

МИНОБРНАУКИ РОССИИ
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
1 Отбор библиотек для парсинга	5
2 Реализация на языке Java.....	7
2.1 Описание библиотеки Jsoup	7
2.2 Описание инструмента Selenium	9
2.3 Описание библиотеки Unirest.....	11
3 Реализация на языке C#	19
3.1 Описание библиотеки HtmlAgilityPack.....	19
3.2 Описание библиотеки Fizzler	20
3.3 Описание библиотеки AngleSharp	21
3.4 Описание библиотеки CsQuery	23
3.5 Описание библиотеки RestSharp.....	24
3.6 Описание инструмента Selenium	27
4 Реализация на языке Python	29
4.1 Описание библиотеки BeautifulSoup.....	29
4.2 Описание фреймворка Scrapy	31
4.3 Описание библиотеки Request	34
4.4 Описание инструмента Selenium	36
5 Сравнение скорости реализаций	39
5.1 Сравнение реализаций парсера для сайта MegaBonus.....	39
5.2 Сравнение реализаций парсера для сайта LetyShops.....	41
6 Алгоритм парсинга сайта MegaBonus.....	43
7 Алгоритм парсинга сайта LetyShops.....	50
8 Описание визуализации результатов	57
8.1 Использование MVC	57
8.2 Вывод в Excel файл.....	59
8.3 Вывод в CSV файл.....	60
9 Решение проблем при парсинге сайтов.....	62

10	Заключение.....	64
11	Список использованной литературы.....	65
12	Приложение А. Реализация парсера с помощью JSoup.....	69
13	Приложение Б. Реализация парсера с помощью Selenium Java.....	70
14	Приложение В. Реализация парсера с помощью Unirest	71
15	Приложение Г. Реализация парсера с помощью Unirest для сайта Kopikot.ru 72	
16	Приложение Д. Реализация парсера с помощью Unirest для сайта Cash4Brands.ru	73
17	Приложение Е. Реализация парсера с помощью HtmlAgilityPack.....	74
18	Приложение Ж. Реализация парсера с помощью Fizzler	75
19	Приложение З. Реализация парсера с помощью AngleSharp	76
20	Приложение И. Реализация парсера с помощью CsQuery	77
21	Приложение К. Реализация парсера с помощью RestSharp	78
22	Приложение Л. Реализация парсера с помощью Selenium C#.....	79
23	Приложение М. Реализация парсера с помощью BeutifulSoup	80
24	Приложение Н. Реализация парсера с помощью Scrapy.....	81
25	Приложение О. Реализация парсера с помощью Selenium Python.....	82

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 начинает переходить по всем внутренним ссылкам, тщательно собирая 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 и дать оценку скорости.

1 Отбор библиотек для парсинга

Для парсинга сайтов были отобраны специальные библиотеки для парсинга для трёх языков программирования: Java, C# и Python. Приведённые ниже библиотеки взяты из-за их удобства 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 библиотеки:

1. BeautifulSoup;
2. Request;

Также для Python был использован фреймворк Scrapy.

Помимо библиотек был использован Selenium – это набор различных программных бесплатных 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. Эти операции очень гибкие, 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.

2 Реализация на языке Java

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

2.1 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;
- getElementByClass(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 Рисунок 1.

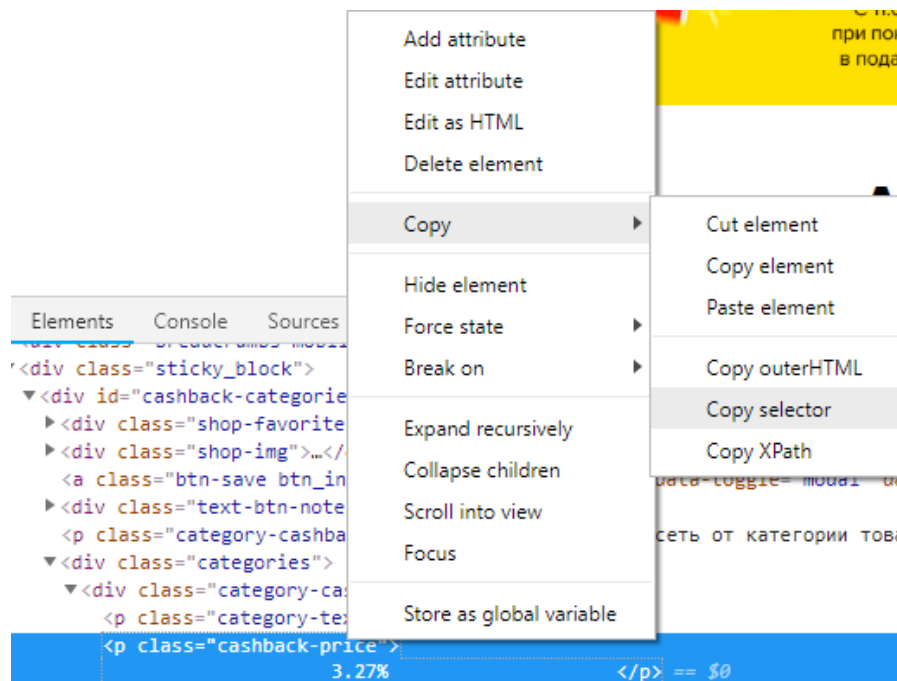


Рисунок 1 – использование возможностей панели разработчика в Google Chrome

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().

2.2 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].

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 Рисунок 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.

2.3 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 [Korikot.ru](http://korikot.ru), который в последствии обрабатывается.

