Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)
https://doi.org/10.36930/40340716
Article received 18.10.2024
Article accepted 21.11.2024
UDC 004.62
Correspondence author: Ye. V. Levus yelevus@gmail.com

***А. А. Дзендзя, Є. В. Левус, А. С. Вовк***
*Національний університет "Львівська політехніка", м. Львів, Україна*

# АНАЛІЗ ЕФЕКТИВНОГО ЗАСТОСУВАННЯ МЕТОДІВ АВТОМАТИЗОВАНОГО ЗБИРАННЯ ДАНИХ З ВЕБСАЙТІВ

Розглянуто актуальну для сфер електронної комерції, соціальних мереж, наукових досліджень завдань автоматизованого збирання даних з вебсайтів. Проаналізовано особливості застосування двох методів – вебскрейпінгу та інтерфейсу прикладного програмування API (англ. *Application Programming Interface*) для розроблення ефективного комбінованого методу вирішення цього науково-практичної завдання як з погляду продуктивності, так і повноти отриманого результату. Розроблено власну систему автоматизованого збирання даних Harvester з використанням Microsoft.NET, AngleSharp, JSONpath, React.js. Оцінено можливість методів на здатність обробляти дані про вінілові платівки, що містять ціну, автора, назву релізу та штрихкод. Проведено низку експериментів для п'яти вебсайтів, із кожного з яких отримано від 500 до 4000 записів. Загальна кількість отриманих записів – 14995. В обчислювальних експериментах використано два типи джерел: 1) масові джерела – джерела, коли обробляється тільки сторінка каталогу з переліком товарів, де розміщені основні параметри продуктів; 2) одиничні джерела – коли після оброблення сторінки каталогу виконують додатковий збір даних зі сторінок окремих товарів для отримання дещо детальнішої інформації про кожен продукт. Вебскрейпінг було виконано за допомогою PuppeteerSharp для імітації користувача та AngleSharp для синтаксичного аналізу (парсингу) даних, тоді як доступ до API був структурований через REST. Запропоновано для аналізу продуктивності методів виокремити три основні етапи процесу оброблення сторінки вебсайту: завантаження, витягування, простій. Аналіз показав, що API забезпечує значно швидший (у середньому в 10 разів) і надійніший доступ до структурованих даних порівняно з вебскрейпінгом. API-запити дають можливість безпосередньо отримувати конкретну інформацію у стиснутому форматі (зазвичай JSON), що знижує загальну тривалість оброблення. На відміну від цього, вебскрейпінг, хоча й забезпечує більшу гнучкість, проте вимагає більше обчислювальних ресурсів через потребу синтаксичного аналізу HTML, оброблення динамічного контенту та навігації вебсторінками. Запропоновано для збирання даних з вебсайтів використовувати комбінований метод, що поєднує вебскрейпінг та запити інтерфейсу прикладного програмування. Цей метод забезпечує більшу повноту зібраних даних, ніж метод інтерфейсу прикладного програмування з оптимальною швидкістю порівняно з вебскрейпінгом.

**Ключові слова:** оброблення даних; вебскрейпінг; інтерфейс прикладного програмування (API); неструкторовані дані; витягування даних; простій.

## Вступ / Introduction

Вебскрейпінг (англ. *Scraping* "вишкрібання", вебзбирання або витягнення вебданих) – процес збирання великої кількості даних з вебсайтів шляхом синтаксичного аналізу HTML-коду вебсторінки, коли напівструктуровані або неструктуровані дані перетворюють у структуровані. Зазвичай процес виконують за допомогою програмних інструментів, таких як спеціальні браузери або HTTP-запити, що імітують дії користувача на вебсторінках [18].

Отже, вебскрейпінг та інтерфейс прикладного програмування API (англ. *Application Programming Interface*) – два ключові методи збирання даних з мережі Інтернет, які використовують для автоматизації процесу отримання інформації з вебсайтів. Робота вебскрейпінгу полягає в автоматизованому витягуванні даних з вебсторінок через синтаксичний аналіз HTML-коду, тоді як API-метод надає структурований та безпосередній доступ до даних через стандартизовані запити. Обидва підходи є надзвичайно важливими у сучасному світі, де обсяг інформації зростає щодня, а потреба у швидкому доступі до цих даних стає критичною для різних сфер її використання [16, 18].

Актуальність теми зумовлена широким використанням методів вебскрейпінгу та API у бізнесі, аналітиці та наукових дослідженнях. Наприклад, компанії у сфері електронної комерції використовують ці методи для моніторингу цін конкурентів, виявлення трендів і аналізу споживчих вподобань [13]. В аналітичних дослідженнях, таких як аналіз соціальних мереж, вебскрейпінг та

**Інформація про авторів:**
**Дзендзя Андрій Андрійович**, магістрант, кафедра програмного забезпечення.
**Email:** andrii.dzendzia.mpzip.2022@lpnu.ua; https://orcid.org/0009-0002-8379-4020
**Левус Євгенія Василівна**, канд. техн. наук, доцент, кафедра програмного забезпечення.
**Email:** yevheniia.v.levus@lpnu.ua; https://orcid.org/0000-0001-5109-7533
**Вовк Андрій Святославович**, магістрант, кафедра програмного забезпечення.
**Email:** andrii.vovk.mpzip.2023@lpnu.ua; https://orcid.org/0009-0003-2071-5249

**Цитування за ДСТУ:** Дзендзя А. А., Левус Є. В., Вовк А. С. Аналіз ефективного застосування методів автоматизованого збирання даних з вебсайтів. Науковий вісник НЛТУ України. 2024, т. 34, № 7. С. 128–136.
**Citation APA:** Dzendzia, A. A., Levus, Ye. V., & Vovk, A. S. (2024). Analysis of the effective use of the methods of automated data collection from websites. *Scientific Bulletin of UNFU*, 34(7), 128–136. https://doi.org/10.36930/40340716

метод API забезпечують доступ до масивів даних, необхідних для оцінювання поведінки користувачів, прогнозування тенденцій і покращення маркетингових стратегій [17]. Водночас, у наукових дослідженнях [3, 9, 15] вебскрейпінг та метод API широко застосовують для автоматизованого збирання даних, а саме: для побудови моделей та проведення різних аналітичних досліджень.

*Об'єкт дослідження* – автоматизоване збирання даних з вебсайтів.

*Предмет дослідження* – поєднання методів вебскрейпінгу та API для автоматизованого збирання даних з вебсайтів, що забезпечить належний рівень повноти отримання даних за мінімально можливих витрат часу.

