Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

Парсер пошукової системи Bing та аналіз навантажувальних проблем

Текстова частина до курсової роботи
за спеціальністю „Комп'ютерні науки"

Керівник курсової роботи
Доктор тех. наук Глибовець А. М.

_____
(підпис)
"____" _____ 2021 р.

Виконав студент Костенко Я. В.
"____" _____ 2021 р.

Київ 2021

Зміст

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри мережних технологій,
доктор технічних наук
_____А. М. Глибовець
(підпис)
„____”_____2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студенту Костенку Ярославу Володимировичу
факультету інформатики 4-го курсу

ТЕМА: “Парсер пошукової системи Bing та аналіз навантажувальних проблем”

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1 Огляд парсерів та способів їх використання

2 Огляд існуючих рішень та розробка парсеру пошукової системи Bing

3 Аналіз навантажувальних проблем

Висновки

Список літератури

Дата видачі „___” _____ 2021 р. Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Календарний план виконання курсової роботи

Тема: “ Парсер пошукової системи Bing та аналіз навантажувальних проблем ”

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	15.10.2020	
2.	Огляд технічної літератури за темою роботи	15.01.2021	
3.	Аналіз сучасних методів	29.02.2021	
4.	Написання теоретичної частини	15.03.2021	
5.	Створення застосунку	01.04.2021	
6.	Написання практичної частини	04.04.2021	
7.	Корегування курсової роботи	09.04.2021	
8.	Створення слайдів для доповіді та написання доповіді	11.04.2021	
9.	Захист курсової роботи	19.04.2021	

Студент _____

Керівник _____

“_____” _____

Анотація

В даній курсовій роботі розглянуто способи парсингу пошукової системи Bing, розроблено парсер пошукової системи Bing та проаналізовано проблеми, які виникли під час розробки. Метою роботи було розробити власний парсер пошукової системи Bing та проаналізувати навантажувальні проблеми. Були отримані необхідні результати і зроблено висновки.

Вступ

На сьогоднішній день пошукові системи збирають і видають величезні обсяги інформації, яка може бути важливою для певних груп користувачів. Тому автоматизований парсинг і обробка даних є актуальним завданням.

Зацікавленість в парсингу проявляють різні IT-компанії, які вкладають інвестиції в розвиток та розробку методів парсингу. Тому вже існує багато різноманітних сервісів, програм, бібліотек, які пропонують користувачу отримати всі дані, котрі його цікавлять за лічені хвилини.

Курсова робота складається з трьох розділів.

Перший розділ присвячено загальному огляду парсингу, способів його використання, видам та обмеженням в роботі парсерів.

В другому розгляді розглянуто існуючі рішення парсингу пошукової системи Bing і описується створення власного парсеру та наводяться результати роботи.

У третьому розділі проводиться аналіз навантажувальних проблем та пропонуються можливі способи їх вирішення.

Створено веб-додаток для парсингу пошукової системи Bing, який дозволяє користувачу за короткий термін отримати результати.

Розділ 1: Парсинг

1.1 Що таке парсер

В сучасному світі інформація є найціннішим ресурсом, а її найбільшим джерелом є інтернет. В таких випадках мова йде про величезні обсяги даних, обробити які вручну – неможливо. Тому автоматизований збір і обробка даних є надзвичайно корисним і важливим процесом. Для швидкої обробки інформації використовуються парсери.

Парсер – це програмне забезпечення, яке здатне швидко обробляти інформацію згідно з алгоритмом і повертати потрібний результат. Він є ефективним рішенням для автоматизації збору і зміни інформації. Парсер здатний швидко обходити сотні веб-сторінок, знаходити необхідну інформацію і повертати її в визначеному вигляді.

Найпоширенішими парсерами є пошукові роботи, які використовуються пошуковими системами. Вони аналізують сторінки, зберігають інформацію про них в базі даних і потім, під час пошуку, повертають користувачу всю актуальну інформацію.

Для розробки парсерів використовується багато мов програмування, серед яких: Python, JavaScript, Java, C#, Perl, Ruby, Go та інші.

1.2 Для чого використовуються

Переваг у використанні парсерів багато, серед них:

- Автоматизація процесів.
- Висока швидкість.
- Висока продуктивність.

Проаналізуємо, для чого можна використовувати парсер:

- Актуальність інформації. Наприклад, користувачам не потрібні вчорашні новини і сенсації. Також, коли мова йде про сайти-

обмінники валют, де необхідно змінювати інформацію про курс валют по декілька разів на день, створення парсера є надзвичайно потрібним. В таких випадках парсер буде цілодобово відстежувати зміну курсу валют в Нацбанку.

- Автоматичне оновлення інформації. Усім ресурсам необхідне регулярне оновлення інформації, але здійснювати це вручну - не завжди є доцільно і можливо.
- Наповнення даними. Парсер здатний зібрати дані з великої кількості ресурсів і наповнити ними нашу базу даних.
- Централізація даних. За допомогою парсеру можна зібрати і об’єднати дані на одну тему, які розкидані по безлічі Інтернет-ресурсів. [3]

Отже, парсер корисний при роботі з великими обсягами даних, які потрібно швидко обробити і систематизувати.

1.3 Види парсингу

Краулінг (crawling)– це процес сканування сайту автоматизованою системою. [4] Одним з прикладів краулінгу є пошуковий робот. Пошуковий робот (web crawler) – програма, що є складовою частиною пошукової системи та призначена для обходу сторінок інтернету з метою занесення інформації про них (ключових слів) до бази даних. Він здійснює загальний пошук інформації в Інтернеті і повідомляє про зміст знайденого документа, індексує його й добуває підсумкову інформацію. [5]

Веб-скрапінг – перетворення у структуровані дані інформації з веб-сторінок, які призначені для перегляду людиною за допомогою браузера. Як правило виконується за допомогою комп’ютерних програм, що імітують поведінку людини в Інтернеті, або з’єднуючись з веб-сервером напряму, або керуючи повноцінним веб-браузером. [6] Наш парсер, який буде розроблятися в подальшому, є саме веб-скрапінгом.

1.4 Обмеження парсингу

1.4.1 Robots.txt

З точки зору розробника веб-сайтів, парсери можна сприймати як неприємність. Вони маскуються під справжніх відвідувачів веб-сайту і можуть здійснювати багато запитів, що значно збільшує навантаження на сервер. [1]

Для того, щоб контролювати кількість та типи поданих запитів, багато доменів містять robots.txt файл, який повідомляє розробникам, як вони повинні взаємодіяти з їхнім сайтом.