Пример с использованием `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. }

```

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 Korikot.ru, запрос будет отправлен при нажатии кнопки «Больше». После этого в панели разработчика в разделе Network будет нужный XHR. Результат изображён на Рисунок 2 – XHR для 100 магазинов с сайта Korikot.ru.

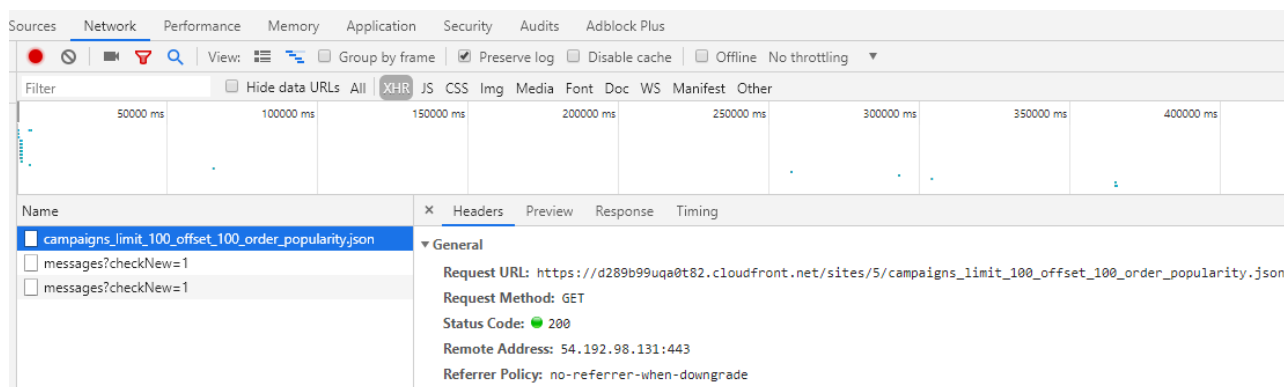


Рисунок 2 – XHR для 100 магазинов с сайта Корикот.ru

Как видно на Рисунок 2 – XHR для 100 магазинов с сайта Корикот.ru, 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 Рисунок 3 – XHR для 200 магазинов с сайта Корикот.ru.



Рисунок 3 – XHR для 200 магазинов с сайта Корикот.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 Рисунок 4 – результат запроса к URL через Postman.

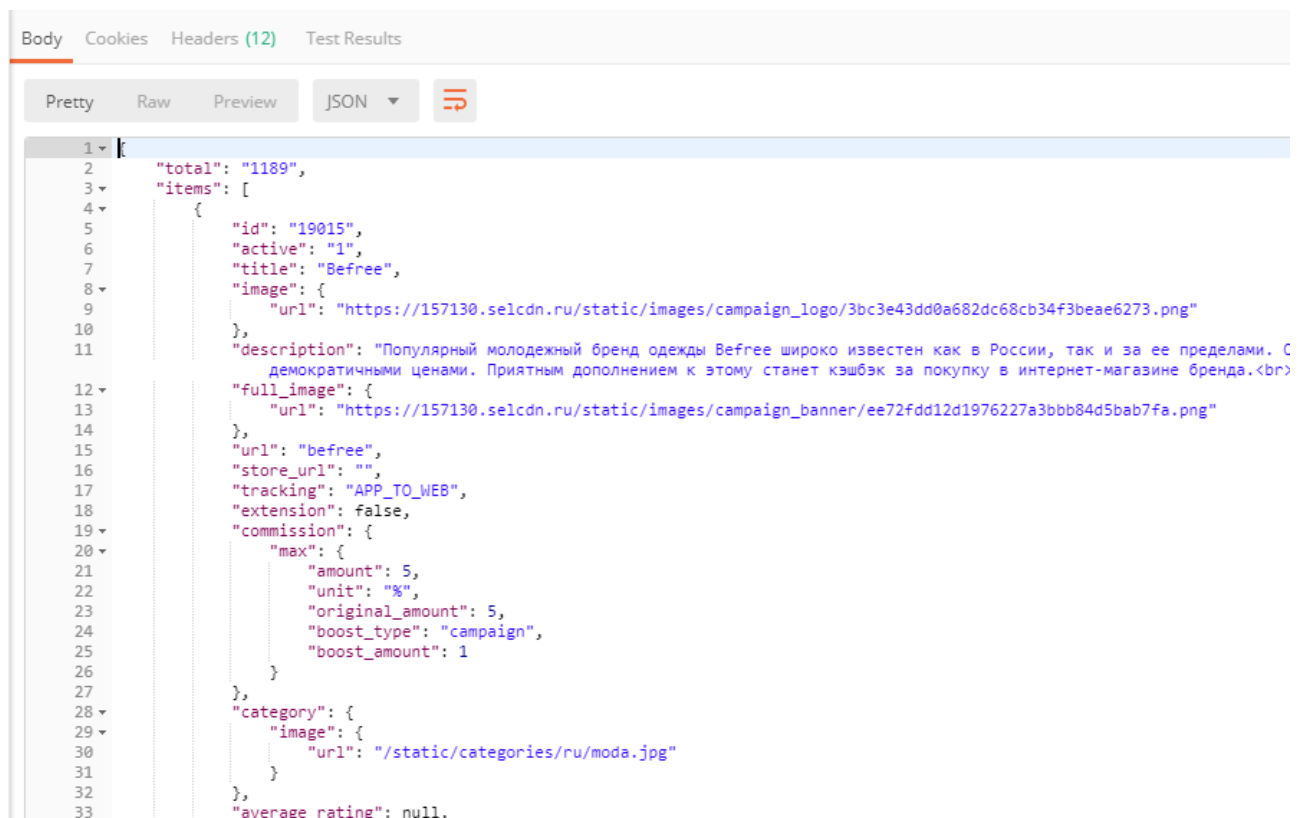


Рисунок 4 – результат запроса к URL через Postman

При просмотре какие запросы происходят на сервере важно обращать внимание на ту информацию, которая показана в Headers. Пример изображён на Рисунок 5 – Headers сайта Kopikot.ru.

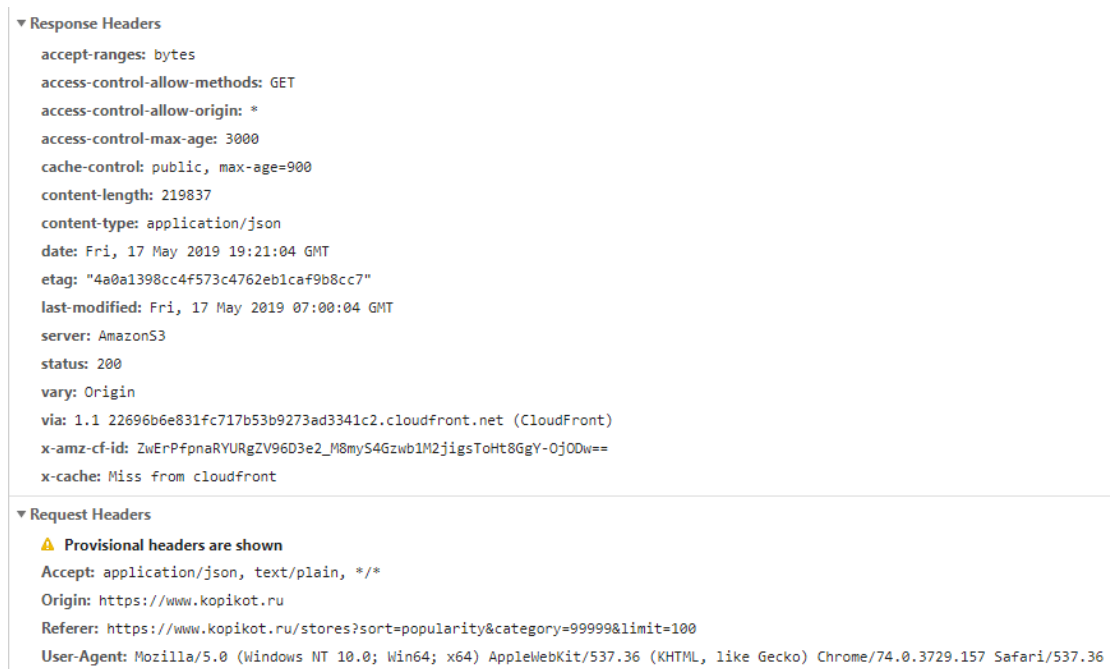


Рисунок 5 – Headers сайта Kopikot.ru

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 на Рисунок 6 – Headers сайта Cash4Brands.ru часть 1 и Рисунок 7 – Headers сайта Cash4Brands.ru часть 2.

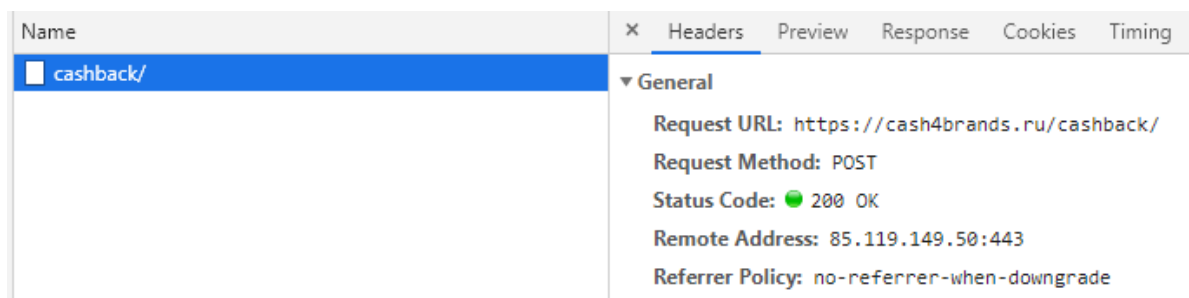


Рисунок 6 – Headers сайта Cash4Brands.ru часть 1

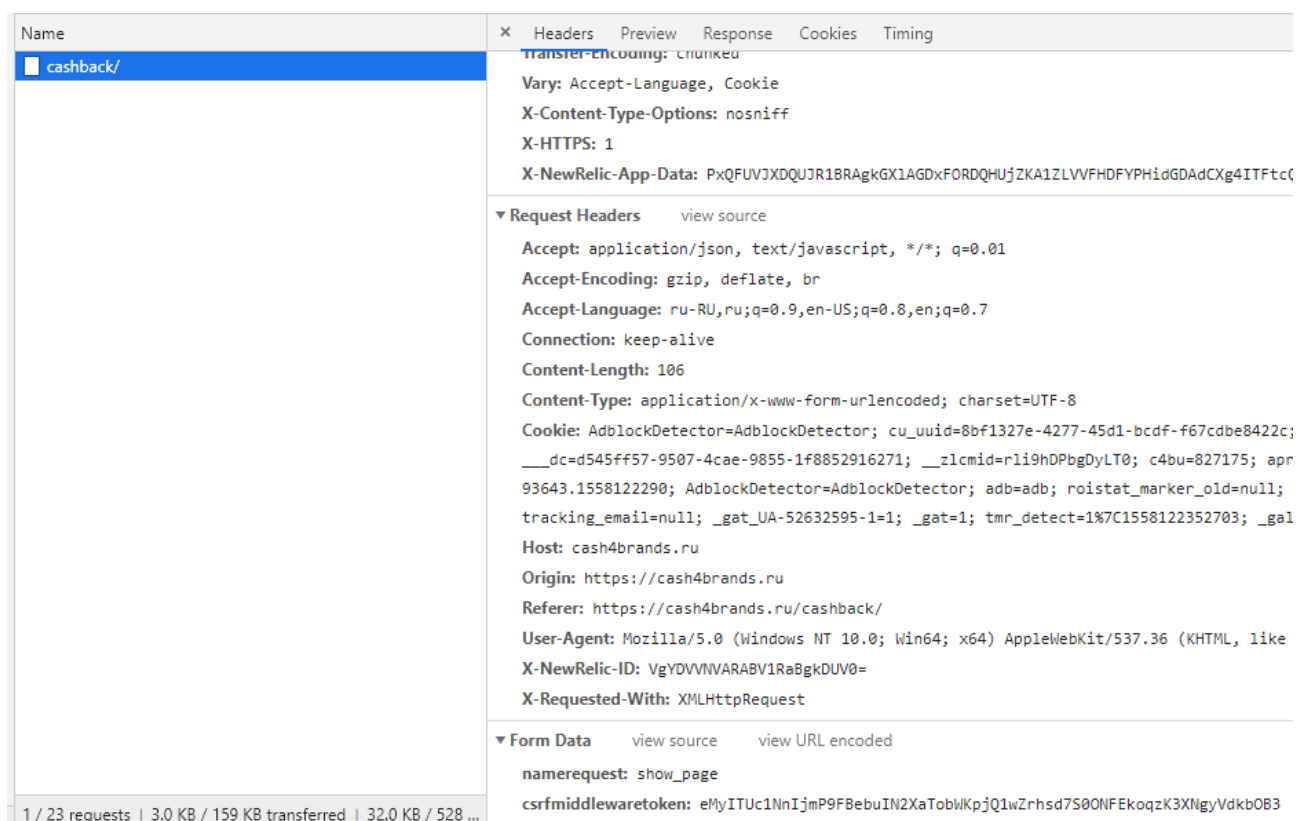


Рисунок 7 – Headers сайта Cash4Brands.ru часть 2

На Рисунок 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 Рисунок 8 – параметры для запроса на Cash4Brands.ru.

KEY	VALUE
<input checked="" type="checkbox"/> namerequest	show_page
Key	Value

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 Рисунок 9 – генерация Unirest кода в Postman.