*Мета роботи* – розробити вебзастосунок для автоматизованого збирання даних з вебсайтів, який враховуватиме особливості наявних методів вебскрейпінгу та API для забезпечення необхідної повноти отриманих даних і швидкодії їх збирання.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

1. Проаналізувати на підставі наукових публікацій сучасні підходи до автоматизованого збирання даних з вебсайтів, виявити їх недоліки, переваги, що дасть змогу визначити обмеження у їх застосуванні та здійснити постановку науково-практичної задачі, яка не вирішена повною мірою з погляду досягнення необхідних повноти і швидкодії отримання даних.
2. Розробити розподілений застосунок для реалізації методів вебскрейпінгу та API, що дасть змогу провести обчислювальні експерименти з автоматизованого збирання даних з вебсайтів, врахувавши вплив різних типів джерел даних та тривалості виокремлених етапів зазначених методів на якість отриманого результату.
3. Проаналізувати переваги та недоліки кожного з методів на підставі проведених обчислювальних експериментів, що дасть змогу сформувати комплексний підхід до поєднання базових складових методів вебскрейпінгу та API.
4. Проаналізувати результати щодо продуктивності та повноти отримання даних, що забезпечить можливість верифікувати комплексний підхід до поєднання методів вебскрейпінгу та API.

**Аналіз останніх досліджень та публікацій.** Дослідження на тему вебскрейпінгу та використання методу API для збирання даних активно розвиваються, позаяк багато авторів розглядають ці методи з різних перспектив застосування. Зокрема, аналізують їх ефективність з погляду використання ресурсів, таких як обсяги оперативної пам'яті та тривалість збирання даних.

Автори роботи [12] пропонують рішення на підставі Selenium для збирання даних із соціальних мереж, зокрема Twitter, де метод API не забезпечує доступ до необхідних даних. Детально розглянуто розроблення системи, яка використовує вебскрейпінг для обходу обмежень методом API соціальних мереж. Ця робота вказує на важливість вебскрейпінгу у випадках, коли API обмежені у своїх можливостях.

Методи вебскрейпінгу досліджено в роботі [9] для отримання даних з новинних вебсайтів, що не мають офіційних API. Автори пропонують метод, який використовує інтерфейс прикладного програмування HTML DOM, для збирання даних, таких як заголовки та зміст новин, і вказують на те, що цей метод є ефективним для збирання інформації з динамічних сторінок.

У роботі [1] презентовано автоматизовану систему моніторингу вебсайтів за допомогою вебскрейпінгу на базі Raspberry Pi. Автори розробили систему, яка автоматично перевіряє доступність сайтів і збирає інформацію про їхнє завантаження. Це демонструє застосування вебскрейпінгу не тільки для збирання даних, але й для моніторингу продуктивності сайтів.

Використання вебскрейпінгу для збирання даних з Instagram, що є корисним для маркетингових досліджень та аналізу соціальних мереж, описано в роботі [13]. Ця робота зосереджена на збиранні інформації про підписників, лайки та коментарі з публікацій, а також – обході обмежень, накладених методом API.

Автори роботи [2] вивчають продуктивність бібліотек Python, таких як BeautifulSoup та Selenium, у контексті швидкості та точності збирання даних з динамічних вебсайтів. Різні підходи, використані авторами для отримання результатів, містять бібліотеку запитів, Selenium та інші зовнішні бібліотеки. Використовуючи отримані результати, результати візуалізуються за допомогою різних графіків, що мають контрастні параметри як вимірювання. Результати, отримані дослідниками, свідчать про те, що для захисту вебсайтів від роботів вебзбирання потрібно багато вдосконалити.

У роботі [8] проаналізовано вебскрейпінг як науковий інструмент для збирання теоретичної інформації. Автори демонструють, як автоматизований збір даних допомагає дослідникам швидко збирати великий обсяг інформації з наукових баз даних, що робить цей підхід ефективним для наукових досліджень.

Дослідження [20] описує вебскрейпінг IMDb для автоматизованого збирання інформації про фільми, зокрема їхні рейтинги та рецензії. Використання вебскрейпінгу дає змогу дослідникам збирати велику кількість даних про фільми, які можуть бути використані для аналітики та прогнозування.

Для різних завдань у вебскрейпінгу, таких як пошук шаблонів у тексті, екстракція даних з HTML-тегів і перетворення даних у зручний для аналізу формат, використовують регулярні вирази. Вони особливо корисні під час роботи з неструктурованими або слабо структурованими даними, де стандартні методи синтаксичного аналізу можуть бути недостатньо ефективними [10, 20]. У роботі [10], присвяченій порівнянню технік вебскрейпінгу, регулярні вирази показали свою ефективність використання оперативної пам'яті порівняно з іншими методами. Проте, дослідження показало, що метод з використанням інтерфейсу прикладного програмування HTML DOM забезпечує на 25 % швидше виконання запитів, ніж регулярні вирази чи XPath, що робить його незамінним інструментом для сучасних вебзастосунків.

Окрім цього, юридичні та етичні особливості вебскрейпінгу також викликають занепокоєння, оскільки збір даних може порушувати політику конфіденційності або умови використання вебсайтів у роботі [1].

Хоча вже досягнуто значних результатів у ефективному застосуванні методів для автоматизованого збирання даних з вебсайтів, ще залишаються невирішені питання їх розвитку в різних контекстах, зокрема, досягнення якнайбільшої повноти отриманих даних з мінімально можливими витратами часу.

**Матеріали та методи дослідження.** У дослідженні використано ключові методи збирання даних з вебсайтів – вебскрейпінг і API. Зважаючи, що ці методи мають істотні відмінності з погляду архітектури, обсягу даних і швидкості доступу до інформації, які впливають на ефективність їх застосування, зазначені методи

поєднано в одному застосунку автоматизованого збирання даних з вебсайтів. Для імітації користувача використано інструмент PuppeteerSharp, а для витягування даних – AngleSharp.

## Результати дослідження та їх обговорення / Research results and their discussion

У разі джерела даних поділено на два типи: масові та одиничні. Такий поділ зумовлений різницею в обсязі даних які потрібно опрацювати, а також підходом до оброблення сторінок вебсайтів.

*Масові джерела* – це джерела такого типу, коли обробляється тільки сторінка каталогу. Це може бути сторінка з переліком товарів або наданих послуг, де розміщені основні параметри продуктів (назва, ціна, короткий опис тощо). Додатковий збір інформації зі сторінок каталогу окремих товарів не проводиться з різних причин, що дає змогу швидко отримати загальну картину з даних про всі продукти.

У разі одиничних джерел процес є більш складним, оскільки після оброблення сторінки каталогу виконують додатковий збір даних зі сторінок окремих товарів. Це дає змогу отримати детальнішу інформацію про кожен продукт, зокрема технічні характеристики, детальні описи та відгуки.

Для аналізу продуктивності методів автори наведеного дослідження пропонують виокремити три основні етапи процесу оброблення сторінки вебсайту (табл. 1):

1. *Завантаження.* На цьому етапі потрібно завантажити потрібний ресурс – вебсторінку чи API-шлях, врахувавши особливості роботи та методи безпеки цих ресурсів.
2. *Витягування*. Збір інформації з вебсторінки відбувається після успішного завантаження сторінки або відповіді з API-сервера.
3. *Простій.* Після витягування даних встановлюється короткий інтервал очікування, перш ніж відправляти новий запит, що допомагає уникнути перевантаження сервера або виявлення витягування засобами безпеки вебсайту.