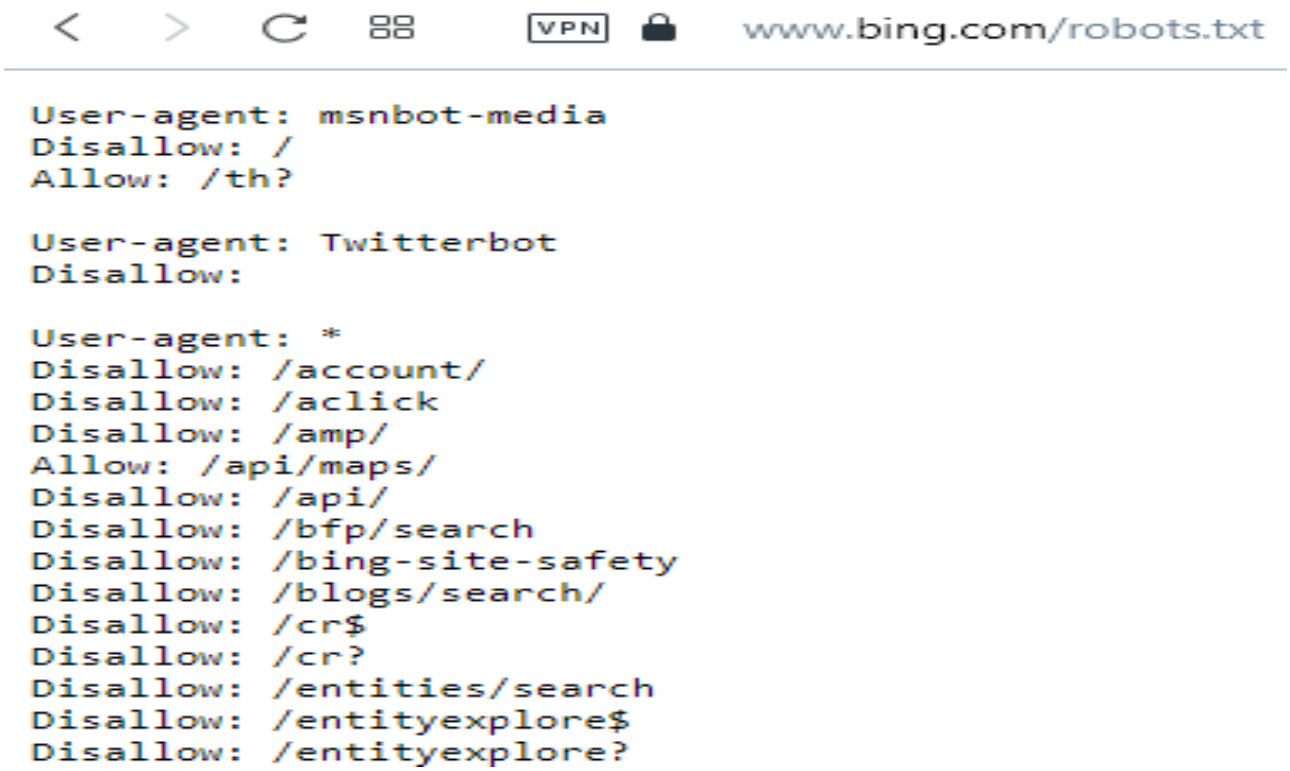

Рисунок 1.1 – Приклад robots.txt пошукової системи Bing

Більшість основних пошукових систем таких як: Baidu, Bing, Google, Yahoo виконують даний файл. Проте файл є суто консультативним і спирається на чесність розробників парсеру. [2]

1.4.2 Протокол Sitemaps

Протокол Sitemaps дозволяє веб-розробникам інформувати парсери про URL-адреси веб-сайту, які доступні для сканування. Також дозволяє включати додаткову інформацію про кожну URL-адресу: коли вона востаннє оновлювалася, як часто вона змінюється та наскільки є важливою у відношенні до інших URL-адрес веб-сайту. Даний протокол, наприклад, дозволяє пошуковим системам ефективніше сканувати сайт та знаходити URL-адреси, які можуть бути ізольованими від решти вмісту сайту. Протокол Sitemaps – це протокол включення або виключення URL-адреси та доповнення до robots.txt.

Формат протоколу Sitemap складається з XML тегів. Сам файл повинен бути закодований UTF-8. Sitemaps також можуть бути просто текстовим списком URL-адрес стиснутим у форматі .gz. Максимальний розмір файлу Sitemaps становить 50 Мб або 50 000 URL-адрес. [11]

Так само, як і robots.txt, даний протокол носить лише консультативний характер і спирається на чесність розробників.

Зразок файлу Sitemaps, який містить лише одну URL-адресу та використовує всі додаткові, показаний нижче.

```
<?xml version="1.0" encoding="utf-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9 http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd">
    <url>
        <loc>http://example.com/</loc>
        <lastmod>2006-11-18</lastmod>
        <changefreq>daily</changefreq>
        <priority>0.8</priority>
    </url>
</urlset>
```

Рисунок 1.2 – Приклад протоколу Sitemaps

1.4.3 Captcha

На відміну від вищенаведених обмежень, даний метод є одним із найефективніших та найбільш часто використовуваних способів захисту від частих автоматизованих запитів.

Captcha (повністю автоматизований публічний тест Тюринга для розрізнення комп’ютерів та людей) – комп’ютерний тест типу запит-відповідь, який використовується для визначення, хто користується системою – комп’ютер чи людина. [12]

У найпоширеніших варіантах Captcha, від користувача вимагається введення символів, які зображені в спотвореному вигляді на пропонованому малюнку, інколи з доданням шуму або напівпрозорості. Також одним з поширених варіантів є розпізнавання зображень.