```
1  HttpResponse<String> response = Unirest.post("https://cash4brands.ru/cashback/")
2  .header("Content-Type", "application/x-www-form-urlencoded")
3  .header("User-Agent", "PostmanRuntime/7.11.0")
4  .header("Accept", "*//*")
5  .header("Cache-Control", "no-cache")
6  .header("Postman-Token", "8775e882-4a32-44f8-a404-407c5968e100,a8e597b4-7677-45b8-b079-7b36a9248920")
7  .header("Host", "cash4brands.ru")
8  .header("cookie", "Cookie_1=value; cu_uuid=3e592ff0-4d01-4f19-9d26-7c3fd24963e4; sessionId=9wq8qalx2xjkyte29xiohch4edgl0a7u")
9  .header("accept-encoding", "gzip, deflate")
10 .header("content-length", "21")
11 .header("Connection", "keep-alive")
12 .header("cache-control", "no-cache")
13 .body("namerequest=show_page")
14 .asString();
```

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 данном случае это процент. Через точку происходит обращение к последующим элементам для извлечения информации.

Плюсы Unirest:

- быстрая скорость выполнения.

Минусы Unirest:

- поиск нужных запросов.

3 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.

3.1 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.

Плюсы `HtmlAgilityPack`:

- средняя скорость выполнения.

Минусы:

- работает только с XPath выражениями.

3.2 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.

Плюсы Fizzler:

- быстрая скорость выполнения;
- имеет все те же методы, что и `HtmlAgilityPack`;
- работает с CSS селекторами.

Минусы Fizzler:

- работает только с XPath выражениями и CSS селекторами.

3.3 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.
- `QuerySelectorAll(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 `InnerText` для извлечения текста между начальным и конечным тэгами 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");`

Плюсы `AngleSharp`:

- быстрая скорость выполнения;
- при отсутствии элемента возвращает `null`, а не выбрасывает ошибку.
- работает с css селекторами

Минусы `AngleSharp`:

- выбор по локаторам возвращает только коллекцию

3.4 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.

Плюсы CsQuery:

- хорошая скорость выполнения;
- 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.

3.5 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 Cash4Brands.ru, который в последствии обрабатывается.

Пример с использованием Dictionary:

1. var client = new RestClient("https://cash4brands.ru/cashback/");
2. var request = new RestRequest(Method.POST);
3. Dictionary<String, String> headers = new Dictionary<string, string>()
4. {
5. { "cache-control", "no-cache" },
6. { "Connection", "keep-alive" },
7. { "content-length", "21" },
8. { "accept-encoding", "gzip, deflate" },
9. { "cookie", "Cookie_1=value; cu_uuid=3e592ff0-4d01-4f19-9d26-7c3fd24963e4; sessionid=9wq8qalx2xjkyte29xiohch4edgl0a7u" },
10. { "Host", "cash4brands.ru" },
11. { "Postman-Token", "91751f4f-c0e5-48c2-9137-cbcbc5e11a0f,1e684343-5161-4efc-8482-495ef99d971b" },
12. { "Cache-Control", "no-cache" },
13. { "Accept", "*/*" },


```

14. { "User-Agent", "PostmanRuntime/7.13.0"},
15. { "Content-Type", "application/x-www-form-urlencoded"}
16. };
17. foreach (var item in headers)
18. {
19. request.AddHeader(item.Key, item.Value);
20. }
21. request.AddParameter("undefined", "namerequest=show_page",
    ParameterType.RequestBody);
22. IRestResponse response = client.Execute(request);

```

Пример без использования Map:

```

1. var client = new RestClient("https://cash4brands.ru/cashback/");
2. var request = new RestRequest(Method.POST);
3. request.AddHeader("cache-control", "no-cache");
4. request.AddHeader("Connection", "keep-alive");
5. request.AddHeader("content-length", "21");
6. request.AddHeader("accept-encoding", "gzip, deflate");
7. request.AddHeader("cookie", "Cookie_1=value; cu_uuid=3e592ff0-4d01-4f19-9d26-
    7c3fd24963e4; sessionid=9wq8qalx2xjkyte29xiohch4edgl0a7u");
8. request.AddHeader("Host", "cash4brands.ru");
9. request.AddHeader("Postman-Token", "91751f4f-c0e5-48c2-9137-
    cbc5e11a0f,1e684343-5161-4efc-8482-495ef99d971b");
10. request.AddHeader("Cache-Control", "no-cache");
11. request.AddHeader("Accept", "*/*");
12. request.AddHeader("User-Agent", "PostmanRuntime/7.13.0");
13. request.AddHeader("Content-Type", "application/x-www-form-urlencoded");
14. request.AddParameter("undefined", "namerequest=show_page",
    ParameterType.RequestBody);
15. IRestResponse response = client.Execute(request);

```

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



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 типа 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:

- быстрая скорость выполнения.

Минусы RestSharp:

- поиск нужных запросов.

3.6 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:

9. `By.Id` – в качестве локатора используется атрибут `id` элемента страницы;

10. `By.Name` – в качестве локатора используется атрибут `name` элемента страницы;

11. `By.XPath` – используется для поиска элемента по XPath выражению;

12.By.TagName – поиск по имени HTML тега;

13.By.ClassName – поиск по классу элемента;

14.By.CssSelector – используется для поиска элемента по CSS селектору;

15.By.LinkText – поиск ссылки с указанным текстом. Текст ссылки должен быть точным совпадением указанного текста;

16.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].

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 Рисунок 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.

4 Реализация на языке Python

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

4.1 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 BeautifulSoup 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».

- 2) 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, который 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().

- 3) image = shop.find('div', class_='b-teaser__cover').find('img').get('src')

В результате в переменной image будет ссылка на картинку магазина.

Плюсы BeautifulSoup:

- может использовать другие парсеры;
- быстрая скорость выполнения.

Минусы BeautifulSoup:

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

4.2 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 spiders 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
```

Полный код приведён в Приложении Л.