**Табл. 1**. Порівняння етапів збирання даних методами Вебскрейпінгу та API / Comparison of data collection stages by API and webscraping methods

| Етап \ Метод | | Вебскрейпінг | API |
|---|---|---|---|
| Завантаження | Особливості | Вимагають імітації дій користувача для зниження ризику блокування. | |
| | Інструменти | Імітація користувача через PuppeteerSharp. Дає змогу автоматично завантажувати вебсторінки та взаємодіяти з ними так, як це зробив би реальний користувач, включно з навігацією, прокруткою та завантаженням динамічного контенту, розробленого засобами технології JavaScript. Підходить і для симулювання користувача під час запитів до методу API, що дає змогу знизити ризик виявлення системою захисту від ботів і подальшого блокування доступу до методу API. | |
| Витягування | Особливості | Завантажені сторінки у форматі html потрібно опрацювати, витягнути з неї корисні дані та привести їх до єдиної структури, для подальшого використання в інших системах. | Завантажену, структуровану відповідь потрібно перебудувати в інший формат (привести до єдиної структури). Хоч дані з API й структуровані – проте кожен вебсайт-джерело має власну структуру, що не є оптимальним для використання. |
| | Інструменти | AngleSharp – бібліотека для синтаксичного аналізу HTML, дає змогу аналізувати структуру вебсторінки та вибирати необхідні елементи, такі як заголовки, ціни, зображення або інша текстова інформація. | JSONpath – інструмент, що дає змогу знаходити та діставати потрібні елементи з JSON-відповіді. |
| Простій | | Після витягування даних встановлюється короткий інтервал очікування, перш ніж відправляти новий запит. Це допомагає уникнути перевантаження сервера або виявлення скрейпера системою безпеки вебсайтів. | |

З погляду архітектури реалізації, вебскрейпінг імітує взаємодію користувача з вебсторінками. Використовуючи такі інструменти, як PuppeteerSharp, вебскрейпінг завантажує вебсторінки та витягує дані безпосередньо з HTML-коду або через DOM-структуру. Цей підхід залежить від структури вебсторінок і може бути чутливим до їхніх змін. Натомість API, такі як REST або GraphQL, надають прямий доступ до структурованих даних через визначені точки доступу. Запити до методу API є більш прямолінійними, оскільки відповіді на них вже оптимізовані для отримання інформації без потреби в синтаксичному аналізі HTML-коду.

Щодо обсягу зібраних даних, то вебскрейпінг дає змогу отримати всі доступні на сторінці дані, навіть якщо вони неструктуровані або недоступні через метод API. Це робить вебскрейпінг гнучким інструментом, особливо для роботи з великими або неструктурованими даними, такими як зображення, коментарі, рейтинги тощо. Проте, його ефективність може знижуватися через технічні перешкоди, такі як CAPTCHA або зміни в DOM-структурі. Методи API, водночас, надають структуровані дані, але обсяг інформації зазвичай обмежений до певних параметрів. Наприклад, багато методів API надають тільки частину необхідної інформації або встановлюють обмеження на кількість запитів за певний період.

З погляду швидкості доступу до інформації, вебскрейпінг вимагає завантаження цілої вебсторінки та витягування даних з її структури, що може бути повільнішим порівняно з методом API. Завантаження великих обсягів даних через вебскрейпінг може займати більше часу, особливо якщо вебсторінка використовує динамічні технології, такі як JavaScript. Доступ до даних через метод API зазвичай швидший, оскільки відповіді методу API оптимізовані для швидкого надання необхідної інформації у форматі JSON або XML. Проте, швидкість доступу через метод API може бути обмежена кількістю запитів, які можна надіслати до сервера за певний проміжок часу.

Архітектура розробленого авторами застосунку автоматизованого збирання даних з вебсайтів Harvester (рис. 1) складається з кількох ключових компонент, кожна з яких виконує свою специфічну функцію для забезпечення безперебійного збирання даних через вебскрейпінг та API-запити. Основні компоненти системи:

1. Користувач – взаємодіє зі системою через браузерне розширення для додавання нових джерел та правил витягування даних.
2. Основний сервер – отримує запити від користувача через браузерне розширення, обробляє їх і зберігає налаштування в базі даних.
3. База даних PostgreSQL – зберігає всі джерела, правила та налаштування, які вводить користувач.

4. Виконавець фонових завдань – відповідає за виконання тривалих завдань, таких як вебскрейпінг та API-запити, зберігає результати у Redis та надсилає їх на сервер користувача.
5. Redis – це сховище зберігає дані за визначений період часу та надає швидкий доступ до них. Зазвичай використовують для гешування часто запитуваних даних, щоб зменшити навантаження на БД (наприклад, популярні товари в мережі Інтернет-магазині). У цій системі використовують для гешування проміжних результатів і зберігання даних, які були зібрані виконавцем фонових завдань.
6. Сервер користувача – отримує дані від виконавця фонових завдань через вебхук і використовує їх для подальшого оброблення чи відображення.

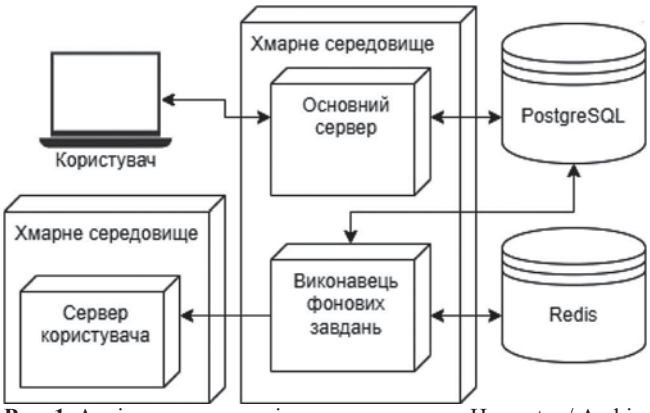


**Рис. 1.** Архітектура розподіленого застосунку Harvester / Architecture of Harvester distributed application

Користувач взаємодіє з основним сервером через спеціальне розширення для браузера. Розширення дає змогу користувачеві додавати нові джерела для збирання даних і визначати правила для їх витягування. Усі введені користувачем правила та налаштування зберігаються в базі даних PostgreSQL. Це забезпечує централізоване зберігання конфігурацій та правил для вебскрейпінгу й API-запитів, що дає змогу зручно керувати і модифікувати їх у майбутньому.

Виконавець фонових завдань відповідає за оброблення тривалих фонових процесів, таких як безпосередній вебскрейпінг та ланцюжки запитів до API. Виконавець отримує завдання з основного сервера, виконує їх у фоновому режимі, а отримані результати зберігає у системі гешування Redis. Це дає змогу тимчасово зберігати отримані дані і швидко їх обробляти.

Окрім цього, виконавець фонових завдань відповідає за надсилання результатів на сервер користувача через механізм вебхуків (англ. *Webhook*) – посилань на відкриті API-шляхи, які очікують на отримання даних. Ця взаємодія є односторонньою: результати надсилаються від виконавця на сервер користувача, що дає змогу уникати складних двосторонніх з'єднань та мінімізувати затримки при передачі даних. Це також дає можливість асинхронного оброблення завдань і не блокує роботу основного сервера під час очікування на завершення завдань.

