fullDiscount);

String url = getUrl(element);

if (name != null & image != null & label != null & discount != Double.NaN & url != null) {

return new MegaBonus(name, discount, label, url, image);

}

return null;

}

private String getUrl(WebElement element) {

String url;

try {

url = element.findElement(By.cssSelector("\"div.holder-img > a\"")).getAttribute("href");

} catch (Exception e) {

log.error(e);

return null;

}

return url;

}

private String getFullDiscount(WebElement element) {

String fullDiscount;

try {

fullDiscount = element.findElement(By.cssSelector("\"div.your-percentage > strong\"")).getText();

} catch (Exception e) {

log.error(e);

return null;

}

return fullDiscount;

}

private String getLabel(String fullDiscount) {

if (fullDiscount == null) {

return null;

}

Matcher matcher = patternForLabel.matcher(fullDiscount);

if (matcher.find()) {

return matcher.group();

}

return null;

}

private Double getDiscount(String fullDiscount) {

if (fullDiscount == null) {

return Double.NaN;

}

String discount = "";

Matcher matcher = patternForDiscount.matcher(fullDiscount);

if (matcher.find()) {

discount = matcher.group();

}

try {

return Double.parseDouble(discount);

} catch (NumberFormatException e) {

log.error(e);

return Double.NaN;

}

}

private String getImage(WebElement element) {

String image;

try {

image = element.findElement(By.tagName("img")).getAttribute("src");

} catch (Exception e) {

log.error(e);

return null;

}

return image;

}

private String getName(WebElement element) {

String name;

try {

name = element.findElement(By.className("holder-more")).findElement(By.tagName("a")).getAttribute("innerHTML");

} catch (Exception e) {

log.error(e);

return null;

}

Matcher matcher = patternForName.matcher(name);

if (matcher.find()) {

return matcher.group(1);

}

return null;

}

}

# Приложение В. Реализация парсера с помощью Unirest для сайта Kopikot.ru

package com.turchenkov.parsing.parsingmethods;

import com.jayway.jsonpath.JsonPath;

import com.mashape.unirest.http.HttpResponse;

import com.mashape.unirest.http.Unirest;

import com.mashape.unirest.http.exceptions.UnirestException;

import com.turchenkov.parsing.customannotation.Timer;

import com.turchenkov.parsing.domains.shop.Kopikot;

import com.turchenkov.parsing.domains.shop.Shop;

import org.apache.log4j.BasicConfigurator;

import org.apache.log4j.Logger;

import org.springframework.stereotype.Component;

import javax.annotation.PreDestroy;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.ExecutionException;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.Future;

public class KopikotParser implements ParserInterface {

private static final Logger log = Logger.getLogger(KopikotParser.class);

private final ExecutorService pool;

public KopikotParser() {

int THREADS = Runtime.getRuntime().availableProcessors();

pool = Executors.newFixedThreadPool(THREADS);

}

@Timer

@Override

public List<Shop> parsing() {

BasicConfigurator.configure();

List<Shop> result = new ArrayList<>();

List<Future<List<Shop>>> futures = new ArrayList<>();

HttpResponse<String> response = null;

for (int i = 0; i <= 2000; i += 100) {

String url = "https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_" + i + "\_order\_popularity.json";

try {

response = Unirest.get(url)

.header("User-Agent", "PostmanRuntime/7.11.0")

.header("Accept", "\*/\*")

.header("Cache-Control", "no-cache")

.header("Postman-Token", "ac6a192d-72f5-4238-9321-55c1308e9846,09ec761f-7c53-4696-bf5a-dc44918e73c5")

.header("Host", "d289b99uqa0t82.cloudfront.net")

.header("accept-encoding", "gzip, deflate")

.header("Connection", "keep-alive")

.header("cache-control", "no-cache")

.asString();

} catch (UnirestException e) {

log.error(e + " : " + i);

break;

}

HttpResponse<String> finalResponse = response;

if (finalResponse != null) {

futures.add(pool.submit(() -> parseElements(finalResponse.getBody())));

}

}

for (Future<List<Shop>> future : futures) {

try {

result.addAll(future.get());

} catch (InterruptedException | ExecutionException e) {

log.error(e);

}

}

try {

Unirest.shutdown();

} catch (IOException e) {

log.error(e);

}

return result;

}

@PreDestroy

private void destroy() {

pool.shutdown();

}

private List<Shop> parseElements(String response) {

List<Shop> elements = new ArrayList<>();

List<Object> items = JsonPath.read(response, "$..items[\*]");

for (Object item : items) {

String name = getName(item);

String shopPage = getShopPage(item);

String image = getImage(item);

Double discount = getDiscount(item);

String label = getLabel(item);

if (name != null & image != null & (discount != Double.NaN & discount != 0) & label != null & shopPage != null) {

elements.add(new Kopikot(name, discount, label, shopPage, image));

}

}

return elements;

}

private String getLabel(Object item) {

String label = JsonPath.read(item, "$.commission.max.unit");

if (label != null) {

return label;

}

return null;

}

private Double getDiscount(Object item) {

Object discount = JsonPath.read(item, "$.commission.max.amount");

if (discount != null) {

try {

return Double.parseDouble(discount.toString());

} catch (NumberFormatException e) {

log.error(e);

return Double.NaN;

}

}

return Double.NaN;

}

private String getImage(Object item) {

String image = JsonPath.read(item, "$.image.url");

if (image != null) {

return image;

}

return null;

}

private String getShopPage(Object item) {

String page = JsonPath.read(item, "$.url");

String id = JsonPath.read(item, "$.id");

if (page != null & id != null) {

return "https://www.kopikot.ru/stores/" + page + "/" + id;

}

return null;

}

private String getName(Object item) {

String name = JsonPath.read(item, "$.title");

if (name != null) {

return name;

}

return null;

