|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| Кафедра КБ-14 «Цифровые технологии обработки данных» |

**ОТЧЕТ**

**по практической работе**

**«№2.** **РАЗРАБОТКА СКРИПТОВ ИЗВЛЕЧЕНИЯ ДАННЫХ ВЕБ-РЕСУРСОВ»**

**по дисциплине «Обработка данных с веб-ресурсов и поисковых машин»**

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил |  |  | Парамонов П.К.  *фамилия, имя, отчество* |
| шифр | 22Б0723 | группа | БСБО-05-22 |
|  |  |  |  |
| Проверил |  | к.т.н., доцент  *ученая степень, должность* | Ильин Д.Ю.  *фамилия, имя, отчество* |

**Москва 2024г.**

## **Цель практической работы**

Цель настоящей практической работы — научиться выполнять   
HTTP-запросы к веб-ресурсам и осуществлять извлечение данных с использованием CSS-селекторов.

## **Задачи практической работы**

Для достижения поставленной цели необходимо решить следующие задачи:

1. Определить вариант задания.
2. Сформировать проект на Node.js.
3. Установить программные библиотеки, соответствующие варианту задания.
4. Разработать программный код скриптов извлечения данных.
5. Выполнить сбор данных и сформировать датасеты.
6. Получить количественные оценки сформированного датасета.
7. Подготовить ответы на контрольные вопросы.
8. Составить отчет о проведенной работе.

## **Вариант задания**

Вариант выполнения задания 52.

## **Ход выполнения практической работы**

В ходе выполнения работы будут использоваться результаты первой практической. В связи с использованием библиотеки request, не получится парсить данные с сайтов с динамической загрузкой данных, а именно: Домклик, Яндек.недвижимость. В связи с этим, они были заменены на Мир квартир и Move.

В практической использовались библиотеки request и cheerio. Request для HTTP-запросов и cheerio для загрузки страниц.  
 Для удобства в дальнейших практических работах данные со всех сайтов будут браться по шаблону

1. Название;
2. Цена;
3. Месторасположение;
4. Станция метро;
5. Описание;
6. Ссылка.

Далее будут приведены примеры данных с каждого сайта, листинг кода для каждого парсера будут представлены в приложении А.