Отже, архітектура системи ефективно розподіляє завдання між основним сервером, виконавцем фонових завдань та базами даних, забезпечуючи надійне та масштабоване рішення для автоматизованого збирання даних через вебскрейпінг та API.

Реалізація цієї архітектури виконана у застосунку Harvester, який був розроблений для збирання даних через вебскрейпінг і роботу з методом API. Для розроблення системи використано Microsoft.NET, інструменти AngleSharp, JSONpath (табл. 1) – для бекенд-частини, що відповідає за збір даних; а також React.js для розширення Chrome, що відповідає за визначення користувачем структури даних, вебсайтів-джерел та API-шляхів для збирання даних. Для аналізу ефективності методів автоматизованого збирання даних проведено вісім експериментів. Із кожного з п'яти джерела отримано від 500 до 4000 записів. Загальна кількість отриманих записів – 14995. Експерименти проводились на ізольованому сервері для забезпечення належної його продуктивності та зменшення можливих затримок у роботі компонент системи.

Технічні характеристики використаного сервера:

- процесор: Intel i7-8750H (1 CPU, 12 логічних і 6 фізичних ядер);
- графічний процесор: Nvidia GeForce GTX 1070;
- оперативна пам'ять: 32GB DDR4;
- накопичувач: SSD NVME2 2TB;
- операційна система: Windows 11.

Експериментальне середовище розроблено за допомогою Docker Desktop версії 4.34.2 із використанням Linux-контейнерів. Це дало змогу побудувати ізольоване середовище для кожного експерименту, що забезпечило незалежність результатів і зменшення впливу зовнішніх факторів на роботу системи.

Для дослідження ефективності методів вебскрейпінгу та API витягувались дані про вінілові платівки з однаковою структурою: ціна, автор, назва релізу та штрихкод. Часові показники різних етапів збирання даних, таких як завантаження, витягування та простій, проаналізовано для джерел [9, 13, 18, 19, 21, 23, 25].

Витягування даних з масових джерел відбувалося значно швидше, оскільки оброблялися тільки конкретні частини HTML-коду, а не окремі сторінки продуктів. Тривалість завантаження та простій не фіксувалися для масових джерел, оскільки ці етапи стосуються оброблення окремих продуктів (табл. 2). При цьому API-запити демонстрували значно меншу тривалість отримання результатів (зокрема, у 5-10 разів для ідентичних даних) порівняно з вебскрейпінгом, оскільки API надає вже структуровані дані, що спрощує та пришвидшує процес витягування.

У результатах проведеного експерименту (табл. 3) наведено тривалість виконання всіх етапів для сторінок каталогів, що дає змогу порівняти масові й одиничні джерела за однаковими операціями. Оскільки у цьому випадку оброблялися тільки сторінки каталогів, різниці між масовими й одиничними джерелами майже не простежуються. Для обох типів джерел виконувались операції їх завантаження, видобування інформації та простій. Це дає змогу дійти висновку, що основна різниця між цими джерелами проявляється тільки тоді, коли обробляються сторінки окремих продуктів, які не внесені в цей аналіз.

Найповніша картина аналізу вебскрейпінгу та API є у разі об'єднання даних, що містить оброблення як сторінок каталогів, так і окремих продуктів (табл. 4). Оброблення сторінки каталогу враховано в усіх продуктах, шляхом поділу тривалості оброблення каталогу на кількість продуктів, які в ньому знаходяться, що забезпечує точні результати для масових і одиничних джерел. Отже, отримано тривалість збирання даних з різних типів джерел з урахуванням усіх необхідних дій.

**Табл. 2**. Тривалість збирання даних зі сторінок продуктів у Harvester / Duration of data collection from product pages in Harvester

| API/WebScraping | | WebScraping | | | | | API | | |
|---|---|---|---|---|---|---|---|---|---|
| Вебсайт | | Джерело 1 | Джерело 2 | Джерело 3 | Джерело 4 | Джерело 4 | Джерело 4 | Джерело 4 | Джерело 5 |
| Масове/одиничне | | масове | одиничне | масове | одиничне | масове | масове | одиничне | масове |
| Завантаження | Середній час, с | – | 10.542 | – | 19.684 | – | – | 1.97 | – |
| | Медіана, с | – | 10.536 | – | 7.535 | – | – | 1.739 | – |
| | Максимальний час, с | – | 27.011 | – | 388.008 | – | – | 9.721 | – |
| | Мінімальний час, с | – | 3.994 | – | 2.046 | – | – | 1.292 | – |
| Витягування | Середній час, с | $2.66\cdot10^{-4}$ | 0.231 | 0.001 | 0.57 | $6.12\cdot10^{-4}$ | $0.67\cdot10^{-4}$ | $2.43\cdot10^{-4}$ | $1.22\cdot10^{-4}$ |
| | Медіана, мс | $1.52\cdot10^{-4}$ | 0.194 | $4.85\cdot10^{-4}$ | 0.534 | $1.86\cdot10^{-4}$ | $0.38\cdot10^{-4}$ | $1.63\cdot10^{-4}$ | 0.5·10–4 |
| | Максимальний час, с | 0.039 | 2.292 | 0.468 | 1.833 | 0.069 | 0.007 | 0.007 | 0.094 |
| | Мінімальний час, с | $0.84\cdot10^{-4}$ | 0.147 | $2.18\cdot10^{-4}$ | 0.428 | $1.45\cdot10^{-4}$ | $0.17\cdot10^{-4}$ | $1.09\cdot10^{-4}$ | $0.18\cdot10^{-4}$ |
| Простій | Середній час, с | – | 7.988 | – | 7.266 | – | – | 7.418 | – |
| | Медіана, с | – | 8.223 | – | 7.33 | – | – | 7.183 | – |
| | Максимальний час, с | – | 12.825 | – | 12.676 | – | – | 12.984 | – |
| | Мінімальний час, с | – | 2.206 | – | 2.061 | – | – | 2.058 | – |
| Загальний час | Середній час, с | $2.66\cdot10^{-4}$ | 18.762 | 0.001 | 27.52 | $6.12\cdot10^{-4}$ | $0.67\cdot10^{-4}$ | 9.389 | $1.22\cdot10^{-4}$ |
| | Медіана, с | $1.52\cdot10^{-4}$ | 18.88 | $4.85\cdot10^{-4}$ | 15.061 | $1.86\cdot10^{-4}$ | $0.38\cdot10^{-4}$ | 8.977 | $0.5\cdot10^{-4}$ |
| | Максимальний час, с | 0.039 | 39.388 | 0.468 | 397.347 | 0.069 | 0.007 | 22.1 | 0.094 |
| | Мінімальний час, с | $0.84\cdot10^{-4}$ | 9.035 | $2.18\cdot10^{-4}$ | 5.835 | $1.45\cdot10^{-4}$ | $0.17\cdot10^{-4}$ | 3.808 | $0.18\cdot10^{-4}$ |