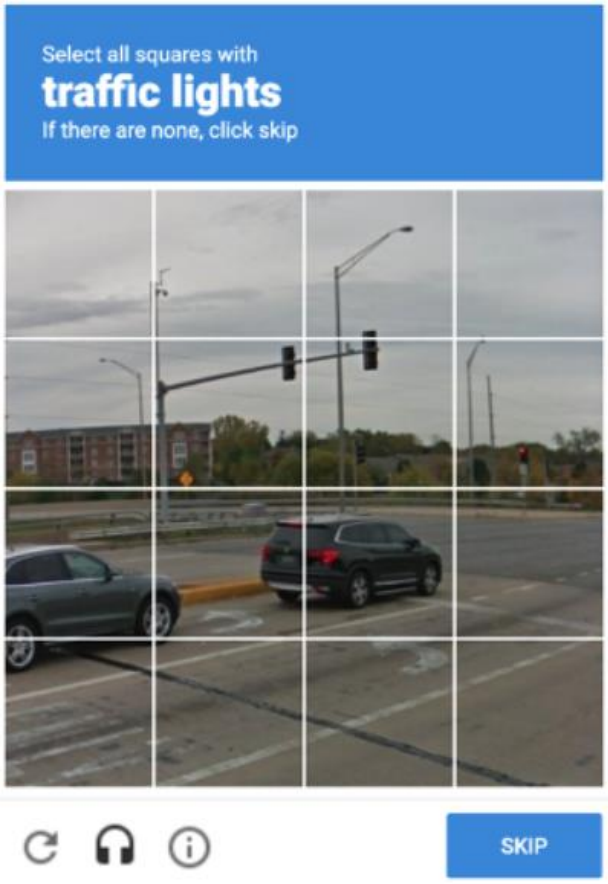

Рисунок 1.3 – Приклад Captcha

Також багато веб-ресурсів використовують власні механізми захисту від частих запитів, адже вони збільшують навантаження на сервер і сповільнюють швидкість взаємодії з справжніми користувачами.

Розділ 2: Парсинг пошукової системи Bing

2.1 Bing API

В Microsoft Azure наявний сервіс під назвою Bing Web Search API, який дозволяє отримувати результати пошуку Bing.

Bing Web Search API – Restful сервіс, який надає миттєві відповіді на запити користувачів. Результати пошуку легко налаштовуються на веб-сторінки, зображення, відео, новини, переклади тощо. Результати надаються у форматі JSON на основі релевантності пошуку.

Також даний сервіс надає ряд функцій, які дозволяють налаштувати результати пошуку:

- Пошукові терміни в режимі реального часу. Відображає запропоновані пошукові терміни під час їх введення.
- Фільтрування та обмеження результатів за типом вмісту. Налаштовує та уточнює результати пошуку за допомогою фільтрів та параметрів запитів для веб-сторінок, зображень, відео, безпечного пошуку тощо.
- Локалізація результатів пошуку за країною або регіоном.
- Аналіз показників пошуку за допомогою статистики Bing. Забезпечує аналітику найпопулярніших рядків запитів, географічного розподілу тощо.

Bing API легко викликати з будь-якої мови програмування, яка здатна робити HTTP-запити та аналізувати відповіді JSON. Сервіс доступний за допомогою REST API або клієнтських бібліотек Bing Web Search. [7]

```java
public static SearchResults SearchWeb (String searchQuery) throws Exception {
    // Construct the URL.
    URL url = new URL(host + path + "?q=" +  URLEncoder.encode(searchQuery, "UTF-8"));

    // Open the connection.
    HttpsURLConnection connection = (HttpsURLConnection)url.openConnection();
    connection.setRequestProperty("Ocp-Apim-Subscription-Key", subscriptionKey);

    // Receive the JSON response body.
    InputStream stream = connection.getInputStream();
    String response = new Scanner(stream).useDelimiter("\\A").next();

    // Construct the result object.
    SearchResults results = new SearchResults(new HashMap<String, String>(), response);

    // Extract Bing-related HTTP headers.
    Map<String, List<String>> headers = connection.getHeaderFields();
    for (String header : headers.keySet()) {
        if (header == null) continue;      // may have null key
        if (header.startsWith("BingAPIs-") || header.startsWith("X-MSEdge-")){
            results.relevantHeaders.put(header, headers.get(header).get(0));
        }
    }
    stream.close();
    return results;
}
```

Рисунок 2.1 – Приклад коду для підключення та парсингу за допомогою Bing Web Search API

За допомогою даного коду ми можемо отримати 10 результатів пошукової системи Bing.


```
"id": "https://api.cognitive.microsoft.com/api/v7/#RelatedSearches",
"value": [
  {
    "text": "microsoft bot framework",
    "displayText": "microsoft bot framework",
    "webSearchUrl": "https://www.bing.com/search?q=microsoft+bot+framework"
  },
  {
    "text": "microsoft cognitive services youtube",
    "displayText": "microsoft cognitive services youtube",
    "webSearchUrl": "https://www.bing.com/search?q=microsoft+cognitive+services+youtu
  },
  {
    "text": "microsoft cognitive services search api",
    "displayText": "microsoft cognitive services search api",
    "webSearchUrl": "https://www.bing.com/search?q=microsoft+cognitive+services+searc
  },
  {
    "text": "microsoft cognitive services news",
    "displayText": "microsoft cognitive services news",
    "webSearchUrl": "https://www.bing.com/search?q=microsoft+cognitive+services+news"
  },
  {
    "text": "ms cognitive service",
    "displayText": "ms cognitive service",
    "webSearchUrl": "https://www.bing.com/search?q=ms+cognitive+service"
  },
```


Рисунок 2.2 – Приклад отриманих результатів парсингу

На рисунку 2.2 знаходиться приклад отриманих результатів, серед яких url сторінки та її назва. [9]

При користуванні даним сервісом необхідно пам’ятати, що він не є безкоштовним і існує ряд обмежень. Безкоштовною є тільки 1000 запитів в місяць, після чого доведеться заплатити від 1 до 7 доларів, в залежності від обраного плану, за кожну наступну 1000 запитів. Також є обмеження по кількості запитів в секунду. Безкоштовними є 3 запити за секунду, для їхнього збільшення також доведеться доплатити. Крім того, можна отримати лише перші 100 результатів пошуку і даний параметр є незмінним у будь-якому тарифі. [8]

Отже, Bing Web Search API є готовим рішенням парсингу пошукової системи Bing, яке може зацікавити розробників, які не бажають витрачати власні обчислювальні ресурси на парсинг пошукової системи, або при роботі з невеликими об’ємами даних.

2.2 Розробка власного парсера

2.2.1 Вимоги

При парсингу пошукової системи Bing, ми будемо намагатись отримати такі дані:

- Уніфікований локатор ресурсу.
- Snippet. Короткий опис веб-сторінки.
- Позицію веб-сторінки в пошуку.
- 50 результатів відповіді.