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].

Плюсы Scrapy:

- быстрая скорость выполнения;

- 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.

4.3 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.

(<http://docs.python-requests.org/en/master/user/quickstart/>
<http://lecturesnet.readthedocs.io/net/requests/python.html#python>)

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(
        i) + "_order_popularity.json"
3. payload = ""
4. headers = {
5. 'User-Agent': "PostmanRuntime/7.11.0",
6. 'Accept': "*/*",
7. 'Cache-Control': "no-cache",
```

- i) '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 Рисунок 11 – генерация Request кода в Postman.

GENERATE CODE SNIPPETS

Python Requests ▾

Copy to Clipboard

```

1 import requests
2
3 url = "https://cash4brands.ru/cashback/"
4
5 payload = "namerequest=show_page&test=tess"
6 headers = {
7     'Content-Type': "application/x-www-form-urlencoded",
8     'User-Agent': "PostmanRuntime/7.13.0",
9     'Accept': "*/*",
10    'Cache-Control': "no-cache",
11    'Postman-Token': "91751f4f-c0e5-48c2-9137-cbcb5e11a0f,6220b853-d11f-4d3b-b9f5-12f7f6af8ee8",
12    'Host': "cash4brands.ru",
13    'cookie': "Cookie_1=value; cu_uid=3e592ff0-4d01-4f19-9d26-7c3fd24963e4; sessionId=9wq8qalx2xjkyte29xiohch4edgl0a7u",
14    'accept-encoding': "gzip, deflate",
15    'content-length': "21",
16    'Connection': "keep-alive",
17    'cache-control': "no-cache"
18 }
19
20 response = requests.request("POST", url, data=payload, headers=headers)
21
22 print(response.text)
```

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 [29].

Плюсы Request:

- быстрая скорость выполнения;
- удобное добавление заголовков и параметров.

Минус Request:

- поиск нужных запросов.

4.4 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.

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 Рисунок 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 несериализуемыми 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 обнаружили некоторые ссылки, которые использовала библиотека multiprocessing. Из-за этой ошибки дальнейшее распараллеливание было не возможным. Есть 2 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;