}

}

# Приложение Г. Реализация парсера с помощью HtmlAgilityPack

using HtmlAgilityPack;

using NLog;

using Parsing\_on\_.net.Models;

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Web;

namespace Parsing\_on\_.net.BLL.Parsing\_Methods

{

public class HtmlAgilityPackParsing : IParser

{

private static readonly Logger logger = LogManager.GetCurrentClassLogger();

private const int defaultPageCount = 30;

private const string addressOfSiteForMaxPage = "https://letyshops.com/shops?page=1";

private const string addressOfSiteForParsing = "https://letyshops.com/shops?page=";

private const string addressOfSite = "https://letyshops.com";

private HtmlWeb htmlWeb;

public HtmlAgilityPackParsing()

{

htmlWeb = new HtmlWeb

{

OverrideEncoding = Encoding.UTF8

};

}

public List<Shop> Parsing()

{

ConcurrentBag<Shop> shops = new ConcurrentBag<Shop>();

logger.Info("Начался парсинг " + typeof(HtmlAgilityPackParsing).Name);

for (int i = 1; i <= GetMaxPage(); i++)

{

var document = htmlWeb.Load(addressOfSiteForParsing + i);

var nodes = document.DocumentNode.SelectNodes("\*//a[@class='b-teaser\_\_inner']");

Parallel.ForEach(nodes, node =>

{

var shop = ParseElements(node);

if (shop != null)

{

shops.Add(shop);

}

});

}

return shops.ToList();

}

private Shop ParseElements(HtmlNode item)

{

String name = GetName(item);

double discount = GetDiscount(item);

String label = GetLabel(item);

String url = GetURL(item);

String image = GetImage(item);

if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(url)))

{

return new Shop(ConvertString(name), discount, label, image, url);

}

return null;

}

private string GetName(HtmlNode node)

{

var name = node.SelectSingleNode("\*//div[@class='b-teaser\_\_title']").InnerText;

if (!String.IsNullOrWhiteSpace(name))

{

return name.Trim();

}

return null;

}

private double GetDiscount(HtmlNode node)

{

string discount = "";

try

{

discount = node.SelectSingleNode("\*//span[@class='b-shop-teaser\_\_cash']").InnerText;

if (!String.IsNullOrWhiteSpace(discount))

{

return Double.Parse(discount, CultureInfo.InvariantCulture);

}

}

catch (NullReferenceException e)

{

logger.Info("Вторая попытка взять discount " + e);

discount = node.SelectSingleNode("\*//span[@class='b-shop-teaser\_\_new-cash']").InnerText;

if (!String.IsNullOrWhiteSpace(discount))

{

return Double.Parse(discount, CultureInfo.InvariantCulture);

}

}

return Double.NaN;

}

private string GetLabel(HtmlNode node)

{

var label = "";

try

{

label = node.SelectSingleNode("//span[4]").InnerText;

if (!String.IsNullOrWhiteSpace(label))

{

return label;

}

}

catch (NullReferenceException e)

{

logger.Info("Вторая попытка взять label " + e);

label = node.SelectSingleNode("//span[3]").InnerText;

if (!String.IsNullOrWhiteSpace(label))

{

return label;

}

}

return null;

}

private string GetURL(HtmlNode node)

{

var url = node.GetAttributeValue("href", "");

if (!String.IsNullOrWhiteSpace(url))

{

return addressOfSite + url;

}

return null;

}

private string GetImage(HtmlNode node)

{

var image = node.SelectSingleNode("\*//div[@class='b-teaser\_\_cover']/img").GetAttributeValue("src", "");

if (!String.IsNullOrWhiteSpace(image))

{

return image;

}

return null;

}

private int GetMaxPage()

{

var document = htmlWeb.Load(addressOfSiteForMaxPage);

var maxPageString = document.DocumentNode.SelectSingleNode("//ul[@class='b-pagination js-pagination']/li[5]");

if (maxPageString == null)

{

return defaultPageCount;

}

if (int.TryParse(maxPageString.InnerText, out int maxPageInt))

{

return maxPageInt;

}

else

{

return defaultPageCount;

}

}

private string GetHTML(string url)

{

string pageUrl = "";

var request = (HttpWebRequest)WebRequest.Create(url);

using (var response = (HttpWebResponse)request.GetResponse())

{

if (response.StatusCode == HttpStatusCode.OK)

{

var receiveStream = response.GetResponseStream();

if (receiveStream != null)

{

StreamReader streamReader;

if (response.CharacterSet == null)

{

streamReader = new StreamReader(receiveStream);

}

else

{

streamReader = new StreamReader(receiveStream, Encoding.GetEncoding(response.CharacterSet));

}

pageUrl = streamReader.ReadToEnd();

streamReader.Close();

}

}

}

return pageUrl;

}

private string ConvertString(string text)

{

return WebUtility.HtmlDecode(text);

}

}

}

# Приложение Д. Реализация парсера с помощью Fizzler

using HtmlAgilityPack;

using Parsing\_on\_.net.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Fizzler.Systems.HtmlAgilityPack;

using NLog;

using System.Collections.Concurrent;

namespace Parsing\_on\_.net.BLL.Parsing\_Methods

{

public class FizzlerParsing : IParser

{

private static readonly Logger logger = LogManager.GetCurrentClassLogger();

private const int defaultPageCount = 30;

private const string addressOfSiteForMaxPage = "https://letyshops.com/shops?page=1";

private const string addressOfSiteForParsing = "https://letyshops.com/shops?page=";

private const string addressOfSite = "https://letyshops.com";

private ConcurrentBag<Shop> Shops { get; set; }

public FizzlerParsing()

{

Shops = new ConcurrentBag<Shop>();

}

public List<Shop> Parsing()

{

logger.Info("Начался парсинг " + typeof(FizzlerParsing).Name);

var maxPage = GetMaxPage();