**Табл. 3**. Тривалість збирання даних через оброблення каталогів у Harvester / Duration of data collection via catalog processing in Harvester

| API/WebScraping | | WebScraping | | | | | API | | |
|---|---|---|---|---|---|---|---|---|---|
| Вебсайт | | Джерело 1 | Джерело 2 | Джерело 3 | Джерело 4 | Джерело 4 | Джерело 4 | Джерело 4 | Джерело 5 |
| Масове/одиничне | | масове | одиничне | масове | одиничне | масове | масове | одиничне | масове |
| Завантаження | Середній час, с | 8.37 | 5.159 | 7.406 | 45.597 | 48.228 | 5.839 | 6.73 | 8.849 |
| | Медіана, с | 8.119 | 2.543 | 3.93 | 35.954 | 23.443 | 5.613 | 7.861 | 9.854 |
| | Максимальний час, с | 23.676 | 19.41 | 131.999 | 145.437 | 133.879 | 10.987 | 8.979 | 10.639 |
| | Мінімальний час, с | 2.138 | 1.394 | 2.094 | 11.437 | 12.147 | 1.757 | 1.353 | 2.83 |
| Витягування | Середній час, с | 0.719 | 0.893 | 1.192 | 1.471 | 1.342 | 0.001 | 0.039 | 0.572 |
| | Медіана, с | 0.323 | 0.837 | 0.797 | 0.641 | 0.713 | $5.38\cdot10^{-4}$ | 0.003 | 0.001 |
| | Максимальний час, с | 4.494 | 1.142 | 10.503 | 2.441 | 2.237 | 0.053 | 0.185 | 2.893 |
| | Мінімальний час, с | 0.238 | 0.669 | 0.329 | 0.271 | 0.442 | $3.31\cdot10^{-4}$ | 0.001 | $1.15\cdot10^{-4}$ |
| Простій | Середній час, с | 8.338 | 6.327 | 7.081 | 8.128 | 9.128 | 7.286 | 8.433 | 8.095 |
| | Медіана, с | 9.15 | 6.119 | 6.609 | 7.645 | 8.904 | 7.173 | 8.854 | 8.674 |
| | Максимальний час, мс | 12.381 | 10.579 | 12.317 | 13.042 | 11.605 | 12.92 | 10.222 | 12.533 |
| | Мінімальний час, с | 3.631 | 2.862 | 2.066 | 3.988 | 7.098 | 2.036 | 6.605 | 2.001 |
| Загальний час | Середній час, с | 17.427 | 12.379 | 15.678 | 74.324 | 62.723 | 13.126 | 15.203 | 17.515 |
| | Медіана, с | 18.081 | 9.565 | 13.091 | 53.245 | 36.145 | 13.328 | 15.345 | 18.744 |
| | Максимальний час, с | 31.8 | 30.658 | 143.119 | 160.921 | 150.060 | 20.876 | 19.203 | 22.943 |
| | Мінімальний час, с | 7.426 | 5.721 | 5.109 | 15.698 | 28.543 | 6.08 | 10.392 | 7.685 |

**Табл. 4**. Зведений час оброблення каталогів і сторінок окремих продуктів у Harvester / Aggregate processing time of catalogs and individual product pages in Harvester

| API/WebScraping | | WebScraping | | | | | API | | |
|---|---|---|---|---|---|---|---|---|---|
| Вебсайт | | Джерело 1 | Джерело 2 | Джерело 3 | Джерело 4 | Джерело 4 | Джерело 4 | Джерело 4 | Джерело 5 |
| Масове/одиничне | | масове | одиничне | масове | одиничне | масове | масове | одиничне | масове |
| Завантаження | Середній час, с | 0.161 | 10.8 | 0.156 | 21.995 | 0.765 | 0.098 | 2.082 | 0.026 |
| | Медіана, с | 0.156 | 11.131 | 0.082 | 9.846 | 0.372 | 0.094 | 1.856 | 0.02 |
| | Максимальний час, с | 0.455 | 27.981 | 2.75 | 390.319 | 2.125 | 0.183 | 9.743 | 0.202 |
| | Мінімальний час, с | 0.041 | 4.083 | 0.044 | 4.357 | 0.192 | 0.029 | 1.416 | 0.015 |
| Витягування | Середній час, с | 0.014 | 0.276 | 0.026 | 0.593 | 0.085 | $0.87\cdot10^{-4}$ | $8.99\cdot10^{-4}$ | 0.001 |
| | Медіана, с | 0.007 | 0.238 | 0.017 | 0.558 | 0.086 | $0.48\cdot10^{-4}$ | $2.29\cdot10^{-4}$ | $0.52\cdot10^{-4}$ |
| | Максимальний час, с | 0.126 | 2.35 | 0.686 | 1.857 | 0.166 | 0.007 | 0.011 | 0.1 |
| | Мінімальний час, с | 0.005 | 0.191 | 0.007 | 0.451 | 0.072 | $0.24\cdot10^{-4}$ | $1.36\cdot10^{-4}$ | $0.18\cdot10^{-4}$ |
| Простій | Середній час, с | 0.16 | 8.305 | 0.149 | 7.44 | 0.144 | 0.122 | 7.559 | 0.027 |
| | Медіана, с | 0.176 | 8.456 | 0.138 | 7.505 | 0.141 | 0.12 | 7.325 | 0.017 |
| | Максимальний час, с | 0.238 | 13.354 | 0.435 | 12.851 | 0.184 | 0.215 | 13.111 | 0.223 |
| | Мінімальний час, с | 0.07 | 2.643 | 0.043 | 2.236 | 0.112 | 0.034 | 2.222 | 0.008 |
| Загальний час | Середній час, с | 0.335 | 19.381 | 0.332 | 30.029 | 0.996 | 0.22 | 9.642 | 0.054 |
| | Медіана, с | 0.348 | 19.82 | 0.273 | 17.569 | 0.608 | 0.222 | 9.263 | 0.038 |
| | Максимальний час, с | 0.651 | 40.921 | 2.984 | 399.856 | 2.382 | 0.357 | 22.273 | 0.309 |
| | Мінімальний час, с | 0.143 | 9.412 | 0.107 | 8.344 | 0.453 | 0.101 | 4.063 | 0.028 |

У процесі завантаження сторінок вебскрейпінг демонстрував більшу варіацію у часі (від десятих до десятків секунд) порівняно з методом API. Це пов'язано з тим, що вебскрейпінг потребує повного завантаження HTML-сторінок, які можуть містити динамічний контент. Натомість метод API дає змогу отримувати тільки

необхідні дані без зайвого навантаження, що зменшує тривалість завантаження. Середня тривалість завантаження для методу API був значно меншим (у 8-10 разів), що робить його вагомо ефективнішим під час роботи з великими обсягами запитів. Проте, за замовчуванням, вебскрейпінг забезпечує більшу повноту даних, ніж метод API.