- не нужно скачивать дополнительные файлы.

Минусы Selenium:

- использует реальный браузер для работы, из-за чего скорость падает;
- типы объектов являются несериализуемыми.

5 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.

5.1 Сравнение реализаций парсера для сайта MegaBonus

На Рисунок 12 – время выполнения Selenium в секунда изображены замеры времени выполнения работы парсеров на Java и C#, используя только Selenium.

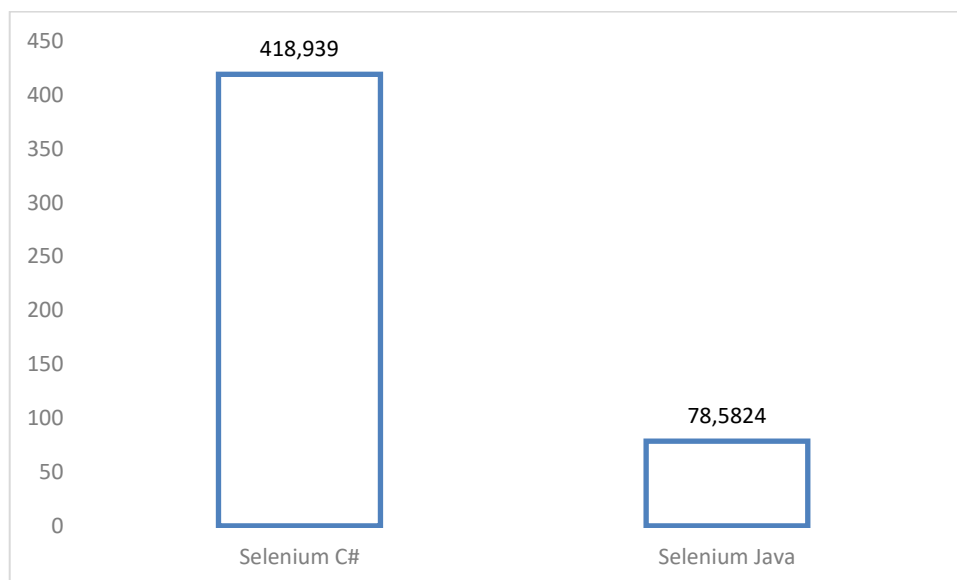


Рисунок 12 – время выполнения Selenium в секундах

На Рисунок 13 – время выполнения Selenium и другой библиотеки в секундах 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 Рисунок 14 – время выполнения парсеров на C# в секундах.

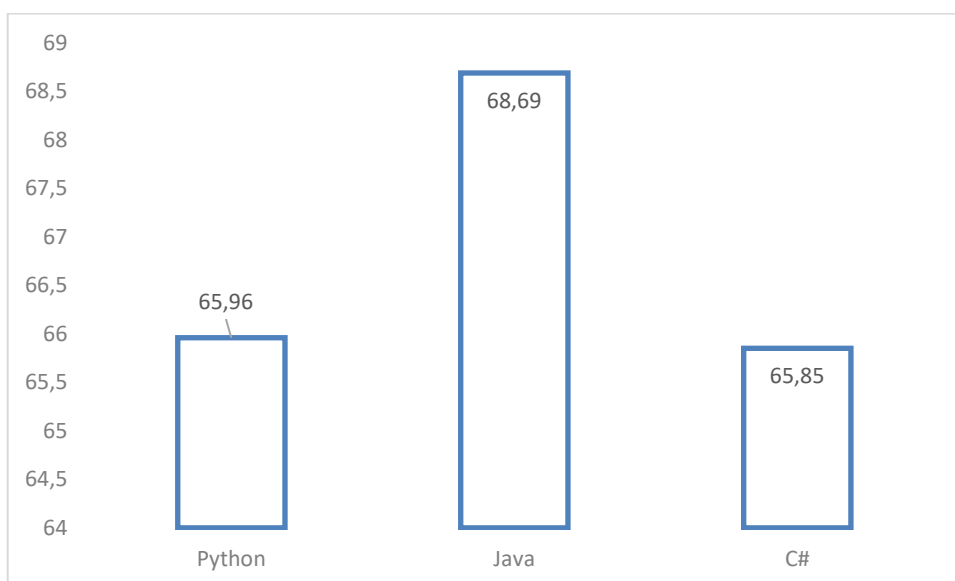


Рисунок 13 – время выполнения Selenium и другой библиотеки в секундах

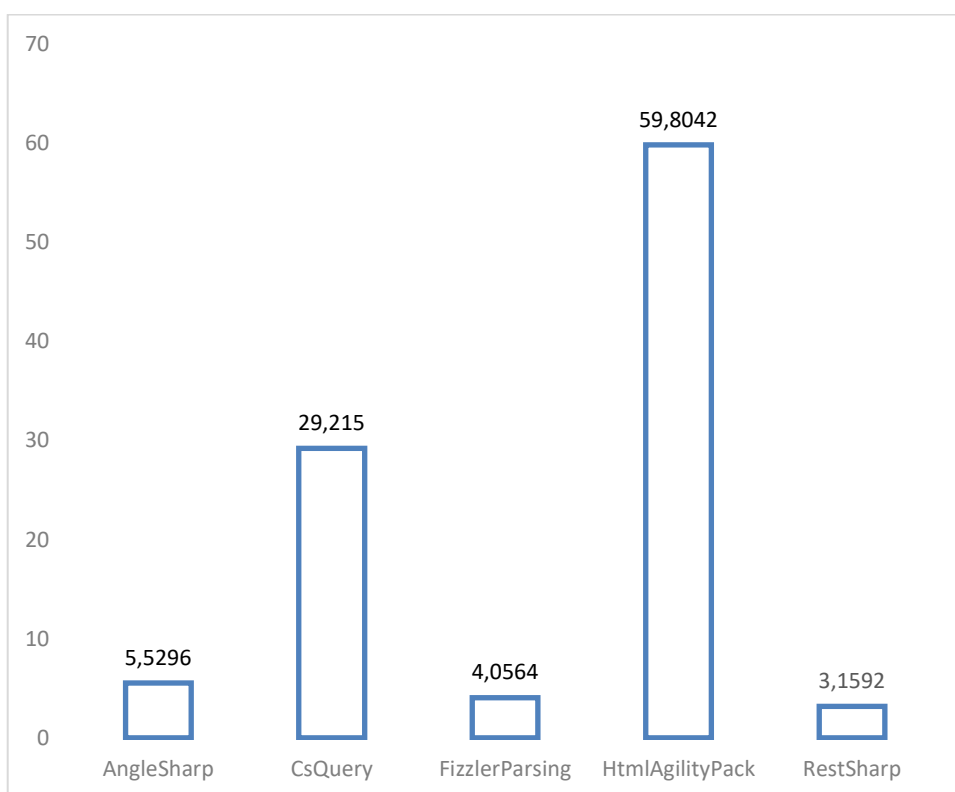


Рисунок 14 – время выполнения парсеров на C# в секундах

5.2 Сравнение реализаций парсера для сайта LetyShops

На Рисунок 15 – время выполнения парсеров на Python в секундах изображены замеры времени выполнения работы парсеров на Python.

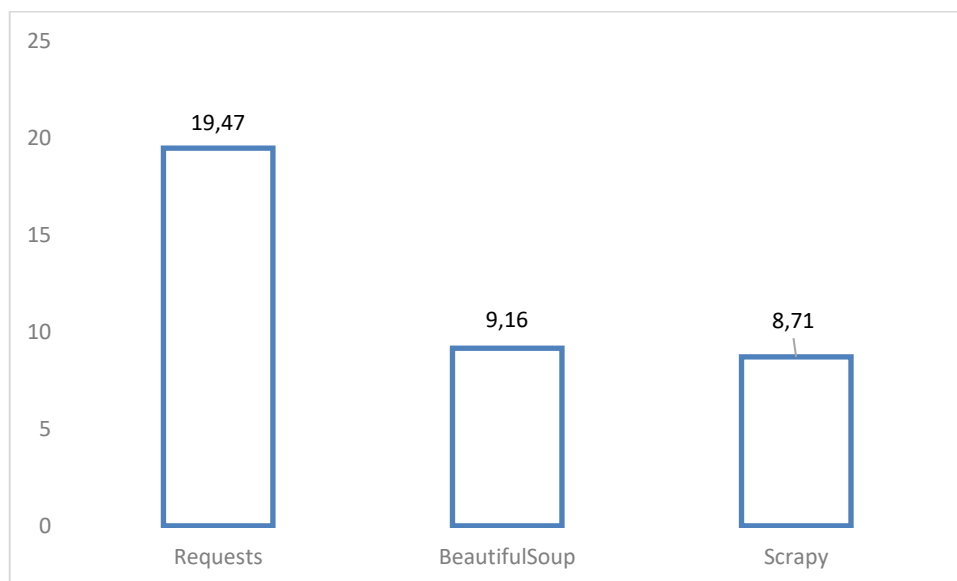


Рисунок 15 – время выполнения парсеров на Python в секундах

На Рисунок 16 – время выполнения парсеров на Java в секундах изображены замеры времени выполнения работы парсеров на Java.

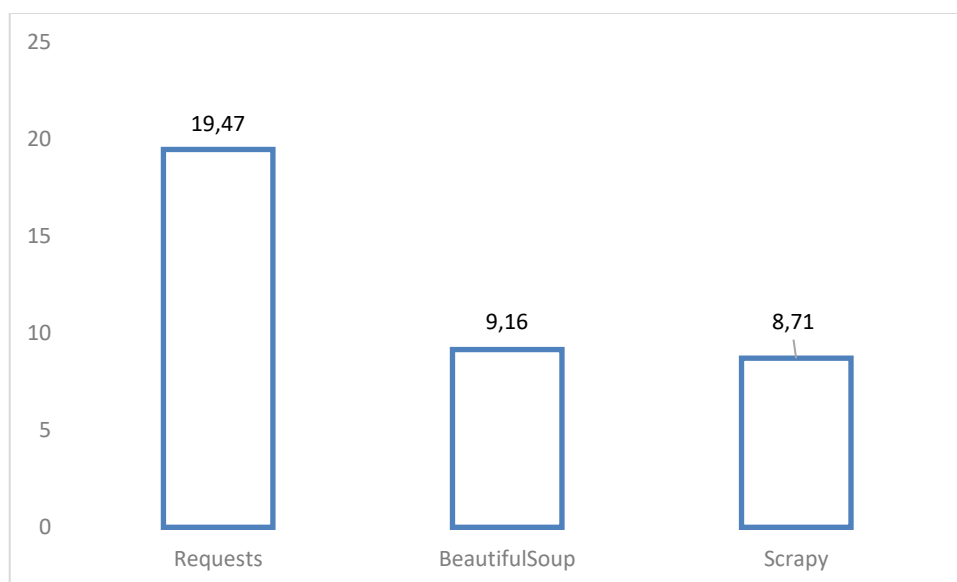


Рисунок 16 – время выполнения парсеров на Java в секундах

На Рисунок 14 – время выполнения парсеров на C# в секундах изображены замеры времени выполнения работы парсеров на C#.

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 однозначно лучше использовать версию на Java.

Рассматривая скорость выполнения других парсеров, можно сделать вывод о библиотеках для парсинга. Для C# быстрее всего работает Fizzler. Для двух других языков программирования не имеет смысл 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.

6 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 Рисунок 17 – страница с магазинами сайта MegaBonus. При просмотре данной страницы следует обратить внимание на саму структуру сайта, то есть на расположение магазинов, а также на те элементы, которые позволяют получить следующие магазины.