Parallel.For(1, maxPage + 1, ParseElements);

return Shops.ToList();

}

private void ParseElements(int i)

{

var htmlWeb = new HtmlWeb

{

OverrideEncoding = Encoding.UTF8

};

var document = htmlWeb.Load(addressOfSiteForParsing + i);

var html = document.DocumentNode;

var listOfShops = html.QuerySelectorAll("div.b-teaser > a.b-teaser\_\_inner");

foreach (var item in listOfShops)

{

String name = GetName(item);

Double discount = GetDiscount(item);

String label = GetLabel(item);

String url = GetUrl(item);

String image = GetImage(item);

if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(url)))

{

Shops.Add(new Shop(name, discount, label, image, url));

}

}

}

private String GetName(HtmlNode html)

{

return html.QuerySelector("div.b-teaser\_\_title").InnerText.Trim();

}

private String GetLabel(HtmlNode html)

{

var labelList = html.QuerySelectorAll("div.b-teaser\_\_caption > div.b-teaser\_\_cashback-rate > div > div > span.b-shop-teaser\_\_label");

return labelList.Last().InnerText.Trim();

}

private String GetUrl(HtmlNode html)

{

return addressOfSite + html.GetAttributeValue("href", "");

}

private String GetImage(HtmlNode html)

{

return html.QuerySelector("div.b-teaser\_\_top > div.b-teaser\_\_cover > img").GetAttributeValue("src", "");

}

private Double GetDiscount(HtmlNode html)

{

string discount = "";

try

{

discount = html.QuerySelector("div.b-teaser\_\_caption > div.b-teaser\_\_cashback-rate > div > div > span.b-shop-teaser\_\_cash").InnerText.Trim();

}

catch (NullReferenceException e)

{

logger.Info("Вторая попытка взять discount " + e);

discount = html.QuerySelector("div.b-teaser\_\_caption > div.b-teaser\_\_cashback-rate > div > div > span.b-shop-teaser\_\_new-cash").InnerText.Trim(); ;

}

if (Double.TryParse(discount.Replace('.', ','), out double result))

{

return result;

}

return Double.NaN;

}

private Int32 GetMaxPage()

{

var htmlWeb = new HtmlWeb

{

OverrideEncoding = Encoding.UTF8

};

var document = htmlWeb.Load("https://letyshops.com/shops?page=1");

var html = document.DocumentNode;

string maxPage = html.QuerySelector("div.b-content.b-content--shops > ul > li:nth-child(5) > a").InnerText.Trim();

if (Int32.TryParse(maxPage, out int result))

{

return result;

}

return 30; } }}

# Приложение Ж. Реализация парсера с помощью AngleSharp

using AngleSharp.Html.Parser;

using NLog;

using Parsing\_on\_.net.Models;

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

namespace Parsing\_on\_.net.BLL.Parsing\_Methods

{

public class AngleSharpParsing : IParser

{

private static readonly Logger logger = LogManager.GetCurrentClassLogger();

private const int \_defaultPage = 30;

private ConcurrentBag<Shop> shops;

private const string address = "https://letyshops.com";

public AngleSharpParsing()

{

this.shops = new ConcurrentBag<Shop>();

}

public List<Shop> Parsing()

{

logger.Info("Начался парсинг " + typeof(AngleSharpParsing).Name);

WebClient webClient = new WebClient

{

Encoding = Encoding.UTF8

};

var html = webClient.DownloadString("https://letyshops.com/shops?page=1");

var maxPage = GetMaxPage(html);

Parallel.For(1, maxPage + 1, ParseElements);

return shops.ToList();

}

private void ParseElements(int i)

{

WebClient webClient = new WebClient

{

Encoding = Encoding.UTF8

};

string page = "https://letyshops.com/shops?page=" + i;

string html = webClient.DownloadString(page);

HtmlParser parser = new HtmlParser();

var result = parser.ParseDocument(html).QuerySelectorAll("div.b-teaser-list > div.b-teaser > a");

foreach (var item in result)

{

var name = GetName(item);

var discount = GetDiscount(item);

string label = GetLabel(item);

string image = GetImage(item);

string url = GetUrl(item);

if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(url)))

{

shops.Add(new Shop(name, discount, label, image, url));

}

}

}

private string GetUrl(AngleSharp.Dom.IElement item)

{

return address + item.GetAttribute("href");

}

private string GetImage(AngleSharp.Dom.IElement item)

{

return item.QuerySelector("div.b-teaser\_\_top > div.b-teaser\_\_cover > img").GetAttribute("src");

}

private string GetLabel(AngleSharp.Dom.IElement item)

{

var label = item.QuerySelectorAll("div.b-teaser\_\_caption > div.b-teaser\_\_cashback-rate > div > div > span.b-shop-teaser\_\_label ").Last().TextContent;

return label;

}

private double GetDiscount(AngleSharp.Dom.IElement item)

{

double discount;

var discounts = item.GetElementsByClassName("b-shop-teaser\_\_cash");

if (discounts.Count() == 0)

{

string temp = item.GetElementsByClassName("b-shop-teaser\_\_new-cash").First().TextContent;

discount = Double.Parse(temp, CultureInfo.InvariantCulture);

}

else

{

string temp = item.GetElementsByClassName("b-shop-teaser\_\_cash").First().TextContent;

discount = Double.Parse(temp, CultureInfo.InvariantCulture);

}

return discount;

}

private string GetName(AngleSharp.Dom.IElement item)

{

var name = item.GetElementsByClassName("b-teaser\_\_title").First().InnerHtml.ToString().Substring(17);

return name.Substring(0, name.Length - 13);

}

Int32 GetMaxPage(string html)

{

HtmlParser parser = new HtmlParser();

var result = parser.ParseDocument(html).GetElementsByClassName("b-pagination\_\_link");

try

{

if (Int32.TryParse(result.Last().TextContent, out int page))