З одного ключового слова або фрази, ми отримуємо 50 результатів, які потрібно обробити і записати в базу даних.

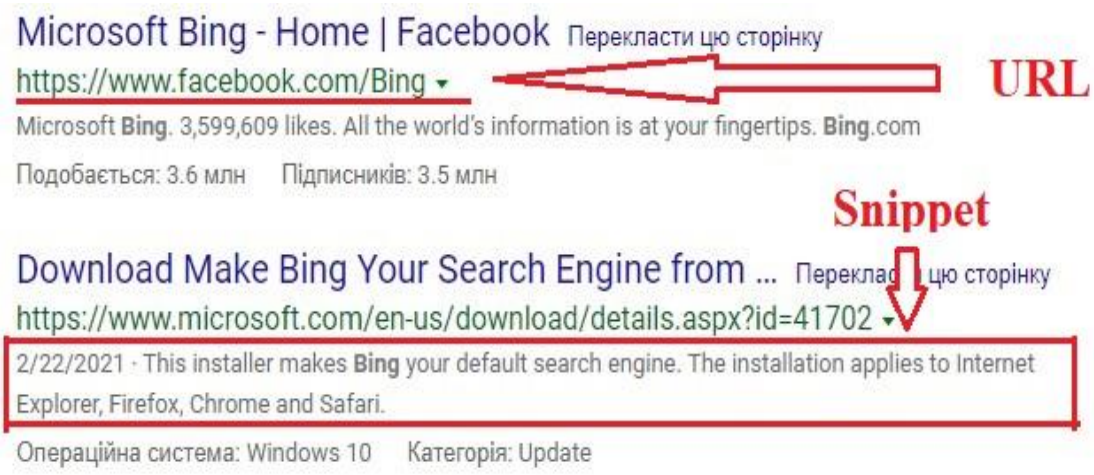

Рисунок 2.3 – Приклад потрібних даних

2.2.2 Схема бази даних

Для того, щоб зберігати дані в базі даних, була розроблена наступна ER-модель.

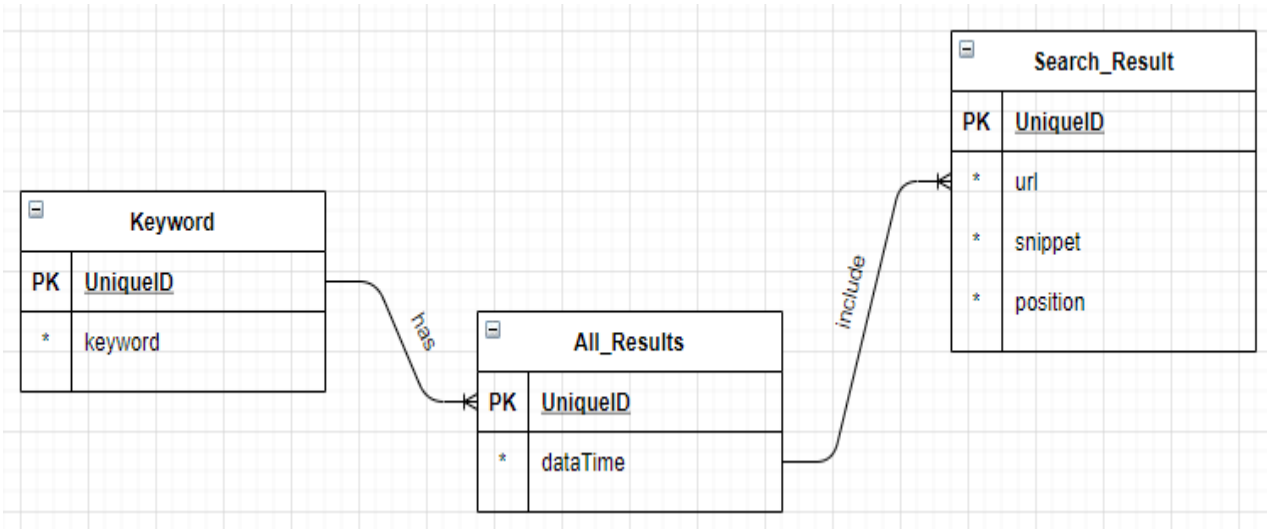

Рисунок 2.4 – ER-модель бази даних

Сутність Keyword потрібна для збереження ключового слова, для якого ми знарсили результати пошуку.

Сутність All_Results використовується для збереження дати та часу парсингу конкретного ключового слова. Ця інформація потрібна для аналізу та порівняння змін отриманих результатів.

Сутність Search_Result зберігає всі необхідні дані з одного отриманого результату.

2.2.3 Парсинг

Для парсингу пошукової системи Bing була використана мова програмування Java. З її допомогою ми відправляємо запит на сервер Bing та отримуємо відповідь у форматі HTML.

При відправлення запиту до сервера Bing потрібно задати такий параметр, як User Agent. Для успішного запиту він є ключовим значенням, оскільки без нього ваш запит буде заблокований або ви отримаєте каптчу. В User Agent міститься інформація про браузер і його версію, систему і платформу користувача, розширення і плагіни.

```
private static final String USER_AGENT =
        "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.15 (KHTML, like Gecko) Chrome/24.0.1295.0 Safari/537.15";
```

Рисунок 2.5 – Приклад User Agent, який використовувався

Також було збільшено час очікування відповіді від Bing до 20 секунд. Даний параметр допомагає при поганому Інтернет-з’єднанні.

Після відправлення запиту ми отримуємо відповідь у HTML форматі. Інформація, яка нас цікавить, знаходиться у у списку <ol id=”b_results”> (Рис. 2.6). Щоб обробити дану відповідь, ми використали бібліотеку jsoup.

Jsoup – це бібліотека Java для роботи з HTML. Вона забезпечує дуже зручний API для отримання URL-адрес, вилучення та обробки даних, використовуючи найкращі з методів HTML5 DOM і CSS селекторів. [10]

Також розглядалася бібліотека Selenium, але в порівнянні з нею, jsoup значно швидше обробляє HTML, тому вибір був зроблений на користь бібліотеки jsoup.