**Авито**

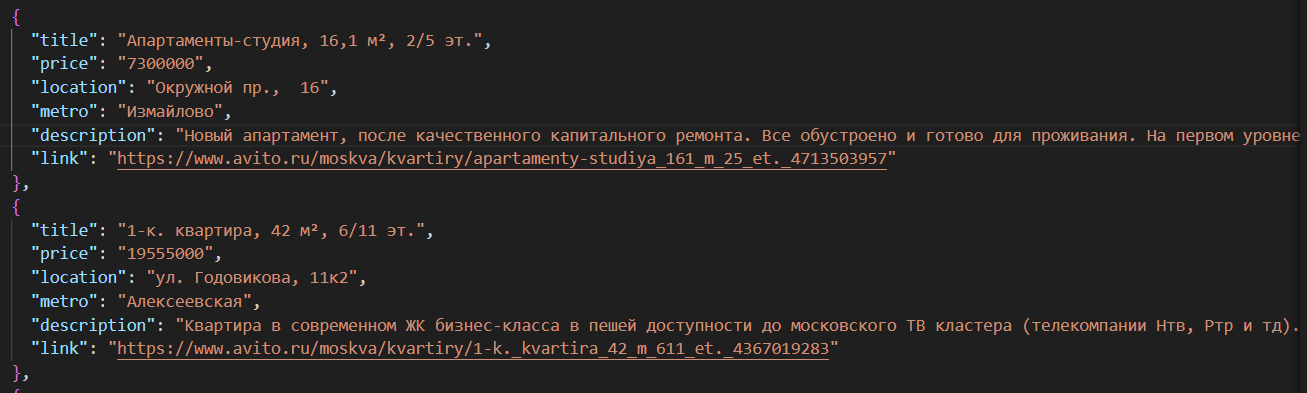
****

Рис 1 – пример данных Авито

**Циан**

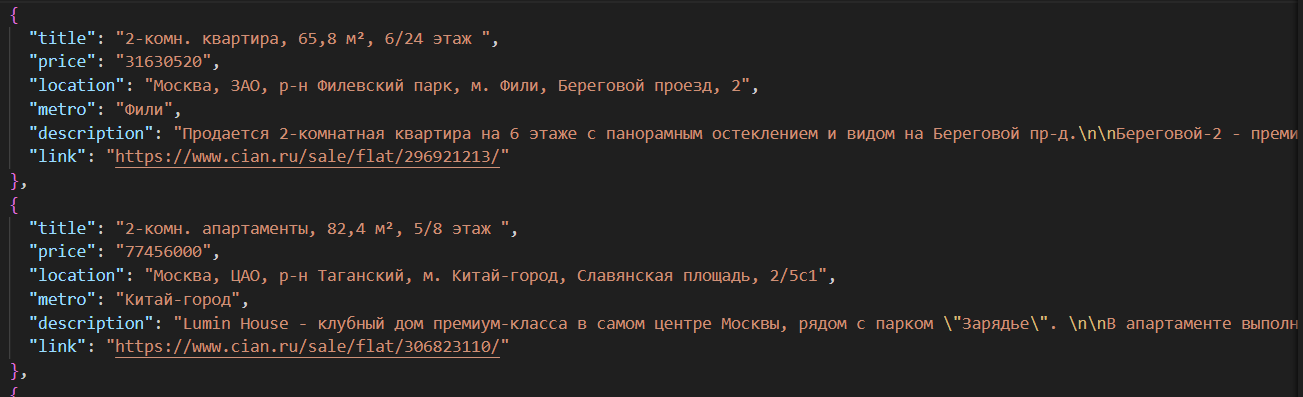
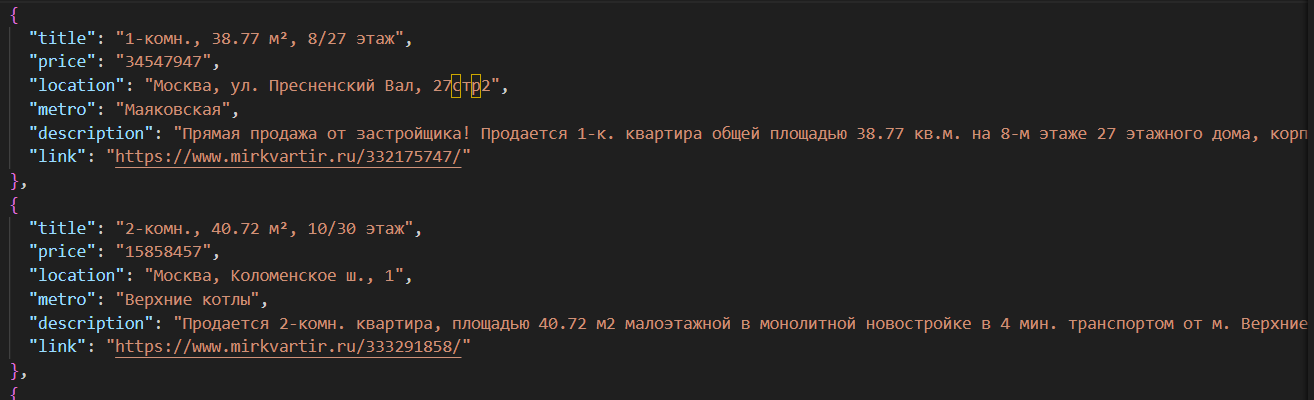


Рис 2 – пример данных Циан

**Мир Квартир**

****Рис 3 – пример данных Мира квартир

**Move**

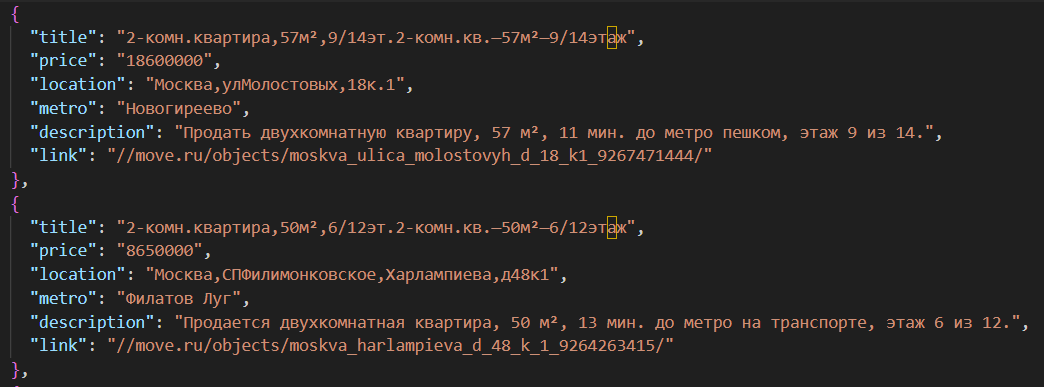
****

Рис 4 – пример данных Move

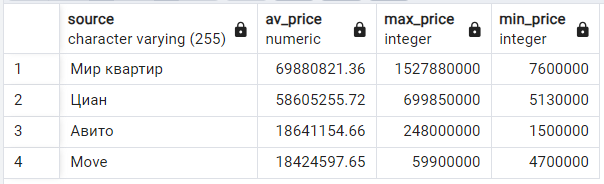
Для анализа данных воспользуемся результатами 3 практической  