{

return page;

}

}

catch (InvalidOperationException e)

{

logger.Error("Элемент не найден " + e);

}

return \_defaultPage;

}

}

}

# Приложение И. Реализация парсера с помощью CsQuery

using CsQuery;

using NLog;

using Parsing\_on\_.net.Models;

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Threading.Tasks;

namespace Parsing\_on\_.net.BLL.Parsing\_Methods

{

public class CsQueryParsing : IParser

{

private static readonly Logger logger = LogManager.GetCurrentClassLogger();

private const string address = "https://letyshops.com/shops?page=";

private readonly int defaultMaxpage = 20;

private ConcurrentBag<Shop> Shops { get; set; }

public CsQueryParsing()

{

Shops = new ConcurrentBag<Shop>();

}

public List<Shop> Parsing()

{

logger.Info("Начался парсинг " + typeof(CsQueryParsing).Name);

var maxPage = GetMaxPage();

for (int i = 1; i <= GetMaxPage(); i++)

{

var document = CQ.CreateFromUrl(address + i);

var domObjects = document["a.b-teaser\_\_inner"];

Parallel.ForEach(domObjects, domObject =>

{

Shop shop;

if ((shop = ParseElements(domObject)) != null)

{

Shops.Add(shop);

}

});

}

return Shops.ToList();

}

private Shop ParseElements(IDomObject domObject)

{

var document = CQ.CreateDocument(domObject.InnerHTML);

string name = GetName(document);

double discount = GetDiscount(document);

string label = GetLabel(document);

string image = GetImage(document);

string url = GetUrl(domObject);

if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(url)))

{

return new Shop(name, discount, label, image, url);

}

else

{

return null;

}

}

private string GetUrl(IDomObject domObject)

{

return "https://letyshops.com" + domObject.Attributes.GetAttribute("href");

}

private string GetImage(CQ document)

{

return document["img"].First().Attr<string>("src");

}

private string GetLabel(CQ document)

{

string label = document["span.b-shop-teaser\_\_label "].Last().Text().Trim();

return label;

}

private double GetDiscount(CQ document)

{

string discount = document["span.b-shop-teaser\_\_cash"].First().Text().Trim();

if (String.IsNullOrWhiteSpace(discount))

{

discount = document["span.b-shop-teaser\_\_new-cash"].First().Text().Trim();

}

if (double.TryParse(discount, NumberStyles.Number, CultureInfo.InvariantCulture, out double trueDiscount))

{

return trueDiscount;

}

return double.NaN;

}

private string GetName(CQ document)

{

return document["div.b-teaser\_\_title"].First().Text().Trim();

}

private Int32 GetMaxPage()

{

var document = CQ.CreateFromUrl("https://letyshops.com/shops?page=1");

var numbers = document["a.b-pagination\_\_link"];

if (Int32.TryParse(numbers[numbers.Length - 2].InnerText.Trim(), out Int32 maxPage))

{

return maxPage;

}

return defaultMaxpage;

}

}

}

# Приложение К. Реализация парсера с помощью RestSharp для сайта Kopikot.ru

using Newtonsoft.Json.Linq;

using NLog;

using Parsing\_on\_.net.Models;

using RestSharp;

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

namespace Parsing\_on\_.net.BLL.Parsing\_Methods

{

public class RestSharpParsing : IParser

{

private static readonly Logger logger = LogManager.GetCurrentClassLogger();

public ConcurrentBag<Shop> Shops { get; set; }

public RestSharpParsing()

{

Shops = new ConcurrentBag<Shop>();

}

public List<Shop> Parsing()

{

logger.Info("Начался парсинг " + typeof(RestSharpParsing).Name);

int[] pages = new int[]

{

0,100,200,300,400,500,600,700,800,900,1000,1100,1200,1300

};

List<JToken> listOfItems = new List<JToken>();

Parallel.ForEach(pages, page =>

{

try

{

listOfItems.Add(GetJSon(page));

}

catch (Exception e)

{

logger.Error(e);

}

});

Parallel.ForEach(listOfItems, item =>

{

try

{

Shops.Add(ParseElements(item));

}

catch (Exception e)

{

logger.Error(e);

}

});

return Shops.ToList();

}

private JToken GetJSon(int i)

{

string url = $"https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_{i}\_order\_popularity.json";

var client = new RestClient(url);

var request = new RestRequest(Method.GET);

request.AddHeader("cache-control", "no-cache");

request.AddHeader("Connection", "keep-alive");

request.AddHeader("accept-encoding", "gzip, deflate");

request.AddHeader("Host", "d289b99uqa0t82.cloudfront.net");

request.AddHeader("Postman-Token", "048aef15-143b-4f61-8c44-60467f64a33d,e85413f5-28a6-4878-b792-942c640071cc");

request.AddHeader("Cache-Control", "no-cache");

request.AddHeader("Accept", "\*/\*");

request.AddHeader("User-Agent", "PostmanRuntime/7.11.0");

IRestResponse response = client.Execute(request);

JObject jsonParse = JObject.Parse(response.Content);

var listOfItems = jsonParse["items"];

return listOfItems;

}

private Shop ParseElements(JToken jToken)

{

var name = GetName(jToken);

var image = GetImage(jToken);

var shopUrl = GetUrl(jToken);

var discount = GetDiscount(jToken);

var label = GetLabel(jToken);

if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(shopUrl)))

{

return new Shop(name, discount, label, image, shopUrl);

}

else

{

return null;

}

}

private String GetName(JToken token)

{

return token["title"].ToString();

}

private String GetImage(JToken token)

{

return token["image"]["url"].ToString();

}

private String GetUrl(JToken token)

{

return "https://www.kopikot.ru/stores/" + token["url"].ToString() + "/" + token["id"].ToString();

}

private Double GetDiscount(JToken token)