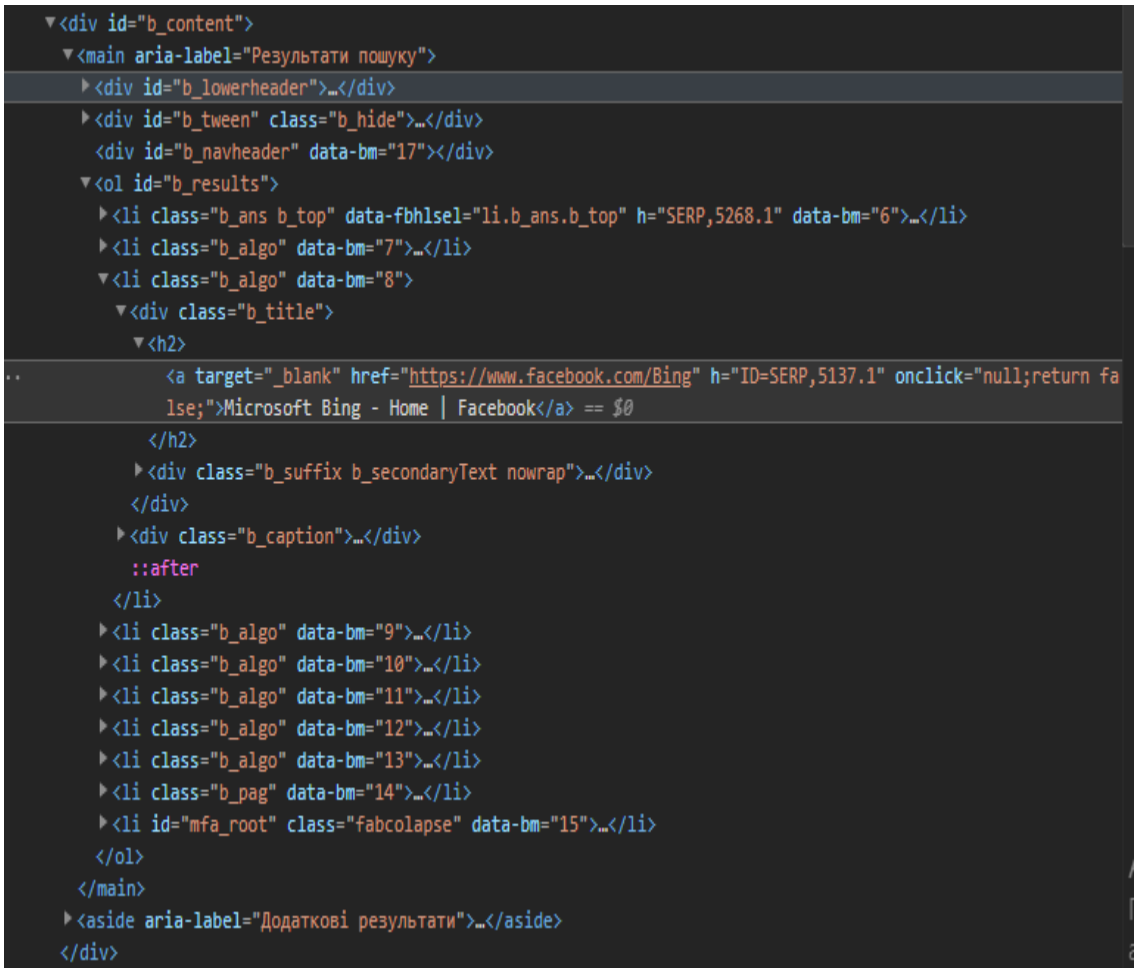

Рисунок 2.6 – Приклад отриманої HTML-відповіді від пошукової системи Bing

```
Connection connection = Jsoup.connect(url).timeout(20000).userAgent(USER_AGENT);
try {
    htmlDocument = connection.get();
} catch (IOException e) {
    e.printStackTrace();
}

if (connection.response().statusCode() == 200) {
    System.out.println("Success. Visiting: " + url);
} else {
    System.out.println("Failure. Code: " + connection.response().statusCode() + " "
            + url);
}

Elements linksOnPage = null;
if (htmlDocument != null) {
    linksOnPage = selectData(htmlDocument);
}
results.addAll(addData(linksOnPage));
connection.execute();
```

Рисунок 2.7 – Код для парсингу пошукової системи Bing

За допомогою вищенаведеного коду, ми отримуємо результати з однієї сторінки пошуку Bing. Для отримання 50 результатів необхідно знарсити 5 сторінок пошукової системи за заданим ключовим словом.

Таким чином ми отримуємо 50 результатів пошуку для одного ключового слова. В пам’яті вони займають близько 2 Мб.

Проте, не все так добре як здається. Після частих запитів з однієї ip-адреси (під одним запитом мається на увазі отримання всіх потрібних результатів для одного ключового слова), Bing перестає надсилати результати і користувач отримує пусту HTML сторінку.

Тому, шляхом тестування, було встановлено, що якщо робити паузу розміром в 400 мс., то можна обійти дане обмеження.

Для збільшення швидкості парсингу ми скористалися потоками в Java і для кожного ключового слова створили потік, який запускається через 400 мс після старту попереднього, що значно пришвидшило процес парсингу.

Отже, нами були отримані такі результати. Для парсингу 1000 ключових слів з однієї IP-адреси і отримання 50000 результатів потрібно в середньому 6,5 хвилин (значення залежить від швидкості Інтернет-з’єднання і потужності ПК). Код парсеру знаходиться в *Додатку А*.