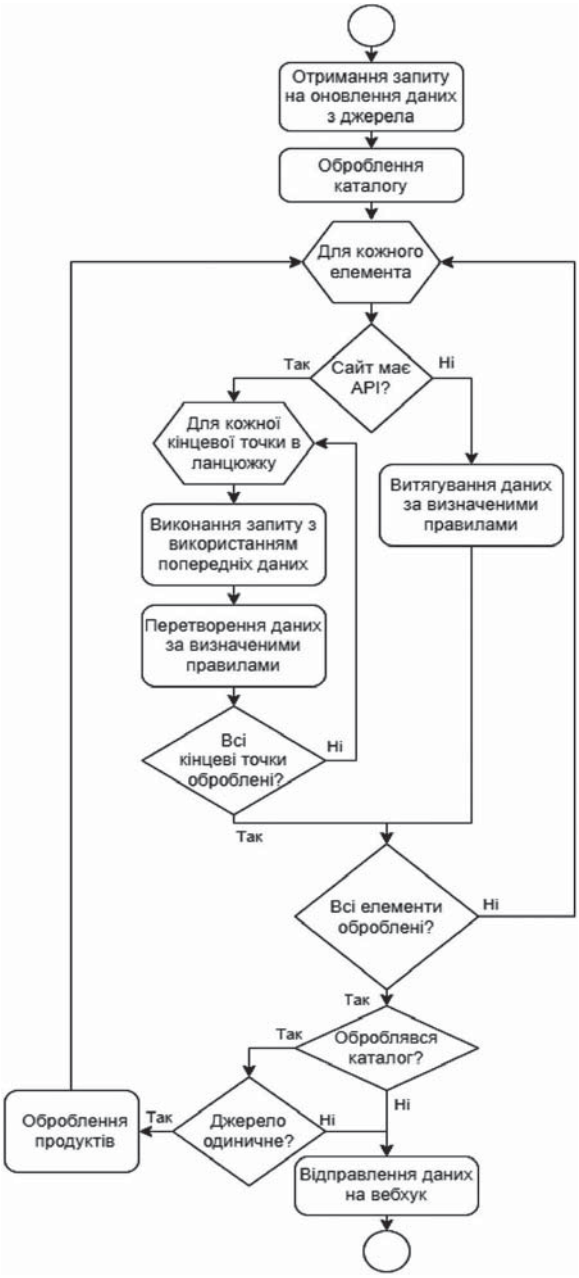


**Рис. 2.** Комбінований метод витягування даних / Combined method of data extraction

На етапі витягування даних через вебскрейпінг потрібно більше часу через потребу синтаксичного аналізу HTML-коду. Складніші сайти з великою кількістю вкладених елементів або динамічними компонентами додатково збільшували цей час. Метод API на цьому етапі продемонстрував свою перевагу, оскільки дані надавалися у структурованому форматі, що спрощувало і пришвидшувало процес витягування.

Простій додано для обох підходів для уникнення блокування серверів через надмірну кількість запитів. У випадку вебскрейпінгу простій був довший через складність запитів і системи захисту на деяких сайтах. Метод API дав змогу зменшити час простою завдяки наявності чітких лімітів на запити.

Загальна тривалість виконання істотно більша для одиничних джерел через потребу оброблення сторінок кожного окремого продукту. У деяких випадках це призводило до значного збільшення тривалості виконання операцій як для вебскрейпінгу, так методу API (у десятки разів).

Результати комплексного аналізу дають можливість запропонувати поєднання розглянутих методів в один комбінований метод для автоматизованого збирання даних, що особливо важливо для швидкого формування повної інформації з різного типу джерел (рис. 2).

Комбінований метод продемонстрував більшу тривалість отримання даних, ніж використання виключно методу API (рис. 3).

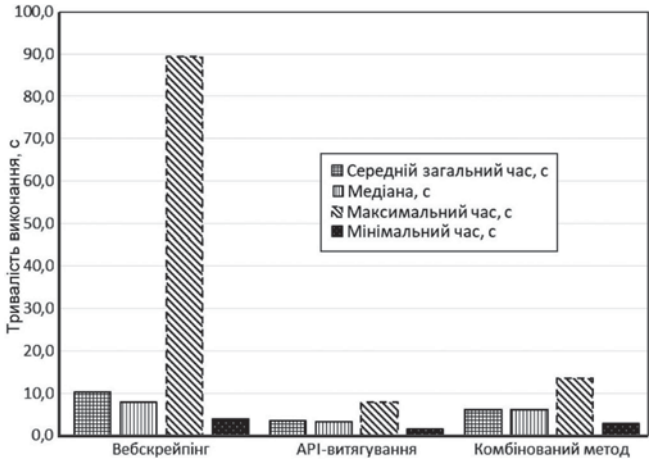


**Рис. 3.** Порівняння тривалості збирання даних методами вебскрейпінгу, API-витягування та комбінованим / Comparison of data collection time using web scraping, API and combined methods

Проте, якщо взяти до уваги, що кінцева мета – зібрати якомога повніші дані, то комбінований метод є найкращим, оскільки заповнює за допомогою вебскрейпінгу дані, до яких немає доступу через метод API (рис. 4).

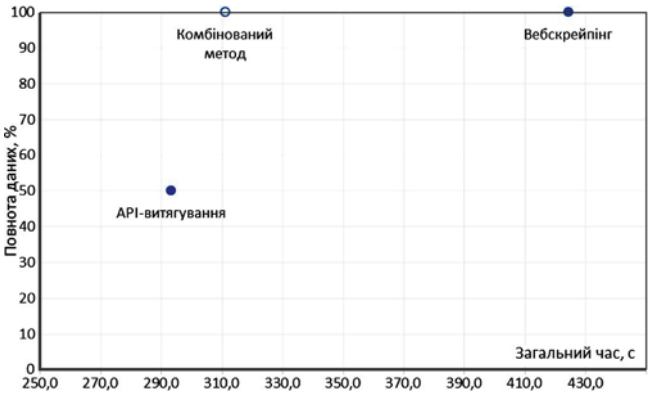


**Рис. 4.** Відношення повноти даних до тривалості виконання / Relationship of data completeness to execution time

Аналіз результатів наведеного дослідження виконано на підставі порівняння обох методів не тільки на рівні завантаження сторінок та витягування даних, але й з урахуванням реальних умов роботи, таких як простій та обробляння динамічного контенту. Це дає змогу краще зрозуміти, за яких умов кожен із методів є найбільш ефективним, а також як тип джерела впливає на загальну продуктивність системи. Як результат, запропоновано комплексний метод, що враховує сильні та слабкі сторони залежно від типу джерела та вимог до даних. Вебскрейпінг продемонстрував гнучкість у тих випадках, коли метод API не надає всіх необхідних даних. Це

особливо корисно для збирання інформації з вебсторінок будь-якої складності, неструктурованих даних (текстові відгуки, зображення) або інформації з динамічних вебсайтів, що не мають стандартних точок доступу до даних. Метод API є значно швидшим і ефективнішим, коли дані вже структуровані і доступні через чітко визначені запити. Використання комбінації цих підходів забезпечує оптимальний компроміс між швидкістю збирання та повнотою даних.