{

string discount = token["commission"]["max"]["original\_amount"].ToString();

if (double.TryParse(discount.Replace('.', ','), out double result))

{

return result;

}

return Double.NaN;

}

private String GetLabel(JToken token)

{

return token["commission"]["max"]["unit"].ToString();

}

}

}

# Приложение Л. Реализация парсера с помощью Selenium C#

using NLog;

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using Parsing\_on\_.net.Models;

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Linq;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Web;

namespace Parsing\_on\_.net.BLL.Parsing\_Methods

{

public class WebDriverParsing : IParser

{

private static readonly Logger logger = LogManager.GetCurrentClassLogger();

public List<Shop> Parsing()

{

logger.Info("Начался парсинг " + typeof(WebDriverParsing).Name);

ConcurrentBag<Shop> shops = new ConcurrentBag<Shop>();

IWebDriver driver = new ChromeDriver();

driver.Navigate().GoToUrl("https://megabonus.com/feed");

var button = driver.FindElement(By.ClassName("see-more"));

while (button.Displayed)

{

try

{

button.Click();

button = driver.FindElement(By.ClassName("see-more"));

}

catch (Exception e)

{

logger.Info("Кнопка 'Показать ещё' не была найдена. Поиск продолжится " + e);

}

}

var ul = driver.FindElement(By.ClassName("cacheback-block-list"));

var webElements = ul.FindElements(By.TagName("li"));

Parallel.ForEach(webElements, WebElement =>

{

var shop = ParseElements(WebElement);

if (shop != null)

{

shops.Add(shop);

}

});

driver.Close();

return shops.ToList();

}

private Shop ParseElements(IWebElement item)

{

var name = GetName(item);

var fullDiscount = GetFullDiscount(item);

var discount = GetDiscount(fullDiscount);

var label = GetLabel(fullDiscount);

var image = GetImage(item);

var url = GetPage(item);

if (!(String.IsNullOrEmpty(name) || Double.IsNaN(discount) || String.IsNullOrEmpty(label) || String.IsNullOrEmpty(image) || String.IsNullOrEmpty(url)))

{

return new Shop(name, discount, label, image, url);

}

return null;

}

private String GetPage(IWebElement element)

{

String page = "";

try

{

page = element.FindElement(By.CssSelector("div.holder-img > a")).GetAttribute("href");

}

catch (Exception e)

{

if (e is NullReferenceException || e is NoSuchElementException)

{

logger.Error("Произошла ошибка: " + e);

return null;

}

}

return page;

}

private String GetLabel(String fullDiscount)

{

if (fullDiscount == null)

{

logger.Error("Пустой fullDiscount");

return null;

}

Regex regex = new Regex("[$%€]|руб|(р.)|cent|р|Р|RUB|USD|EUR|SEK|UAH|INR|BRL|GBP|CHF|PLN");

Match matcher = regex.Match(fullDiscount);

if (matcher.Success)

{

return matcher.Value;

}

return null;

}

private Double GetDiscount(String fullDiscount)

{

if (fullDiscount == null)

{

logger.Error("Пустой fullDiscount");

return Double.NaN;

}

Regex regex = new Regex("\\d+[.|,]\*\\d\*");

String discount = "";

Match matcher = regex.Match(fullDiscount);

if (matcher.Success)

{

discount = matcher.Value;

}

if (Double.TryParse(discount.Replace('.', ','), out double result))

{

return result;

}

return Double.NaN;

}

private String GetFullDiscount(IWebElement element)

{

String fullDiscount = "";

try

{

fullDiscount = element.FindElement(By.CssSelector("div.your-percentage > strong")).Text;

}

catch (Exception e)

{

if (e is NullReferenceException || e is NoSuchElementException)

{

logger.Error("Произошла ошибка: " + e);

return null;

}

}

return fullDiscount;

}

private String GetImage(IWebElement element)

{

String image = "";

try

{

image = element.FindElement(By.TagName("img")).GetAttribute("src");

}

catch (Exception e)

{

if (e is NullReferenceException || e is NoSuchElementException)

{

logger.Error("Произошла ошибка: " + e);

return null;

}

}

return image;

}

private String GetName(IWebElement element)

{

Regex regex = new Regex("Подробнее про кэшбэк в ([\\w\\s\\d\\W]+)");

String name = "";

try

{

name = element.FindElement(By.ClassName("holder-more")).FindElement(By.TagName("a")).GetAttribute("innerHTML");

}

catch (Exception e)

{

if (e is NullReferenceException || e is NoSuchElementException)

{

logger.Error("Произошла ошибка: " + e);

return null;

}

}

Match matcher = regex.Match(name);

if (matcher.Success)

{

return matcher.Groups[1].Value;

}

return null;

}

}

}

# Приложение М. Реализация парсера с помощью BeautifulSoup

**from** multiprocessing **import** Pool  
  
**import** requests  
**from** bs4 **import** BeautifulSoup  
  
**import** parsing\_methods.valueForParsing **as** v  
**from** parsing\_methods.parsingAbstractClass **import** Parsing  
**from** parsing\_methods.shop **import** Shop  
  
  
**class** BS4Parsing(Parsing):  
  
 **def** get\_name\_class(self):  
 **return** type(self).\_\_name\_\_  
  
 **def** \_\_init\_\_(self):  
 **pass  
  
 def** parsing(self):  
 *"""Возвращает список элементов"""* urls = []  
 max\_page = self.\_\_get\_max\_page()  
 **for** i **in** range(1, max\_page + 1):  
 urls.append(v.url\_for\_parsing\_letyShops + str(i))  
 pool = Pool(processes=4)  
 result = pool.map(self.parse\_elements, urls)  
 shops = []  
 **for** items **in** result:  
 **for** item **in** items:  
 shops.append(item)  
 **return** shops  
  