2.2.4 Архітектура

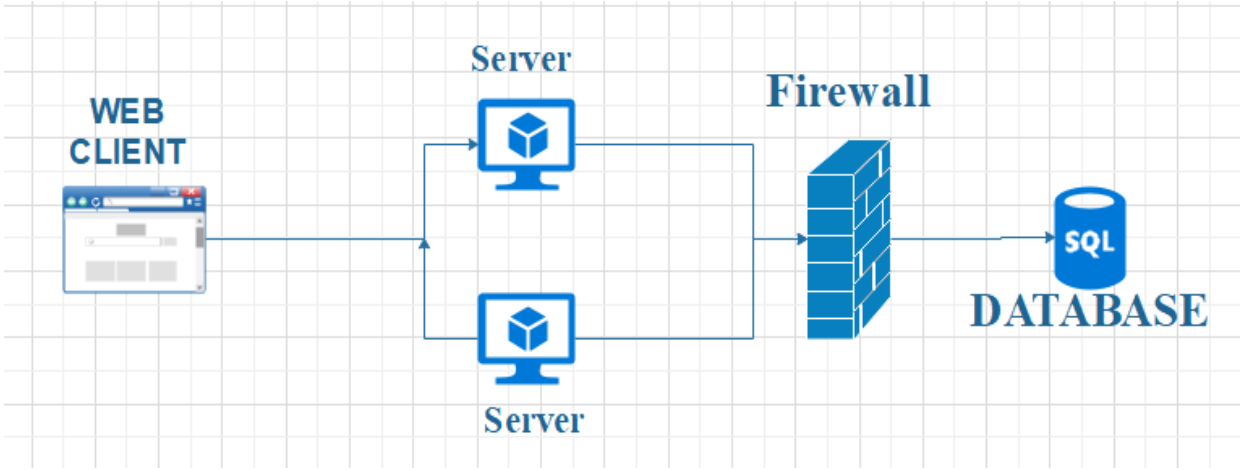

Рисунок 2.7 – Розроблена архітектура для додатку

Для розгортання парсера був вибраний Microsoft Azure. В ньому був створений SQL Server з фаєрволом та два сервери для парсингу, що відповідно мають різні IP -адреси, що в свою чергу дозволяє збільшити швидкість парсингу у два рази. Також на Heroku був створений веб-клієнт розроблений на React, за допомогою якого користувач може передавати ключові слова для парсингу на сервери.

Для створення серверу був використаний Spring Framework. Spring Framework – це програмний фреймворк з відкритим кодом та контейнерами для платформи Java. [13]

2.2.5 Підсумок

В результаті був створений та розгорнутий парсер пошукової системи Bing з наступними можливостями:

- За допомогою веб-клієнта користувач може передати на парсинг одне ключове слово або завантажити csv-файл зі списком ключових слів.
- Швидкий парсинг. За рахунок використання двох серверів, час роботи парсеру пришвидшується в два рази. При розгортанні додаткових серверів час роботи буде пропорційно зменшуватись.
- Запис результатів в базу даних. Всі отримані результати записуються в базу даних, що дозволяє користувачу отримати результати за датою, за ключовим словом або за Url. Наприклад, це дозволяє переглядати результати для одного ключового слова за різний період часу, що може бути використано при аналізі пошукових результатів.
- У веб-клієнті користувач може переглядати результати парсингу для одного ключового слова або індекс Url-адреси для різних ключових слів в яких вона присутня.

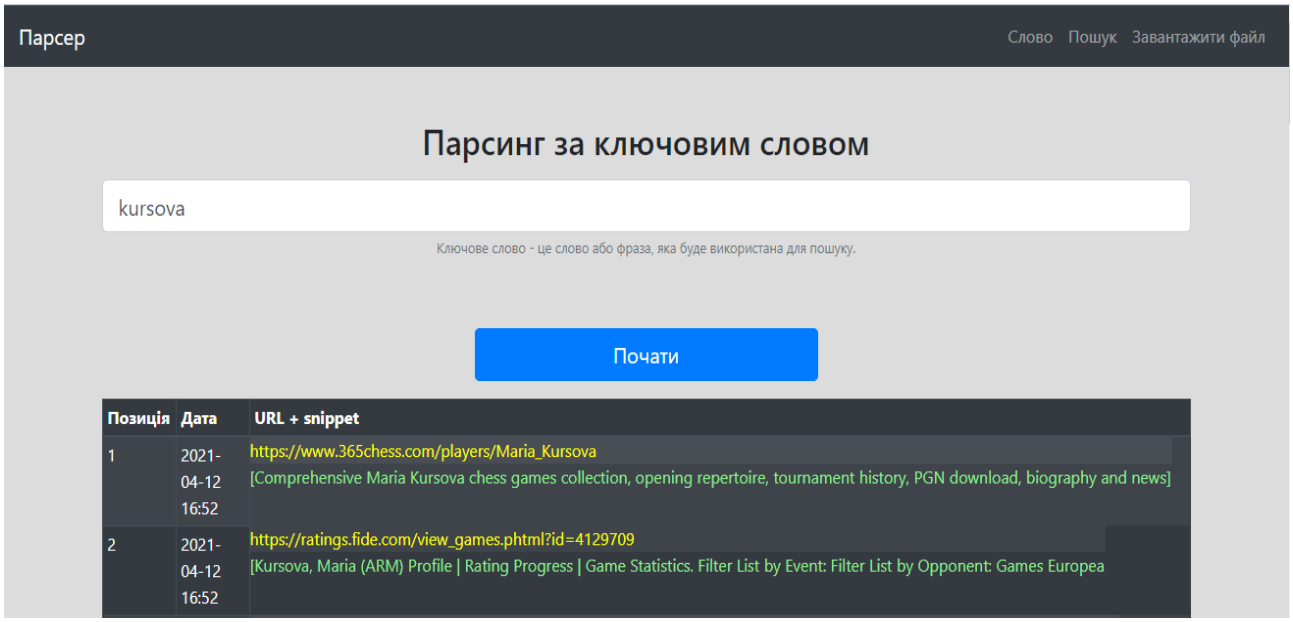

Рисунок 2.8 – Результат роботи веб-застосунку для парсингу одного ключового слова

Позиція	Дата	URL + snippet
1	2021-04-12 16:56	https://binghomepagequiz.org/ [Bing, as a web page of full knowledge about different topics, offers its users, quizzes to test their brains on what they know
2	2021-04-12 16:56	https://blogs.bing.com/search/2012/09/06/take-the-bing-it-on-challenge/ [06/09/2012 · The Bing It On Challenge is an online tool that makes it easy for you to compare Bing and Google's web search res
3	2021-04-12 16:56	https://blogs.bing.com/search/2013/02/06/bing-your-brain-test-then-test-again/ [06/02/2013 · Bing Your Brain: Test, Then Test Again One of the unique interplays between psychology and language is the way in
4	2021-04-12 16:56	https://4.bing.com/images/search?q=Google+vs+Bing+Test&FORM=RESTAB&qft=%2bfilterui%3alicenseType-Any [The photos you provided may be used to improve Bing image processing services.]
5	2021-04-12 16:56	https://www.theverge.com/2016/1/12/10754322/bing-speed-test-tool [12/01/2016 · Bing's speed test widget. Next Up In Tech. Sign up for the newsletter Verge Deals Subscribe to get the best Verg
6	2021-04-12 16:56	https://www.microsoft.com/en-us/download/details.aspx?id=41702 [02/04/2021 · This installer makes Bing your default search engine. The installation applies to Internet Explorer, Firefox, Chr
7	2021-04-12 16:56	https://www.zdnet.com/article/is-google-better-than-bing-i-asked-google-and-bing-and-got-surprising-results/ [Microsoft is about to force Bing onto Office 365 Plus users. But does even Bing think it's better than Google? I asked each s
8	2021-04-12 16:56	https://testdivertidos.es/ [TestDivertidos.es Los tests gratis mas divertidos de internet: inteligencia, personalidad, psicologicos, de amor, para ninos..