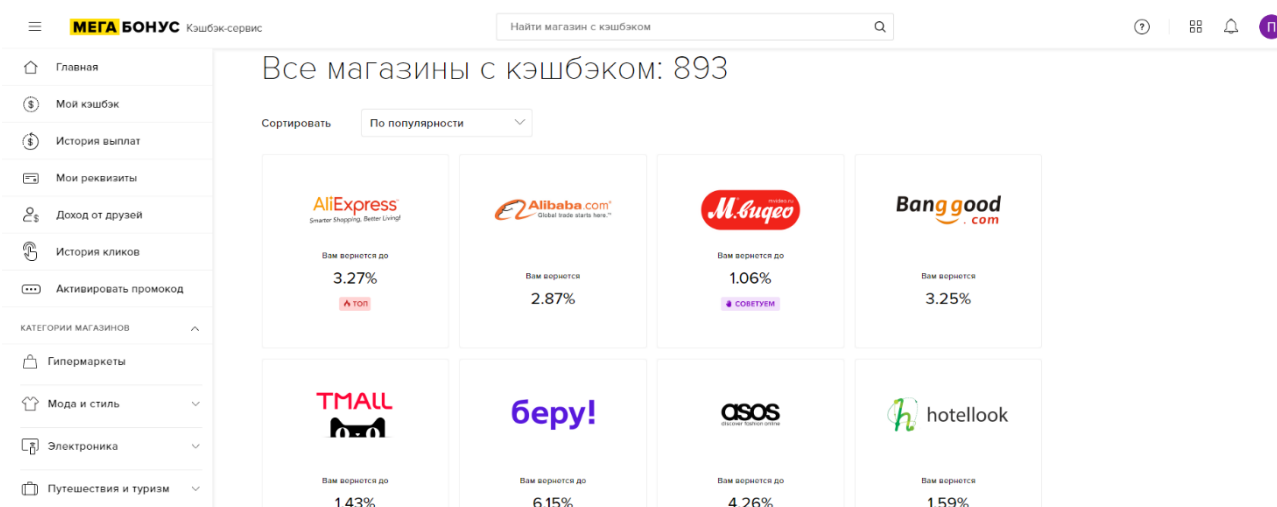


Рисунок 17 – страница с магазинами сайта MegaBonus

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. Элемент, с помощью которого добавляются магазины изображен на Рисунок 18.

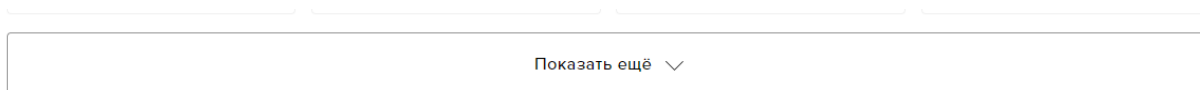


Рисунок 18 - элемент страницы для вывода магазинов

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 Рисунок 19 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"));`

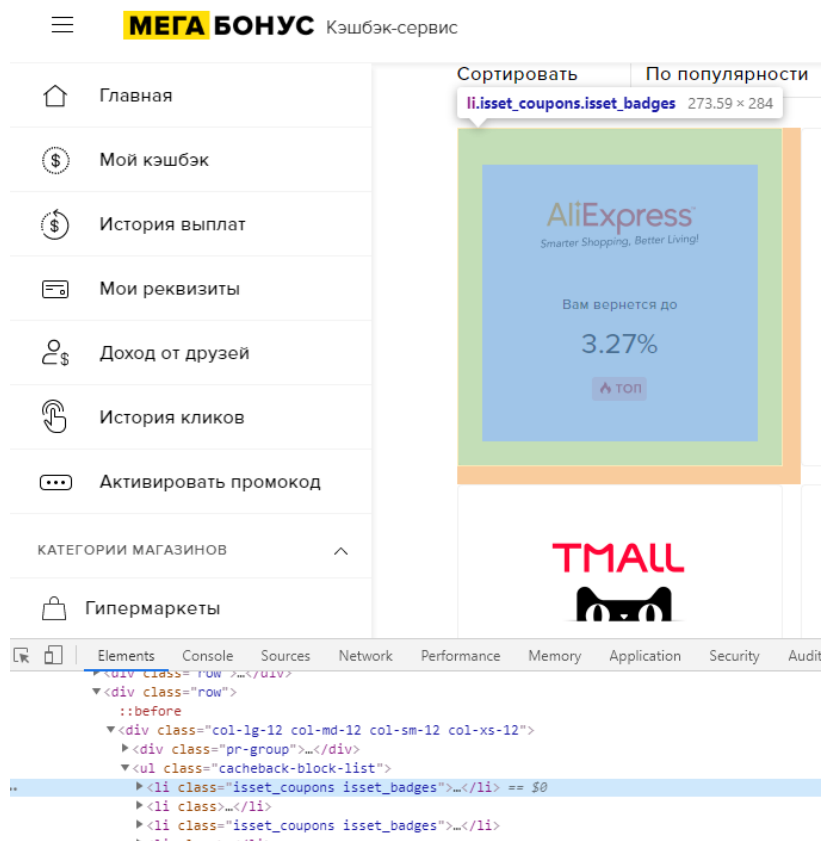


Рисунок 19 - выбор элемента с магазином на MegaBonus

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|(p.)|cent|p|P|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.

7 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 Рисунк 20. При просмотре данной страницы следует обратить внимание на саму структуру сайта, то есть на расположение магазинов, а также на те элементы, которые позволяют получить следующие магазины.

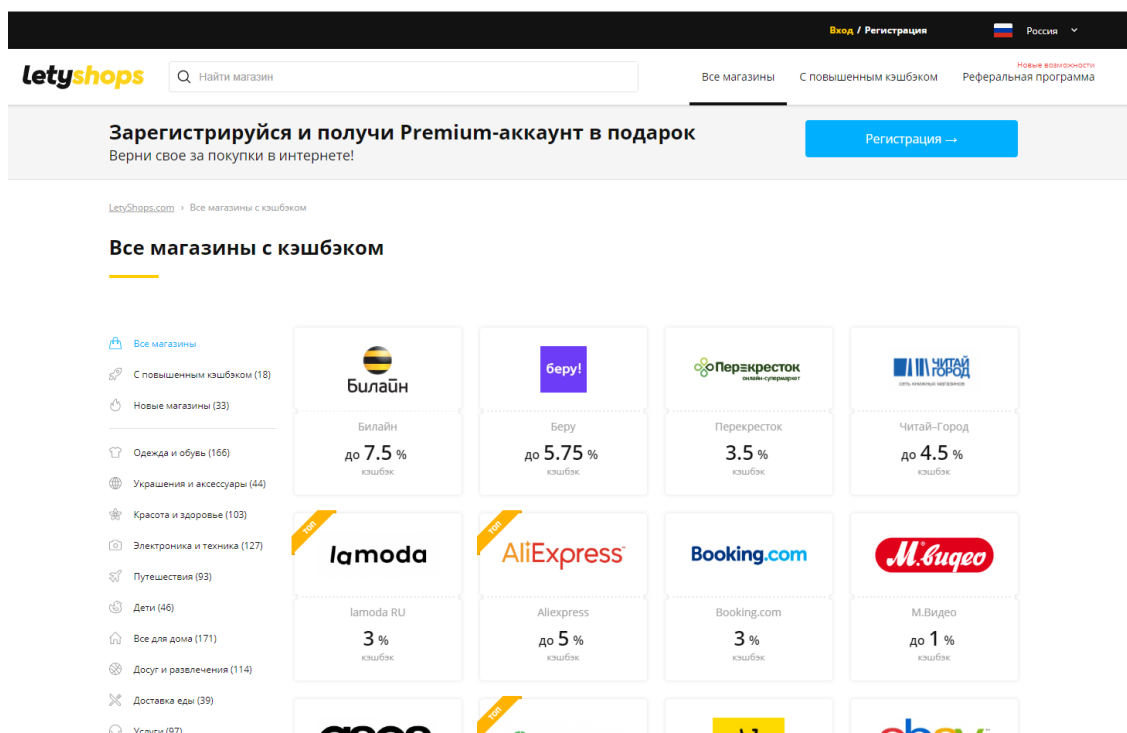


Рисунок 20 – страница с магазинами сайта LetyShops

После всех магазинов расположен элемент, который позволяет переходить по страницам магазина. Элемент изображен на Рисунок 21.

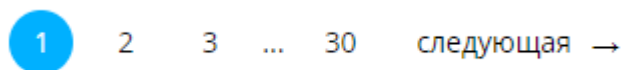


Рисунок 21 – элемент страницы для перехода по страницам

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 Рисунок 22 **Ошибка! Источник ссылки не найден.** изображен выбор конкретного магазина.

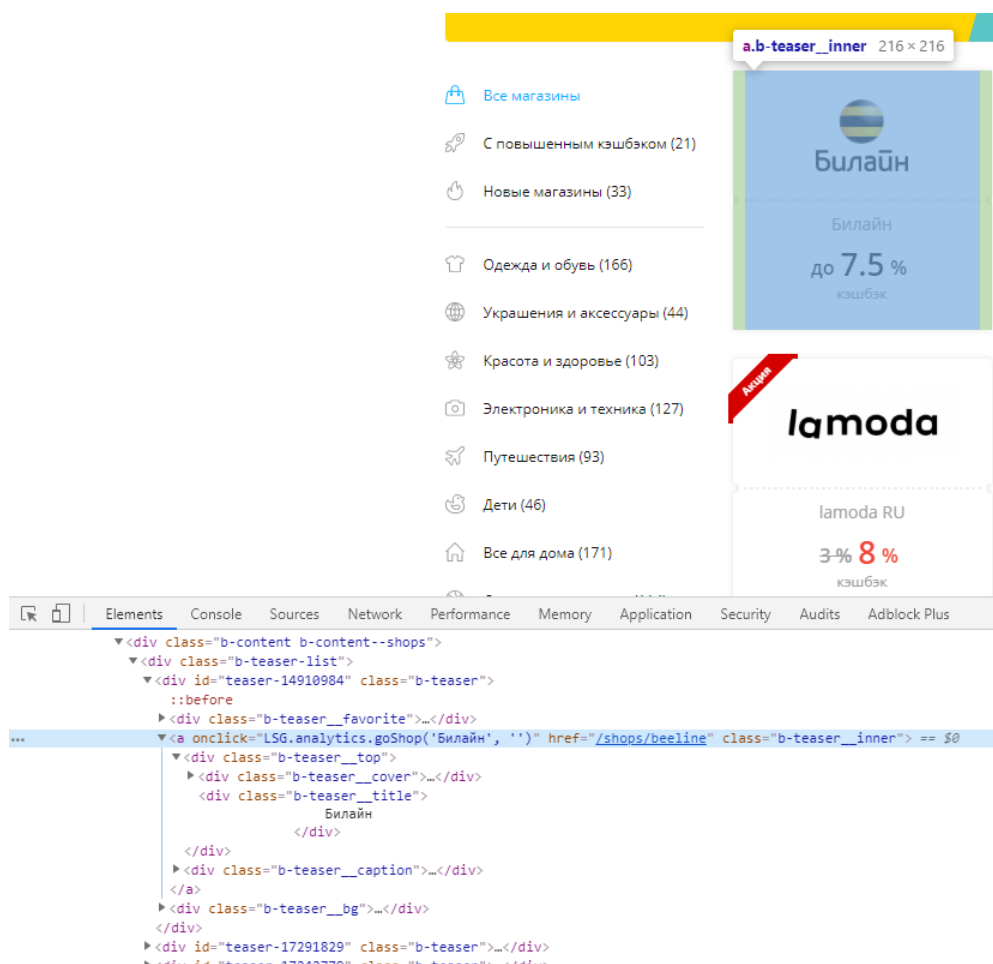


Рисунок 22 — выбор элемента с магазином на LetyShops

Из предыдущего рисунка видно, что элемент находится в тэге «a» 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.Selector("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 Рисунок 23, 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.Selector("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.Selector("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 23 – 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 Рисунок 23, 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);
```