 **def** parse\_elements(self, url):  
 result = []  
 soup = BeautifulSoup(self.\_\_get\_Html(url), **'lxml'**)  
 shops = soup.find\_all(**'div'**, class\_=**'b-teaser'**)  
 **for** shop **in** shops:  
 name = self.\_\_get\_name(shop)  
 discount = self.\_\_get\_discount(shop)  
 label = self.\_\_get\_label(shop)  
 url = self.\_\_get\_url(shop)  
 image = self.\_\_get\_image(shop)  
 **if** name **is not None and** discount **is not None and** label **is not None and** image **is not None and** url **is not None**:  
 result.append(Shop(name=name, discount=discount, label=label, image=image, url=url))  
 **return** result  
  
 **def** \_\_get\_image(self, shop):  
 image = shop.find(**'div'**, class\_=**'b-teaser\_\_cover'**).find(**'img'**).get(**'src'**)  
 **return** image  
  
 **def** \_\_get\_url(self, shop):  
 url = shop.find(**'a'**, class\_=**'b-teaser\_\_inner'**).get(**'href'**)  
 **return** v.clear\_url\_letyShops + url  
  
 **def** \_\_get\_label(self, shop):  
 label = shop.find\_all(**'span'**, class\_=**'b-shop-teaser\_\_label'**)[-1]  
 **if** label **is None**:  
 label = shop.find(**'span'**, class\_=**'b-shop-teaser\_\_label--red'**)  
 **else**:  
 label = label  
 **return** label.text.strip()  
  
 **def** \_\_get\_discount(self, shop):  
 discount = shop.find(**'span'**, class\_=**'b-shop-teaser\_\_cash'**)  
 **if** discount **is None**:  
 discount = shop.find(**'span'**, class\_=**'b-shop-teaser\_\_new-cash'**).text.strip()  
 **else**:  
 discount = discount.text.strip()  
 **return** discount  
  
 **def** \_\_get\_name(self, shop):  
 name = shop.find(**'div'**, class\_=**'b-teaser\_\_title'**).text.strip()  
 **return** name  
  
 **def** \_\_get\_Html(self, url):  
 **try**:  
 r = requests.get(url)  
 **return** r.text  
 **except** ConnectionError **as** e:  
 print(**"Error"**)  
  
 **def** \_\_get\_max\_page(self):  
 soup = BeautifulSoup(self.\_\_get\_Html(v.letyShops), **'lxml'**)  
 new\_pages = []  
 pages = soup.find\_all(**'a'**, class\_=**'b-pagination\_\_link'**)  
 **for** page **in** pages:  
 new\_page = int(page.get(**'data-page'**))  
 new\_pages.append(new\_page)  
 **return** max(new\_pages)  
  
 **def** print\_array(self, array: []):  
 **if** len(array) > 0:  
 **for** item **in** array:  
 print(item.\_\_str\_\_())  
 **else**:  
 print(**"Пустой список"**)

# Приложение Н. Реализация парсера с помощью Scrapy

**import** csv  
**import** io  
**import** json  
**import** os  
**import** time  
**from** datetime **import** datetime  
  
**import** requests  
**import** scrapy  
**from** bs4 **import** BeautifulSoup  
**from** scrapy.crawler **import** CrawlerProcess  
**from** scrapy.linkextractors **import** LinkExtractor  
**from** scrapy.spiders **import** Rule  
  
**import** parsing\_methods.valueForParsing **as** v  
  
items = []  
  
  
**def** get\_max\_page():  
 soup = BeautifulSoup(get\_Html(v.letyShops), **'lxml'**)  
 new\_pages = []  
 pages = soup.find\_all(**'a'**, class\_=**'b-pagination\_\_link'**)  
 **for** page **in** pages:  
 new\_page = int(page.get(**'data-page'**))  
 new\_pages.append(new\_page)  
 **return** max(new\_pages)  
  
  
**def** get\_Html(url):  
 **try**:  
 r = requests.get(url)  
 **return** r.text  
 **except** ConnectionError **as** e:  
 print(**"Error"**)  
  
  
**class** MainClassForScrapy():  
 *"""Класс для запуска паука и записи данных в json"""* file\_name = **'E:/Документы/PyCharmProject/Django/files/shops.json'** items = []  
  
 **def** \_\_init\_\_(self):  
 self.delete\_file(self.file\_name)  
 start\_time = time.time()  
 process = CrawlerProcess()  
 process.crawl(ArcadySpider)  
 process.start()  
 end\_time = time.time()  
 self.create\_file\_with\_time(end\_time - start\_time)  
 self.create\_json\_file(items)  
  
 **def** get\_data\_from\_json(self):  
 **with** io.open(self.file\_name, **'r'**, encoding=**'utf8'**) **as** file:  
 data = json.load(file)  
 file.close()  
 **return** data  
  
 **def** create\_json\_file(self, shops):  
 **with** io.open(self.file\_name, **'a+'**, encoding=**'utf8'**) **as** file:  
 json.dump(shops, file, indent=2, ensure\_ascii=**False**)  
 file.close()  
  
 **def** delete\_file(self, file):  
 **if** os.path.exists(file):  
 os.remove(file)  
  
 **def** create\_file\_with\_time(self, timer):  
 now\_date = datetime.today()  
 file\_name\_for\_time = **'E:/Документы/PyCharmProject/Django/files/times\_of\_scrapy\_for\_'** \  
 + str(now\_date).replace(**':'**, **'.'**) + **'.csv'  
 with** open(file\_name\_for\_time, **'w'**) **as** file:  
 columns = [**'name'**, **'time'**, **'date'**]  
 writer = csv.DictWriter(file, fieldnames=columns)  
 new\_time = {**'name'**: **'ArcadySpider'**,  
 **'time'**: timer,  
 **'date'**: datetime.today(),  
 }  
 writer.writerow(new\_time)  
  