Рис 5 – Анализ цен по сайтам



Рис 6 – распределение объявлений по станциям метро

## **Трудности в ходе выполнения практической работы**

Основные трудности возникли при попытке парсить данные с сайтов с динамической загрузкой данных, из-за этого пришлось поменять несколько сайтов из первой практической и прописать новые селекторы. Следующая трудность возникла при составление заголовков http-запросов, для того чтобы сайт принимал за человека и не блокировал запросы. Для каждого сайта нужно было продумывать собственные заголовки.

**Ответы на контрольные вопросы**

**1. В чем различия синхронного и асинхронного исполнения кода в JavaScript?**

* Синхронное исполнение: Операции выполняются последовательно, каждая следующая ожидает завершения предыдущей. Если одна операция "зависнет", весь код остановится.
* Асинхронное исполнение: Операции могут выполняться параллельно. Например, выполнение запроса к серверу не блокирует остальные части программы. Асинхронность достигается через callbacks, Promises или async/await.

### **2. Какие количественные характеристики могут описывать содержание датасета?**

* **Размер датасета:** Общее количество записей (строк).
* **Количество признаков (колонок):** Число уникальных параметров, описывающих записи.
* **Пропуски:** Процент или количество отсутствующих значений.
* **Среднее, медиана, мода:** Для числовых данных.
* **Диапазон:** Разница между максимальным и минимальным значениями.
* **Скорая корреляция:** Для анализа взаимосвязи между числовыми признаками.

**3. В чем состоят отличия работы библиотеки Cheerio от работы браузера при обработке HTML-страниц?**

* **Cheerio**:
  + Обрабатывает только HTML-текст.
  + Не выполняет JavaScript на странице.
  + Быстрее и легче, так как не загружает ресурсы (CSS, изображения).
* **Браузер**:
  + Полностью рендерит страницу.
  + Выполняет встроенные скрипты и загружает все ресурсы, что может потребовать больше времени и ресурсов.

**4. Какие методы обработки текста можно применить в случае, если селекторы не позволяют извлечь желаемое атомарное значение с веб-ресурса (элемент HTML содержит посторонние данные)?**

* Регулярные выражения (RegEx): Используются для извлечения нужной информации из сложного текста.
* Методы работы со строками:
  + split() — для разделения текста.
  + substring() / slice() — для извлечения нужных частей.
  + trim() — для удаления лишних пробелов.
* Поиск по ключевым словам: Проверка текста на наличие определённых паттернов.
* Предобработка текста: Удаление HTML-тегов, лишних символов и т.д.

**5. В чем отличия программных библиотек Axios, Got, Needle, Node-fetch, Request, Superagent?**

| **Библиотека** | **Особенности** |
| --- | --- |
| **Axios** | Простота, поддержка Promise, работает с JSON по умолчанию, автоматическое управление тайм-аутами и заголовками. |
| **Got** | Высокая производительность, лучше подходит для больших запросов, удобна для работы с потоками. |
| **Needle** | Лёгкая, поддерживает низкоуровневые HTTP-запросы, быстрое преобразование данных. |
| **Node-fetch** | Простая и минималистичная библиотека, подходит для работы с fetch в Node.js. |
| **Request** | Устаревшая библиотека, которая раньше была популярна для HTTP-запросов. |
| **Superagent** | Удобный интерфейс, поддержка Promise, хороший выбор для простых API-запросов. |

## **Вывод**

В ходе выполнения работы «Разработка скриптов извлечения данных веб-ресурсов» были рассмотрены особенности работы с библиотекой Request, Cheerio, благодаря которым удалось собрать данные с веб-ресурсов таких как: Авито, Мирквартир, Move и Циан.