8 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.

8.1 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 Рисунок 24.

Hello ADMIN!
[Sign Out](#)
Authenticated username:
admin
Authenticated user roles:
[ADMIN]

Обновить таблицу
[Update](#)









Упорядочить по возрастанию скидки
[Order](#)

Упорядочить по убыванию скидки
[Order](#)

Сохранить данные в Excel
[Save in Excel](#)

Сохранить данные в CSV
[Save in CSV](#)

Сохранить данные о времени в Excel
[Save time in Excel](#)

Name	Discount	Image	URL
СовкомБанк	4000.0 руб		Страница сайта
СовкомБанк	4000.0 руб		Страница сайта
ОСАГО и КАСКО от Ренессанс Страхование	2500.0 руб		Страница сайта
РКО Тинькофф Банк	2500.0 руб		Страница сайта
Восточный Банк	2500.0 руб		Страница сайта
Mr Doors	2500.0 руб		Страница сайта
ОСАГО и КАСКО от Ренессанс Страхование	2500.0 руб		Страница сайта
РКО Тинькофф Банк	2500.0 руб		Страница сайта

Отображение image.mythops.com...

Рисунок 24 – 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 Рисунок 25.

Обновить магазины	Скачать магазины в формате Excel	Скачать магазины в формате CSV	Скачать время парсинг
Real Madrid Shop	2,5 %		Ссылка на сайт
Manchester United Shop	2,5 %		Ссылка на сайт

Рисунок 25 – 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 Рисунок 26.

Обновить таблицу	Сохранить данные в Excel	Сохранить данные в CSV	Сохранить данные о времени в Excel
Обновить	Сохранить в Excel	Сохранить в CSV	Сохранить время в Excel
Яркий Фотомаркет	4.85 %		Ссылка на сайт
Яндекс.Афиша	1.6 %	Яндекс Афиша	Ссылка на сайт
Юный Папа	2.5 %		Ссылка на сайт

Рисунок 26 – вывод результата на страницу сайта, используя Django

8.2 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 [30]. Результат изображён на Рисунок 27.

	A	B	C	D
1	НАЗВАНИЕ	ЗНАЧЕНИЕ КЭШБЭКА	ВАЛЮТА	КАРТИНКА МАГАЗИНА
2	1-800-PetMeds	0 %		https://cdn.megabonus.com/images/shop_logo/1800petmeds.png
3	10 store	0,53 %		https://cdn.megabonus.com/images/shop_logo/10store.png
4	101 чай	6,4 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/101tea-logo.png
5	101 Чай	3,73 %		https://cdn.megabonus.com/images/shop_logo/101chay.png
6	101 чай	6,4 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/101tea-logo.png
7	101 Чай	3,73 %		https://cdn.megabonus.com/images/shop_logo/101chay.png
8	123.ru	4 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_14682498_cde9c276733dfdc
9	123.ru	3,73 %		https://cdn.megabonus.com/images/shop_logo/123ru.png
10	123.ru	4 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_14682498_cde9c276733dfdc
11	123spareparts.co.uk	3,19 %		https://cdn.megabonus.com/images/shop_logo/123spareparts.png
12	12go	2,66 %		https://cdn.megabonus.com/images/shop_logo/12goasia.png
13	1Dayfly	1,59 %		https://cdn.megabonus.com/images/shop_logo/1dayfly.png
14	1mmtt	250 pyb		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/1mmtt.jpg
15	1mmtt	250 pyb		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/1mmtt.jpg
16	2 Бера	2,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/2_beraga2.jpg
17	2 Бера	2,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/2_beraga2.jpg
18	21Shop	6,6 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/21shop1.png
19	21Shop	6,6 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/21shop1.png
20	21vek.by	4,09 %		https://cdn.megabonus.com/images/shop_logo/21vek.png
21	220 вольт	5,33 %		https://cdn.megabonus.com/images/shop_logo/220volt.png
22	220volt	2,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/220.png
23	220volt	2,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/220.png
24	24shop BY	0,85 %		https://cdn.megabonus.com/images/shop_logo/24shop.png
25	2Gud	1,59 %		https://cdn.megabonus.com/images/shop_logo/2gud.png
26	3avape	3 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/3avape.png
27	3avape	3 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/3avape.png
28	3Egon	6 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/3Egon.png

Рисунок 27 – Excel файл на Java

Для записи данных в Excel на языке программирования C# использовалась библиотека OfficeOpenXml [31]. Результат изображён на Рисунок 28.

	A	B	C	D
1	НАЗВАНИЕ	ЗНАЧЕНИЕ КЭШБЭКА	ВАЛЮТА	КАРТИНКА МАГАЗИНА
2	Real Madrid Shop	2,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291954_6c093af89923ca77258454d9a433db7
3	Manchester United Shop	2,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291953_cea5dc3f9b1070192ee121569daeecc
4	Insta360	7,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291941_a948c89c7b418c7654f06bfc58bc5046
5	ТОП Квестов	2,3 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291940_2b7b89e365a582cad4e9fda408154
6	Vivo	1,45 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291910_674d62ac61fa2e98c976cf63dbb69b
7	Exclaim	12 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291905_b1f39e578cdd72d5dbb80f14441543f
8	UltraVDS	14 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291880_2eebc70287f5a66e8aa34d0c5db0
9	Danaco	6,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291876_c5987d06f22cbb0f41e74ef92fe4ea
10	Роза Хутор	2,3 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291870_12d07754077afa80f5dfb1c18e1a19e
11	Charuel	3,5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291861_8e3e2bce50c5f9fb95fa9d98c341
12	Deppa	8,75 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291854_5838a1fcea9e21ad51c63c062356e2cd
13	BeTechno	0,29 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291848_4e30ade591e4d62421188cea12d3fd
14	RebelHouse	5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291847_4e30ade591e4d62421188cea12d3fd
15	Lilabjorn	5,75 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291822_1e20a43ac2d6f91b182ac9e666cfcc
16	Elari	4,35 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291821_523dbf3d28c3adfe44daae41d567a2
17	Aquanet	3 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291803_75d5706283aa08fb7dd51967c85d81
18	Hansgrohe	2 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291755_48967f9a59863bd79852b77fb562bft
19	Black+Decker	3,25 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291744_4a42f8d6f1ce31d1cdd496c3163742
20	Самоевары.рф	2,9 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291744_4a42f8d6f1ce31d1cdd496c3163742
21	Vaporl	7 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291692_48137a4fb9cc595d9f555f158f99ff
22	Sexcite	5,1 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291677_310443bac908a6ca7ac30f12ae23836
23	Grizzly	4 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291674_95e4034d1ce276a23fcd715e2b39065
24	EVO Impressions	280 pyb.		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291641_eacd7659b64a0004a473e182b2ecb2
25	Globol	4 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291640_7f29542e77b67329ab7ed5e7b8a3c4
26	Cosmogon	5 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/cosmogon-logo2.png
27	Rowenta	2 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/rowenta2.jpg
28	ВашаКомната	2 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/vashakomnata.png
29	Фондация	1 %		https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/fondaciya1.png

Рисунок 28 – Excel файл на C#

Для записи данных в Excel на языке программирования Django использовалась библиотека openpyxl [32]. Результат изображён на Рисунок 29.