  
**class** ArcadySpider(scrapy.Spider):  
 name = **"arcady"** address = **"https://letyshops.com/shops?page="** clear\_address = **'https://letyshops.com'** allowed\_domains = [**'https://letyshops.com'**]  
 start\_urls = []  
 max\_page = get\_max\_page()  
 **for** i **in** range(1, max\_page + 1):  
 start\_urls.append(address + i.\_\_str\_\_())  
  
 rules = (  
 Rule(LinkExtractor(allow=(**''**)), callback=**"parse"**, follow=**False**)  
 )  
  
 **def** parse(self, response):  
 shops = response.xpath(**'//div[@class="b-teaser"]'**)  
 **for** i, shop **in** enumerate(shops):  
 item = {  
 **'name'**: self.get\_name(shop, i),  
 **'discount'**: self.get\_discount(shop, i),  
 **'label'**: self.get\_label(shop, i),  
 **'image'**: self.get\_image(shop, i),  
 **'url'**: self.get\_url(shop, i)  
 }  
 items.append(item)  
 **return** items  
  
 **def** get\_name(self, shop, i):  
 index = i + 1  
 name = shop.xpath(  
 **'//div[@class="b-teaser"]['** + index.\_\_str\_\_() + **']//div[@class="b-teaser\_\_title"]//text()'**).get().strip()  
 **return** name  
  
 **def** get\_url(self, shop, i):  
 index = i + 1  
 url = shop.xpath(  
 **'//div[@class="b-teaser"]['** + index.\_\_str\_\_() + **']//a[@class="b-teaser\_\_inner"]/@href'**).extract()  
 **return** self.clear\_address.\_\_str\_\_() + url[0]  
  
 **def** get\_discount(self, shop, i):  
 index = i + 1  
 discount = shop.xpath(  
 **'//div[@class="b-teaser"]['** + index.\_\_str\_\_() + **']/a//span[@class="b-shop-teaser\_\_cash"]/text()'**).get()  
 **if** discount **is None**:  
 discount = shop.xpath(  
 **'//div[@class="b-teaser"]['** + index.\_\_str\_\_() + **']/a//span[@class="b-shop-teaser\_\_new-cash"]/text()'**).get()  
 print(discount)  
 **return** discount  
 **return** discount  
  
 **def** get\_label(self, shop, i):  
 index = i + 1  
 label = shop.xpath(  
 **'//div[@class="b-teaser"]['** + index.\_\_str\_\_() + **']/a//span[@class="b-shop-teaser\_\_label "]/text()'**).get()  
 **if** label **is None**:  
  
 label = shop.xpath(  
 **'//div[@class="b-teaser"]['** + index.\_\_str\_\_() + **']/a//span[@class="b-shop-teaser\_\_label b-shop-teaser\_\_label--red"]/text()'**)  
 **return** label  
 **return** label  
  
 **def** get\_image(self, shop, i):  
 index = i + 1  
 image = shop.xpath(  
 **'//div[@class="b-teaser"]['** + index.\_\_str\_\_() + **']//div[@class="b-teaser\_\_cover"]/img/@src'**).extract()  
 **return** image  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main = MainClassForScrapy()

# Приложение П. Реализация парсера с помощью Requests для сайта Kopikot.ru

**import** json  
**import** time  
**from** multiprocessing.pool **import** Pool  
  
**import** requests  
  
**from** parsing\_methods.parsingAbstractClass **import** Parsing  
**from** parsing\_methods.shop **import** Shop  
  
  
**class** RequestsParsing(Parsing):  
 **def** get\_name\_class(self):  
 **return** type(self).\_\_name\_\_  
  
 \_\_address = **"https://www.kopikot.ru/"  
  
 def** \_\_init\_\_(self):  
 **pass  
  
 def** parsing(self):  
 *"""Возвращает список элементов"""* i = [0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300]  
 pool = Pool(processes=4)  
 all\_items = (pool.map(self.get\_json, i))  
 result = pool.map(self.parse\_elements, all\_items)  
 shops = []  
 **for** items **in** result:  
 **for** item **in** items:  
 shops.append(item)  
 **return** shops  
  
 **def** get\_json(self, i):  
 url = **"https://d289b99uqa0t82.cloudfront.net/sites/5/campaigns\_limit\_100\_offset\_"** + str(  
 i) + **"\_order\_popularity.json"** payload = **""** headers = {  
 **'User-Agent'**: **"PostmanRuntime/7.11.0"**,  
 **'Accept'**: **"\*/\*"**,  
 **'Cache-Control'**: **"no-cache"**,  
 **'Postman-Token'**: **"b6eeb7b4-63dd-454e-b213-6d2d62b74946,1e851911-db3b-4406-88de-5ffc9ecbaa5d"**,  
 **'Host'**: **"d289b99uqa0t82.cloudfront.net"**,  
 **'accept-encoding'**: **"gzip, deflate"**,  
 **'Connection'**: **"keep-alive"**,  
 **'cache-control'**: **"no-cache"** }  
 response = requests.request(**"GET"**, url, data=payload, headers=headers)  
 data = json.loads(response.text)  
 items = data[**"items"**]  
 **return** items  
  
 **def** parse\_elements(self, items):  
 result = []  
 **for** item **in** items:  
 name = self.\_\_get\_name(item)  
 discount = self.\_\_get\_discount(item)  
 label = self.\_\_get\_label(item)  
 image = self.\_\_get\_image(item)  
 page = self.\_\_get\_url(item)  
 result.append(Shop(name=name, discount=discount, label=label, image=image, url=page))  
 **return** result  
  
 **def** \_\_get\_name(self, item):  
 **return** item[**"title"**]  
  
 **def** \_\_get\_discount(self, item):  
 discount = item[**"commission"**][**"max"**][**"original\_amount"**]  
 **try**:  
 **return** float(discount)  
 **except** ValueError:  
 **return None  
  
 def** \_\_get\_label(self, item):  
 **return** item[**"commission"**][**"max"**][**"unit"**]  
  