**Приложение А. Листинг кода**

Листинг кода для Авито

const request = require('request');

const cheerio = require('cheerio');

const fs = require('fs');

const { hrtime } = require('process');

// URL страницы, которую нужно спарсить

const BaseUrl = 'https://www.avito.ru/moskva/kvartiry/prodam-ASgBAgICAUSSA8YQ?context=H4sIAAAAAAAA\_0q0MrSqLraysFJKK8rPDUhMT1WyLrYyNLNSKk5NLErOcMsvyg3PTElPLVGyrgUEAAD\_\_xf8iH4tAAAA';

function sleep(ms){

    return new Promise(resolve => setTimeout(resolve,ms));

}

async function parsePages(url, pages){

    let results = [];

    for( let i = 1; i < pages+1; i++){

        newUrl = url + `&p=${i}`;

        results.push(parsePage(newUrl));

        await sleep(5000)

        console.log(`${i} page parsed`)

        console.log(results);

    }

    results = results.flat();

    // Сохраняем результат в JSON-файл

    fs.writeFile('Jsons/data-avito.json', JSON.stringify(results, null, 2), (err) => {

        if (err) throw err;

        console.log('Данные успешно сохранены в data-avito.json');

    });

}

function parsePage(url) {

    const options =

    {

        url,

        headers: {

            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36',

            'Accept-Language': 'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7',

        },

    };

    let resList =[]

    request(options, (error, response, html) => {

        if (!error && response.statusCode === 200) {

            const $ = cheerio.load(html);

            // Парсим данные по селекторам

            $('div[data-marker="catalog-serp"] > div[data-marker="item"]').each((index, element) => {

                const title = $(element).find('div[class^="iva-item-title-"]').text().trim();

                const price = $(element).find('p[data-marker="item-price"]').text().trim().replace(/\s/g,'').slice(0,-1);

                const location = $(element).find('div[class^="geo-root-"]>p>span[title]').text().trim();

                const metro = $(element).find('div[class^="geo-root-"]>p>a').text().trim();

                const description = $(element).find('div[class\*="item-description"] > p').text().trim();

                const link = "https://www.avito.ru" + $(element).find('a[itemprop="url"][data-marker="item-title"]').attr('href');

                // Добавляем объект с данными в массив

                resList.push({ title, price, location, metro, description, link});

            });

        } else {

            console.error('Ошибка при загрузке страницы:', error);

        }

    })

    return resList;

}

parsePages(BaseUrl, 5)

Листинг кода для Циан

const request = require('request');

const cheerio = require('cheerio');

const fs = require('fs');

const { console } = require('inspector');

// URL страницы, которую нужно спарсить

const BaseUrl = 'https://www.cian.ru/cat.php?deal\_type=sale&engine\_version=2&offer\_type=flat&region=1&room1=1&room2=1'

function sleep(ms){

    return new Promise(resolve => setTimeout(resolve,ms));

}

async function parsePages(url, pages){

    let results = [];

    for( let i = 1; i < pages+1; i++){

        newUrl = url + `&p=${i}`;

        results.push(parsePage(newUrl));

        await sleep(5000)

        console.log(`${i} page parsed`)

        console.log(results);

    }

    results = results.flat();

    // Сохраняем результат в JSON-файл

    fs.writeFile('Jsons/data-cian.json', JSON.stringify(results, null, 2), (err) => {

        if (err) throw err;

        console.log('Данные успешно сохранены в data-cian.json');

    });

}

function parsePage(url) {

    const options =

    {

        url,

        encoding: null,

        gzip: true,

        headers: {

            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 YaBrowser/24.10.0.0 Safari/537.36',

            'Accept-Language': 'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7',

            'Connection': 'keep-alive',

            'Accept-Encoding': 'gzip, deflate, br, zstd',

        },

    };

    let resList = [];

    request(options, (error, response, html) => {

        if (!error && response.statusCode === 200) {

            const $ = cheerio.load(html);

            // Парсим данные по селекторам

            $('div[data-testid="offer-card"] > div').each((index, element) => {

                const title = $(element).find('span[data-mark="OfferTitle"] > span').text().trim() + ' ' + $(element).find('span[data-mark="OfferSubtitle"] > span').text().trim();

                const price = $(element).find('span[data-mark="MainPrice"]').text().trim().replace(/\s/g,'').slice(0,-1);

                const location = $(element).find('div[class\*="labels"]').text().trim();

                const metro = $(element).find('div[data-name="SpecialGeo"] > a[class\*="link"] div').text().trim();

                const description = $(element).find('div[data-name="Description"] > p').text().trim();

                const link = $(element).find('div[data-name="LinkArea"] > a').attr('href');

                // Добавляем объект с данными в массив

                resList.push({ title, price, location, metro, description, link });

            });

        } else {

            console.error('Ошибка при загрузке страницы:', error);

            console.error('Код ошибки:', response.statusCode );

        }

    })

    return resList;

}

parsePages(BaseUrl, 5)

Листинг кода для Мира квартир

const request = require('request');

const cheerio = require('cheerio');

const fs = require('fs');

const { console } = require('inspector');

// URL страницы, которую нужно спарсить

const BaseUrl = encodeURI('https://www.mirkvartir.ru/Москва/');

function sleep(ms){

    return new Promise(resolve => setTimeout(resolve,ms));