Постійна зміна структури вебсайтів потребує регулярного оновлення скриптів і алгоритмів синтаксичного аналізу, що може значно підвищувати вартість підтримки системи, яка використовує вебскрейпінг. Перспективи розвитку вебскрейпінгу полягають у застосуванні технологій штучного інтелекту та машинного навчання для покращення оброблення динамічних сторінок і адаптації до змін у структурі сайтів.

Перспективи розвитку методу API полягають у розширенні підтримки нових форматів передачі даних, таких як GraphQL, який дає змогу отримувати тільки ті дані, які потрібні, з мінімальними витратами ресурсів. Це допоможе збільшити гнучкість методу API у взаємодії з різноманітними запитами.

**Обговорення результатів дослідження.** Важливою темою в публікаціях, присвячених автоматизованому збиранню даних з вебсайтів мережі Інтернет, є порівняння вебскрейпінгу та API-методу, обговорення їх переваг та недоліків, а також ефективності застосування у різних прикладних контекстах.

У роботі [5, 6] порівнюються методи вебскрейпінгу та API для аналізу достовірності даних на платформі Twitter. Автори показали, що метод API надає структуровані дані, однак має обмеження щодо обсягу доступних історичних даних, тоді як вебскрейпінг дає змогу отримувати великі масиви даних, хоча й залежить від змін у структурі сторінок.

Інше дослідження [11] порівнює вебскрейпінг та API-метод для збирання даних з Twitter. Автори показують, що хоча метод API забезпечує легкий доступ до структурованих даних, вебскрейпінг є більш гнучким інструментом, коли необхідно отримати дані, які метод API не надає. Основні проблеми, з якими стикаються користувачі під час роботи з вебскрейпінгом та методом API, пов'язані з технічними обмеженнями та обмеженнями доступу до даних. Метод застосування методу API зазвичай має обмеження щодо кількості запитів на певний період, що може стати серйозною проблемою для великих проєктів або досліджень, які потребують великого обсягу даних.

У роботі [4] описано використання методу API соціальних медіа, таких як Facebook і Twitter, для доступу до інформації через офіційні канали. Проте, дослідження показує, що навіть з використанням цих методів API багато даних лишаються недоступними через обмеження на кількість запитів, доступ до історичних даних та інші політики захисту. Це обмежує користувачів у можливості отримати повну картину, що змушує їх звертатися до вебскрейпінгу для витягування даних, які метод API не можуть отримати.

Вебскрейпінг, зі свого боку, стикається з низкою проблем, серед яких захист вебсайтів від ботів, таких як CAPTCHA, а також часті зміни у структурі сторінок, що може призводити до збоїв у роботі скриптів, як зазначено в роботі [12]. Дослідження розглядає різні підходи до подолання цих обмежень, зокрема автоматизацію процесу збирання даних через відстеження HTTP-запитів і коректування скриптів під нові структури сторінок. Водночас, захист вебсайтів, як-от CAPTCHA або обмеження на кількість запитів, ускладнює роботу вебскрейперів і вимагає застосування додаткових інструментів, таких як використання проксі-серверів або комбінування вебскрейпінгу з методом API для зменшення ризику блокування доступу до даних.

У роботі [7] наведено результати дослідження особливостей використання неструктурованих даних у бізнесі, застосовуючи для цього структурований підхід. Автори стверджують, що незважаючи на безпрецедентне зростання як обсягу неструктурованих даних UD (англ. *Unstructured Data*), так і пов'язаної з ними методологічної складності їх оброблення, зростає потреба менеджерів у структурованому їхнього уявленні про те, як вибирати джерела даних і методи їх оброблення з огляду на конкретний варіант їх використання або сценарій. Оброблення таких даних зазвичай ресурсомістка, вимагає багатьох етапів і пов'язана з високою невизначеністю, але UD може містити багато інформації, якої немає в структурованих даних. Визнаючи прогалину в чітких інструкціях щодо використання UD в процесі прийняття управлінських рішень, автори розробили систематичний триетапний підхід, згідно з яким відбувається: ідентифікація проблеми, розроблення варіантів рішень і вирішення проблеми. Спираючись на теорію організаційного навчання, дослідники запропонували структуру розроблення обґрунтованих рішень із чотирма концептуально підставі двох вимірів: організаційних цілей навчання (розвідка та експлуатація) та обсягу сканування середовища (внутрішні та зовнішні джерела даних). Окрім цього, автори обговорили наслідки для практиків і окреслили ключові сфери для майбутніх напрямів проведення досліджень.

У дослідженні [22] описується два основні методи обходу CAPTCHA для покращення можливостей вебскрейпінгу: метод сегментації (slicing) та метод обмежувальних рамок (англ. *Bounding Box*). Метод сегментації передбачає поділ зображення CAPTCHA на окремі символи для їхнього подальшого розпізнавання. Цей підхід використовує попереднє оброблення зображення для покращення видимості символів і зменшення шуму на фоні, що дає змогу більш точно витягати символи за допомогою бібліотеки Pytesseract. Другий метод, метод обмежувальних рамок, визначає контури символів і використовує ці рамки для їх подальшого розпізнавання. Обидва методи показали свою ефективність у розпізнаванні CAPTCHA, причому метод сегментації забезпечив вищу точність, тоді як метод обмежувальних рамок був швидшим, але менш точним.

Отже, внаслідок виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

*Наукова новизна отриманих результатів дослідження* – розроблено метод автоматизованого збирання даних, який, на відміну від наявних, поєднує використання методів вебскрейпінгу та API з можливістю виявлення впливу різних типів джерел даних на якість отриманого результату, що забезпечує належний рівень повноти отримання даних за мінімально можливих витрат часу.

*Практична значущість результатів дослідження* – розроблений метод автоматизованого збирання даних

можна застосувати для вдосконалення процесів автоматизованого збирання даних у різних джерел, таких як електронна комерція, бізнес-аналітика, маркетинг, наукові дослідження. Результати дослідження можуть бути корисними для розробників систем вебскрейпінгу й особливостей використання методу API, які працюють над поліпшенням продуктивності своїх рішень та їхньою адаптацією до умов постійної зміни вебсайтів і серверних обмежень.