 **def** \_\_get\_image(self, item):  
 **return** item[**"image"**][**"url"**]  
  
 **def** \_\_get\_url(self, item):  
 id = item[**"id"**]  
 url = item[**"url"**]  
 **return** self.\_\_address + url + **"/"** + id  
  
 **def** print\_array(self, array: []):  
 **if** len(array) > 0:  
 **for** item **in** array:  
 print(item.\_\_str\_\_())  
 **else**:  
 print(**"Пустой список"**)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 parser = RequestsParsing()  
 result = parser.parsing()  
 **for** item **in** result:  
 print(item)

# Приложение Р. Реализация парсера с помощью Selenium Python и BeautifulSoup

**import** logging  
**import** re  
**import** time  
**from** multiprocessing.pool **import** Pool  
  
**from** bs4 **import** BeautifulSoup  
**from** selenium **import** webdriver  
  
**from** parsing\_methods.parsingAbstractClass **import** Parsing  
**from** parsing\_methods.shop **import** Shop  
  
  
**class** WebDriverParsing(Parsing):  
 **def** get\_name\_class(self):  
 **return** type(self).\_\_name\_\_  
  
 logger = logging.getLogger(**"WebDriverParsing"**)  
 logger.setLevel(logging.INFO)  
  
 *# create the logging file handler* fh = logging.FileHandler(**"WebDriverParsing.log"**)  
  
 formatter = logging.Formatter(**'[%(asctime)s] p%(process)s {%(lineno)d} %(levelname)s - %(message)s'**,  
 **'%m-%d %H:%M:%S'**)  
 fh.setFormatter(formatter)  
  
 *# add handler to logger object* logger.addHandler(fh)  
  
 \_\_file\_path\_to\_Chrome = **"E:\Документы\PyCharmProject\Parsing-on-python\chromedriver\_win32\chromedriver.exe"** address = **"https://megabonus.com/feed"  
  
 def** \_\_init\_\_(self):  
 **pass  
  
 def** parsing(self):  
 *"""Возвращает список элементов"""* **global** driver  
 **try**:  
 driver = webdriver.Chrome(  
 self.\_\_file\_path\_to\_Chrome)  
 driver.get(self.address)  
 button = driver.find\_element\_by\_class\_name(**"see-more"**)  
 button.click()  
 **while** button.is\_displayed():  
 **try**:  
 button.click()  
 button = driver.find\_element\_by\_class\_name(**"see-more"**)  
 **except** Exception **as** e:  
 self.logger.error(e)  
 ul = driver.find\_element\_by\_class\_name(**"cacheback-block-list"**).get\_attribute(**"outerHTML"**)  
 soup = BeautifulSoup(ul, **'lxml'**)  
 shops\_str = []  
 shops = soup.find\_all(**"li"**)  
 **for** shop **in** shops:  
 shops\_str.append(shop.\_\_str\_\_())  
 pool = Pool(processes=4)  
 result = pool.map(func=self.parse\_elements, iterable=shops\_str)  
 **return** result  
 **finally**:  
 driver.close()  
  
 **def** parse\_elements(self, element):  
 soup = BeautifulSoup(element, **'lxml'**)  
 name = self.get\_name(soup)  
 full\_discount = self.get\_full\_discount(soup)  
 discount = self.get\_discount(full\_discount)  
 label = self.get\_label(full\_discount)  
 image = self.get\_image(soup)  
 url = self.get\_url(soup)  
 **if** name **is not None and** discount **is not None and** label **is not None and** image **is not None and** url **is not None**:  
 **return** Shop(name=name, discount=discount, label=label, image=image, url=url)  
  
 **def** get\_name(self, element):  
 pattern\_for\_name = **"Подробнее про кэшбэк в ([\\w\\s\\d\\W]+)"  
 try**:  
 name = element.find(**'div'**, class\_=**'holder-more'**).find(**'a'**).text.strip()  
 name\_search = re.search(pattern\_for\_name, name)  
 **return** name\_search.group(1)  
 **except** Exception **as** e:  
 self.logger.error(e)  
  
 **def** get\_full\_discount(self, element):  
 **try**:  
 **return** element.find(**'div'**, class\_=**'percent\_cashback'**).text.strip()  
 **except** Exception **as** e:  
 self.logger.error(e)  
  
 **def** get\_discount(self, full\_discount):  
 pattern\_for\_discount = **"\\d+[.|,]\*\\d\*"  
 if** full\_discount **is not None**:  
 **try**:  
 discount = re.search(pattern\_for\_discount, full\_discount)  
 **return** float(discount.group(0))  
 **except** Exception **as** e:  
 self.logger.error(e)  
  
 **def** get\_label(self, full\_discount):  
 pattern\_for\_label = **"[$%€]|руб|(р.)|cent|р|Р|RUB|USD|EUR|SEK|UAH|INR|BRL|GBP|CHF|PLN"  
 if** full\_discount **is not None**:  
 **try**:  
 label\_search = re.search(pattern\_for\_label, full\_discount)  
 **return** label\_search.group(0)  
 **except** Exception **as** e:  
 self.logger.error(e)  
  
 **def** get\_image(self, element):  
 **try**:  
 **return** element.find(**'img'**).get(**'src'**)  
 **except** Exception **as** e:  
 self.logger.error(e)  
  
 **def** get\_url(self, element):  
 **try**:  
 **return** self.address + element.find(**'div'**, class\_=**'holder-img'**).find(**'a'**).get(**'href'**)  
 **except** Exception **as** e:  
 self.logger.error(e)  
  
 **def** \_\_print\_array(self, array: []):  
 **for** item **in** array:  
 print(item.\_\_str\_\_())  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 webDriverParsing = WebDriverParsing()  
 webDriverParsing.parsing()