}

async function parsePages(url, pages){

    let results = [];

    for( let i = 1; i < pages+1; i++){

        newUrl = url + `?p=${i}`;

        results.push(parsePage(newUrl));

        await sleep(5000)

        console.log(`${i} page parsed`)

        console.log(results);

    }

    results = results.flat();

    // Сохраняем результат в JSON-файл

    fs.writeFile('Jsons/data-mirkv.json', JSON.stringify(results, null, 2), (err) => {

        if (err) throw err;

        console.log('Данные успешно сохранены в data-mirkv.json');

    });

}

function parsePage(url, page = 1) {

    const options =

    {

        url,

        encoding: null,

        gzip: true,

        headers: {

            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 YaBrowser/24.10.0.0 Safari/537.36',

            'Accept-Language': 'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7',

            'Connection': 'keep-alive',

            'Accept-Encoding': 'gzip, deflate, br, zstd',

        },

    };

    let resList = [];

    request(options, (error, response, html) => {

        if (!error && response.statusCode === 200) {

            const $ = cheerio.load(html);

            // Парсим данные по селекторам

            $('div[class\*="OffersListItem\_infoContainer"]').each((index, element) => {

                const title = $(element).find('div[class\*="OffersListItem\_offerTitleContainer"] > a > span').text().trim();

                const price = $(element).find('div[class\*="OffersListItem\_rightContainer"] span[class\*="OfferPrice\_price\_"] ').text().trim().replace(/\s/g,'').slice(0,-1);

                const location = $(element).find('div[class\*="OfferAddress\_address"]').text().trim();

                const metro = $(element).find('div[class\*="PlaceAround"] > a > span').text().trim();

                const description = $(element).find('div[class\*="OffersListItem\_infoText"]').text().trim();

                const link = $(element).find('a[class\*="OffersListItem\_offerTitle"]').attr('href');

                // Добавляем объект с данными в массив

                resList.push({ title, price, location, metro, description, link });

            });

        } else {

            console.error('Ошибка при загрузке страницы:', error);

            console.error('Код ошибки:', response.statusCode );

        }

    })

    return resList;

}

parsePages(BaseUrl, 5)

Листинг кода для Move  
const request = require('request');

const cheerio = require('cheerio');

const fs = require('fs');

const { console } = require('inspector');

// URL страницы, которую нужно спарсить

const BaseUrl = 'https://move.ru/kvartiry/prodazha\_dvuhkomnatnih\_kvartir/';

function sleep(ms){

    return new Promise(resolve => setTimeout(resolve,ms));

}

async function parsePages(url, pages){

    let results = [];

    for( let i = 1; i < pages+1; i++){

        newUrl = url + `?page=${i}&limit=30`;

        results.push(parsePage(newUrl));

        await sleep(5000)

        console.log(`${i} page parsed`)

        console.log(results);

    }

    results = results.flat();

    // Сохраняем результат в JSON-файл

    fs.writeFile('Jsons/data-move.json', JSON.stringify(results, null, 2), (err) => {

        if (err) throw err;

        console.log('Данные успешно сохранены в data-move.json');

    });

}

function parsePage(url) {

    const options =

    {

        url,

        headers: {

            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 YaBrowser/24.10.0.0 Safari/537.36',

            'Accept-Language': 'ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7',

            'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8',

            'Connection': 'keep-alive',

            'Upgrade-Insecure-Requests': '1'

        },

    };

    let resList = [];

    request(options, (error, response, html) => {

        if (!error && response.statusCode === 200) {

            const $ = cheerio.load(html);

            // Парсим данные по селекторам

            $('div[class="search-item move-object"]').each((index, element) => {

                const title = $(element).find('div[class="search-item\_\_header-left"] a[class\*="\_\_title-link search"]').text().trim().replace(/\s/g,'');

                const price = $(element).find('span[class="search-item\_\_price-first"]').text().trim().replace(/\s/g,'').slice(0,-1);

                const location = $(element).find('div[class="geo-block\_\_geo-info"]').text().trim().replace(/\s/g,'');

                const metro = $(element).find('span[class\*="search-item\_\_metro"]').text().trim();

                const description = $(element).find('p[class="search-item\_\_description "]').text().trim();

                const link = $(element).find('a[class="search-item\_\_link-overlay"]').attr('href');

                // Добавляем объект с данными в массив

                resList.push({ title, price, location, metro, description, link });

            });

        } else {

            console.error('Ошибка при загрузке страницы:', error);

            console.error('Код ошибки:', response.statusCode );

        }

    })

    return resList;

}

parsePages(BaseUrl, 5)