А1	НАЗВАНИЕ	В	С	Д
1	НАЗВАНИЕ	ЗНАЧЕНИЕ	ВАЛЮТА	КАРТИНКА МАГАЗИНА
2	Яркий Фотомакет	4,85	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/yarkiy_fotomarket1.png
3	Яркий Фотомакет	4,85	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/yarkiy_fotomarket1.png
4	Яркий Фотомакет	5,21	%	https://cdn.megabonus.com/images/shop_logo/yarki.png
5	Яндекс.Афиша	1,73	%	https://cdn.megabonus.com/images/shop_logo/afisha.png
6	Яндекс.Афиша	1,6	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291216_b47d40ed0518cc4c6b9345b834ae1
7	Яндекс.Афиша	1,6	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291216_b47d40ed0518cc4c6b9345b834ae1
8	Яндекс Такси (для бизнеса)	266,53	RUB	https://cdn.megabonus.com/images/shop_logo/taxiyandex.png
9	Юный Папа	2,5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/yunyy_papa.png
10	Юный Папа	2,5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/yunyy_papa.png
11	Юный Папа	2,66	%	https://cdn.megabonus.com/images/shop_logo/yuniypapa.png
12	ЮниЗоо	2,5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_13436159_7757da89a60d8728d5370134a0a0b
13	ЮниЗоо	2,5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_13436159_7757da89a60d8728d5370134a0a0b
14	Юлмарт	3,5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/ulmart.jpg
15	Юлмарт	3,5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/ulmart.jpg
16	Юлмарт	4,1	%	https://cdn.megabonus.com/images/shop_logo/ulmart.png
17	Ювелирочка	3	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291911_e5f5bb6e4d561a930792bd24bfc60
18	Ювелирочка	3	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17291911_e5f5bb6e4d561a930792bd24bfc60
19	Ювелирочка	3,19	%	https://cdn.megabonus.com/images/shop_logo/uvellirka.png
20	Энергия	5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/energiya1.jpg
21	Энергия	5	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/energiya1.jpg
22	Эльдорадо RU	1,06	%	https://cdn.megabonus.com/images/shop_logo/eldologo.png
23	Эльдорадо RU	1	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17290879_ffa943f7b9c69ae000ae4a83af9cc
24	Эльдорадо RU	1	%	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_17290879_ffa943f7b9c69ae000ae4a83af9cc
25	Элизэ	134,86	RUB	https://cdn.megabonus.com/images/shop_logo/elize.png
26	Элизэ	150	руб.	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/elize.jpg
27	Элизэ	150	руб.	https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/elize.jpg

Рисунок 29 – Excel файл на Python

8.3 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 `орепcsv` [33]. Результат изображён на Рисунок 30.

А	В	С	Д	Е	Г	Н	И	К	М	О	Р	Т	У	В
1	1-800-PetMeds	"0.0 %"	"https://megabonus.com/shop/petmedscom"	"https://cdn.megabonus.com/images/shop_logo/1800petmeds.png"										
2	10 store	"0.53 %"	"https://megabonus.com/shop/10storecom"	"https://cdn.megabonus.com/images/shop_logo/10store.png"										
3	101 CTP	"P№, 6.4 %"	"https://letyshops.com/shops/101tea"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/101tea-logo.png"										
4	101 CTP	"P№, 6.4 %"	"https://letyshops.com/shops/101tea"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/101tea-logo.png"										
5	101 PpP	"P№, 3.73 %"	"https://megabonus.com/shop/tearu"	"https://cdn.megabonus.com/images/shop_logo/101chay.png"										
6	101 PpP	"P№, 3.73 %"	"https://megabonus.com/shop/tearu"	"https://cdn.megabonus.com/images/shop_logo/101chay.png"										
7	123.ru	"3.73 %"	"https://megabonus.com/shop/onetwothree.ru"	"https://cdn.megabonus.com/images/shop_logo/123ru.png"										
8	123.ru	"4.0 %"	"https://letyshops.com/shops/123"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_14682498_cde9c276733df0ff9135acc8d1759de_1543591360.jpg"										
9	123.ru	"4.0 %"	"https://letyshops.com/shops/123"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/shop/logo/shop_logo_14682498_cde9c276733df0ff9135acc8d1759de_1543591360.jpg"										
10	123spareparts.co.uk	"3.19 %"	"https://megabonus.com/shop/sparepartsco.uk"	"https://cdn.megabonus.com/images/shop_logo/123spareparts.png"										
11	12go	"2.66 %"	"https://megabonus.com/shop/goasia"	"https://cdn.megabonus.com/images/shop_logo/12goasia.png"										
12	1dayfly	"1.59 %"	"https://megabonus.com/shop/1dayflycom"	"https://cdn.megabonus.com/images/shop_logo/1dayfly.png"										
13	1mmtt	"250.0 CbCP"	"https://letyshops.com/shops/1mmtt"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/1mmtt.jpg"										
14	1mmtt	"250.0 CbCP"	"https://letyshops.com/shops/1mmtt"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/1mmtt.jpg"										
15	2 PpP	"CbPPIP", 2.5 %"	"https://letyshops.com/shops/2-berega"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/2_berega2.jpg"										
16	2 PpP	"CbPPIP", 2.5 %"	"https://letyshops.com/shops/2-berega"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/2_berega2.jpg"										
17	21Shop	"6.6 %"	"https://letyshops.com/shops/21shop"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/21shop1.png"										
18	21Shop	"6.6 %"	"https://letyshops.com/shops/21shop"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/21shop1.png"										
19	21vek.by	"4.09 %"	"https://megabonus.com/shop/vekby"	"https://cdn.megabonus.com/images/shop_logo/21vek.png"										
20	220 PIPs	"xCHC", 5.33 %"	"https://megabonus.com/shop/volt.ru"	"https://cdn.megabonus.com/images/shop_logo/220volt.png"										
21	220volt	"2.5 %"	"https://letyshops.com/shops/220volt"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/220.png"										
22	220volt	"2.5 %"	"https://letyshops.com/shops/220volt"	"https://image.letyshops.com/sites/default/files/styles/shop_logo_248x151/public/220.png"										
23	24shop BY	"0.85 %"	"https://megabonus.com/shop/shopby"	"https://cdn.megabonus.com/images/shop_logo/24shop.png"										

Рисунок 30 – CSV файл на Java

Для записи данных в CSV файл на языке программирования C# использовались стандартные средства в виде `StreamWriter`. Результат изображён на Рисунок 31.

9 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 прокси-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 [34].

При парсинге сайта может потребоваться войти в аккаунт, чтобы получить 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.

10 Заключение

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

11 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. NAP - Documentations— удобный .NET парсер HTML [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.Python: модуль requests [Электронный ресурс]: сайт. URL: <https://rtfm.co.ua/python-modul-requests/> (дата обращения: 22.05.2019). Загл. с экрана. Яз. рус.
- 30.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.
- 31.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.
- 32.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.
- 33.Opencsv Users Guide [Электронный ресурс]: документация. URL: <http://opencsv.sourceforge.net/#writing> (дата обращения: 22.05.2019). Загл. с экрана. Яз. англ.
- 34.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.

12 Приложение А. Реализация парсера с помощью JSoup

13 Приложение Б. Реализация парсера с помощью Selenium Java

14 Приложение В. Реализация парсера с помощью Unirest

15 Приложение Г. Реализация парсера с помощью Unirest для сайта Kopikot.ru

16 Приложение Д. Реализация парсера с помощью Unirest для сайта Cash4Brands.ru

17 Приложение Е. Реализация парсера с помощью HtmlAgilityPack

18 Приложение Ж. Реализация парсера с помощью Fizzler

19 Приложение 3. Реализация парсера с помощью AngleSharp

20 Приложение И. Реализация парсера с помощью CsQuery

21 Приложение К. Реализация парсера с помощью RestSharp

22 Приложение Л. Реализация парсера с помощью Selenium C#

23 Приложение М. Реализация парсера с помощью BeautifulSoup

24 Приложение Н. Реализация парсера с помощью Scrapy

25 Приложение О. Реализация парсера с помощью Selenium Python