Рисунок 2.9 – Результат пошуку в базі даних результатів одного ключового слова

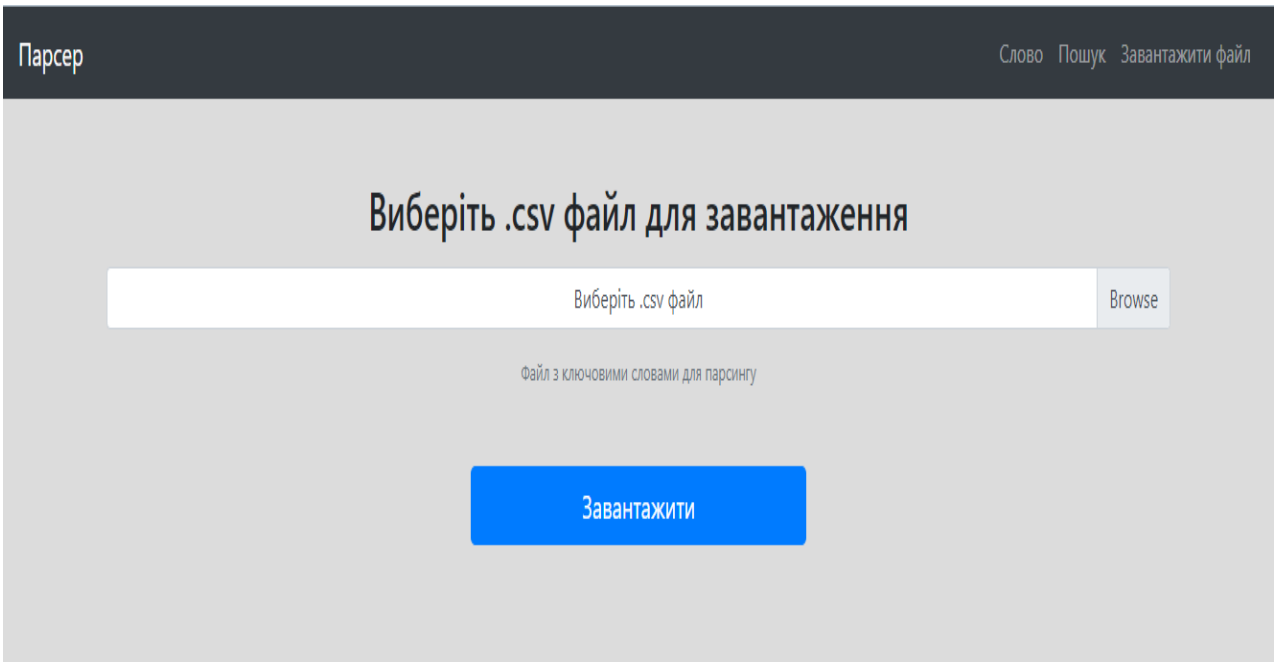

Рисунок 2.10 – Функція завантаження CSV-файлу з ключовими словами для швидкого парсингу

Отже, був розроблений повноцінний парсер пошукової системи Bing з вищеперерахованими можливостями. Увесь код можна переглянути за наступним посиланням https://github.com/kostenk0/Kursova_robota_4.

Розділ 3: Аналіз навантажувальних проблем

Протягом розробки виникали різні навантажувальні проблеми. Деякі з них вдалось вирішити, деякі обійти, а деякі так і залишилися.

3.1 Проблема блокування парсеру по IP-адресі

При частих запитах з однієї IP-адреси, парсер блокувався по ній і на дану адресу не відправлялись дані пошуку. Дана проблема уже згадувалась в роботі і був наведений спосіб як її вирішити.

Для вирішення потрібно робити паузу між запитами до сервера Bing з мінімальним розміром в 400 мс. Дане рішення сповільнює роботу програми, адже сам процес парсингу відбувається швидше, ніж 400 мс.

З цієї причини було створено два сервери, які мають різну IP-адресу, що дозволяє парсити швидше.

3.2 Проблема виникнення Captcha

Під парсингу періодично виникає Captcha, яка просить підтвердити, що користувач є людиною. Звичайно існують різні способи обходу цього тесту, але ніякий з них не гарантує успіху.

Тому був використаний спосіб повторного звернення. При виникненні Captcha, адреса запиту запам’ятовується, Connection закривається і запит відправляється ще раз. При повторному виникненні Captcha, даний спосіб повторюється 3 рази і якщо дані не вдалось отримати, то парсер продовжує свою роботу далі. При втраті даних, втрачаються тільки 10 результатів пошукової системи.

Проте, якщо правильно налаштувати User Agent і не звертатись за однією адресою багато разів, то Captcha буде з’являтись в середньому 1 раз на 100 запитів.

3.3 Швидкий запис в пам’ять

Найбільша проблема, яка виникла під час розробки – це проблема швидкого запису результатів в пам’ять бази даних. При парсингу 1000 ключових слів отримуємо 50 000 результатів, які потрібно обробити і записати в базу даних. Даний процес сповільнює роботу парсеру більш ніж у 2-3 рази. Є ряд причин, які сповільнюють процес запису в базу даних:

- Великий об’єм даних.
- Одночасність запису в базу даних.

Для пришвидшення роботи було використано UUID та executeUpdate(). UUID – клас, який представляє універсально унікальний ідентифікатор. [14] Він дозволяє створювати унікальний ID для даних, що дає можливість одночасного запису в базу даних.

excuteUpdate() – метод, який дозволяє додати всі SQL-запити в чергу і за одни раз відправити їх на SQL-сервер. За його допомогою, кількість запитів до бази даних зменшується, що пришвидшує запис.

Отже, дана проблема найбільше сповільнює роботу парсеру при роботі з великими об’ємами даних.

Висновки

В даній курсовій роботі було розглянуто різні способи парсингу пошукової системи Bing.

Перший спосіб – Bing Web Search API, який за короткий проміжок часу дозволяє отримати готовий парсер для різних мов програмування. Він надає ряд унікальних функцій, які можуть бути корисними для різних користувачів. Проте даний сервіс є платним і має ряд обмежень, котрі будуть значними недоліками при парсингу великої кількості даних.

Другий спосіб – власноруч розроблений парсер. З його допомогою можна парсити великі об’єми даних за короткий термін. Основною затримкою в роботі парсеру є запис в базу даних, оскільки необхідно записати великий об’єм інформації. Також швидкість пошуку по базі даних з часом буде зменшуватись через зростання кількості екземплярів у реляційній базі даних. Проте, розроблений парсер дозволяє користувачу швидко і легко зпарсити ключове слово та отримати необхідні дані. Також запис в базу даних дозволяє користувачу отримати всі попередні результати парсингу для ключового слова і порівняти отримані дані. Розроблений веб-клієнт дозволяє користувачу використовувати парсер на будь-якому пристрої та не переживати, що його IP-адреса буде заблокована пошуковою системою Bing.

В пошуковій системі Bing існують механізми захисту від частих автоматизованих запитів. Основним способом захисту є блокування IP-адреси з якої ідуть запити. Тому для безпечного парсингу, зручно використовувати декілька серверів з різними IP-адресами, що дозволить збільшити продуктивність парсеру.

З економічної точки зору, розробникам необхідно визначитись з очікуваними потужностями, оскільки оренда чи покупка серверів є фінансово затратними, то в залежності від очікуваних результатів, треба зробити вибір більш економічно вигідного методу парсингу.

Отже, процес парсингу пошукової системи Bing є актуальним завданням на сьогоднішній день. Його реалізація можлива за допомогою багатьох методів. Під час процесу парсингу потрібно враховувати очікувану швидкість, кількість отриманих результатів, необхідність збереження даних та вартість використовуваних сервісів.

Список використаних джерел

1. Morgan A. Introduction to Web Crawling & Scraping [Електронний ресурс] / Allison Morgan. – 2018. – Режим доступу до ресурсу: https://medium.com/@allisonmorgan/short-essay-on-web-crawling-scraping-8abf1b232b65.
2. Robots.txt [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Robots.txt.
3. Мельник К. В. АВТОМАТИЧНИЙ ЗБІР ІНФОРМАЦІЇ (ПАРСИНГ) В МЕРЕЖІ / К. В. Мельник, В. М. Мельник, А. М. Григоришин. // Науковий журнал "Комп'ютерно-інтегровані технології: освіта, наука, виробництво". – 2020. – №39.
4. Краулінг [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Краулінг.
5. Пошуковий робот [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Пошуковий_робот.
6. Web scraping [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Web_scraping.
7. What is the Bing Web Search API? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.microsoft.com/en-gb/azure/cognitive-services/bing-web-search/overview.
8. Cognitive Services Pricing - Bing Search API [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/pricing/details/cognitive-services/search-api/.
9. Quickstart: Use Java to search the web with the Bing Web Search REST API, an Azure cognitive service [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-gb/azure/cognitive-services/bing-web-search/quickstarts/java.
10. jsoup: Java HTML Parser [Електронний ресурс] – Режим доступу до ресурсу: https://jsoup.org.

11. Sitemaps [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Sitemaps.
12. CAPTCHA [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/CAPTCHA.
13. Spring Framework [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Spring_Framework.
14. Universally unique identifier [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Universally_unique_identifier.

Додаток А

```
public class Parser{

    private static final String BING_SEARCH_URL =
"http://www.bing.com/search?q=keyword";
    private static final String USER_AGENT =
            "Mozilla/5.0 (Windows NT 6.2; WOW64)
AppleWebKit/537.15 (KHTML, like Gecko) Chrome/24.0.1295.0
Safari/537.15";
    private static String keyword;

    public Parser(String keyword){
        this.keyword = keyword;
    }

    private static String createUrl(String searchUrl, String
searchStr, int page) {
        return searchUrl.replaceAll("keyword",
URLEncoder.encode(searchStr, StandardCharsets.UTF_8))
                + "&first=" +
URLEncoder.encode(String.valueOf(page), StandardCharsets.UTF_8);
    }

    private void parser(String searchStr, int count) throws
Exception {
        ArrayList<SearchResult> results = new ArrayList<>();
        String url;
        Document htmlDocument = null;

        for (int i = 10; i <= count; i += 10) {
            url = createUrl(BING_SEARCH_URL, searchStr, i + 1);
            Connection connection =
Jsoup.connect(url).timeout(20000).userAgent(USER_AGENT);
            try {
                htmlDocument = connection.get();
            } catch (IOException e) {
                e.printStackTrace();
            }

            if (connection.response().statusCode() == 200) {
                System.out.println("Success. Visiting: " + url);
            } else {
                System.out.println("Failure. Code: " +
connection.response().statusCode() + " "
                        + url);
                throw new Exception();
            }

            Elements linksOnPage = null;
            if (htmlDocument != null) {
```

```
                linksOnPage = selectData(htmlDocument);
            }
            results.addAll(addData(linksOnPage));
            connection.execute();
        }
        if(results.size()>0){
            AllResults allResults = new AllResults(searchStr,
results);
            allResults.uploadToBD();}
    }

    private ArrayList<SearchResult> addData(Elements linksOnPage)
{
        String url;
        String snippet;
        ArrayList<SearchResult> results = new ArrayList<>();
        for (Element link : linksOnPage) {
            url =
link.select("h2").select("a[href]").attr("href");
            snippet = String.valueOf(link.select("p").eachText());
            SearchResult searchResult = new SearchResult(url,
snippet);
            results.add(searchResult);
        }
        return results;
    }

    private static Elements selectData(Document htmlDocument) {
        return
htmlDocument.select("ol[id=\"b_results\"]").select("li[class=\"b_a
lgo\"]");
    }

    private static void showResults(Elements linksOnPage) {
        for (Element link : linksOnPage) {

System.out.println(link.select("h2").select("a[href]").attr("href"
));
            System.out.println(link.select("p").eachText());
        }
    }

    public void run() {
        try {
            parser(keyword, 50);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```