## Висновки / Conclusions

Розроблено вебзастосунок для автоматизованого збирання даних з вебсайтів, який враховує особливості наявних методів вебскрейпінгу та API для забезпечення необхідної повноти отриманих даних і швидкодії їх збирання. За результатами проведеного дослідження можна зробити такі основні висновки:

1. Визначено особливості застосування ключових методів автоматизованого збирання даних – вебскрейпінгу та API з погляду архітектури реалізації цих методів і отриманих результатів, а саме: обсягів даних і продуктивності роботи.
2. Запропоновано виокремити три етапи автоматизованого збирання даних для врахування різних умов оброблення інформації: завантаження, витягування, простій та два типи джерел: одиничні, масові. Зазначені етапи та типи використано в обчислювальних експериментах.
3. Реалізовано застосунок автоматизованого збирання даних методами вебскрейпінгу та API на підставі розробленої архітектури, де ключовими компонентами є основний сервер, база даних PostgreSQL, виконавець фонових завдань, сховище даних Redis, сервер користувача.
4. Виконано для кожного виокремленого етапу автоматизованого збирання даних обчислювальні експерименти для джерел даних різних типів (одиничні, масові). Зокрема, визначено, що API-запити демонструють значно меншу тривалість отримання результатів порівняно з вебскрейпінгом (5-10 разів). У процесі завантаження сторінок вебскрейпінг має більшу варіацію у часі порівняно з методом API (від десятих до десятків секунд).
5. Поєднано методи вебскрейпінгу та API для покращення загального процесу автоматизованого збирання даних і подолання обмежень їх застосування для досягнення достатньої повноти та швидкодії.
6. Проведено порівняння роботи методів вебскрейпінгу та API, яке засвідчує, що їхнє поєднання демонструє кращі результати для автоматизованого збирання даних з погляду повноти даних і при цьому мінімізує ресурсозатратний вебскрейпінг.

Перспективи розвитку вебскрейпінгу полягають у впровадженні інтелектуальних алгоритмів для оброблення динамічного контенту, тоді як метод API продовжуватимуть удосконалюватися через підтримку нових форматів, таких як GraphQL.

## References


1. Arhandi, P. P., Mashudi, I. A., & Nugroho, F. A. (2021). Automated Website Monitoring System Using Web Scraping and Raspberry Pi. *Telematika: Jurnal Informatika dan Teknologi Informasi*, *18*(2), 222–230. https://doi.org/10.31315/telematika.v18i2.5506
2. Bale, A. S., Ghorpade, N., Rohith, S., Kamalesh, S., Rohith, R., & Rohan, B. S. (2022). Web scraping approaches and their performance on modern websites. 3rd *International Conference on Electronics and Sustainable Communication Systems* (ICESC), 956–959. https://doi.org/10.1109/ICESC54411.2022.9885689
3. Bricongne, J., Meunier, B., & Pouget, S. (2022). Web-scraping housing prices in real-time: The Covid-19 crisis in the UK. *Journal of Housing Economics*, 59. https://doi.org/10.1016/j.jhe.2022.101906
4. Dewi, L. C., Meiliana, N., & Chandra, A. (2019). Social Media Web Scraping using Social Media Developers API and Regex. *Procedia Computer Science*, 157, 444–449. https://doi.org/10.1016/j.procs.2019.08.237
5. Dongo, I., Cadinale, Y., Aguilera, A., Martínez, F., Quintero, Y., & Barrios, S. (2020). Web scraping versus Twitter API: a comparison for a credibility analysis. *Proceedings of the 22nd International conference on information integration and web-based applications & services*, 263–273. https://doi.org/10.1145/3428757.3429104
6. Dongo, I., Cardinale, Y., Aguilera, A., Martinez, F., Quintero, Y., Robayo, G., & Cabeza, D. (2021). A qualitative and quantitative comparison between Web scraping and API methods for Twitter credibility analysis. *International Journal of Web Information Systems*, *17*(6), 580–06. https://doi.org/10.1108/IJWIS-03-2021-0037
7. Evert de Haan, Manjunath Padigar, Siham El Kihal, Raoul Kübler, & Jaap E. Wieringa. (2024, April). Unstructured data research in business: Toward a structured approach. *Journal of Business Research*, Vol. 177, article ID 114655. https://doi.org/10.1016/j.jbusres.2024.114655
8. Farias, W. A., Melo, D. M., dos Santos, L. M., de Oliveira, Â. A., Medeiros, R. L., & Silva, Y. K. (2024). Web Scraping as a scientific tool for theoretical reference, 17 January 2024, PREPRINT (Version 1) available at Research Square. https://doi.org/10.21203/rs.3.rs-3854342/v1
9. Firdian, M. I., Darwiyanto, E., & Adrian, M. (2022). Web Scraping with HTML DOM Method for Website News API creation. *Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika* (*JIPI*), *7*(4), 1211–1219. https://doi.org/10.29100/jipi.v7i4.3235
10. Gunawan, R., Rahmatulloh, A., Darmawan, I., & Firdaus, F. (2019). Comparison of Web Scraping Techniques: Regular Expression, HTML DOM and Xpath. Atlantis Highlights in Engineering (AHE). *Proceedings of the 2018 International Conference on Industrial Enterprise and System Engineering* (IcoIESE 2018), 2, 283–287. URL: https://www.researchgate.net/publication/332227853_Comparison_of_Web_Scraping_Techniques_Regular_Expression_HTML_DOM_and_Xpath
11. Harrell, N. B., Cruickshank, I., & Master, A. (2024). Overcoming Social Media API Restrictions: Building an Effective Web Scraper. *In Workshop Proceedings of the 18th International AAAI Conference on Web and Social Media*, 7 p. https://doi.org/10.36190/2024.72
12. Hernandez-Suarez, A., Sanchez-Perez, G., Toscano-Medina, K., Martinez-Hernandez, V., Sanchez, V., & Perez-Meana, H. (2018). A Web Scraping Methodology for Bypassing Twitter API Restrictions. https://doi.org/10.48550/arXiv.1803.09875
13. Himawan, A., Priadana, A., & Murdiyanto, A. (2020). Implementation of Web Scraping to build a Web-Based Instagram account data downloader application. *International Journal on Informatics for Development* (IJID), *9*(2), 59–65. https://doi.org/10.14421/ijid.2020.09201
14. Hrytsiuk, Yu. I. (2022). Software quality management system. *Ukrainian Journal of Information Technology*, 4(1), 01–20. https://doi.org/10.23939/ujit2022.01.001
15. Juszczak, A. (2023). The use of web-scraped data to analyse the dynamics of clothing and footwear prices. *Wiadomości Statystyczne the Polish Statistician*, *68*(9), 15–33. https://doi.org/10.59139/ws.2023.09.2
16. Khder, M. A. (2021). Web scraping or Web crawling: state of art, techniques, approaches and application. *International Journal of Advances in Soft Computing and Its Application*, *13*(3), 144–168. https://doi.org/10.15849/IJASCA.211128.11
17. Mustapha, S., Man, M., Wan Abu Bakar, W. A., Yusof, M. K., & Ahmad Sabri, I. A. (2024). A Demystified Overview of Data Scraping. *International Journal of Data Science and Advanced Analytics*, *6*(6), 290–296. https://doi.org/10.69511/ijdsaa.v6i6.205
18. Nyunt, K. T., & Khin, N. T. W. (2022). Web scraping for career analysis based on YouTube data APIs using Web content mining. *Journal of Information Technology, Research and Innovation*, *2*(1). URL: https://www.researchgate.net/publication/369094316_WEB_SCRAPING_FOR_CAREER_ANALY-

SIS_BASED_ON_YOUTUBE_DATA_APIs_USING_WEB_CONTENT_MINING
19. PLAY VINYL. Vinyl players, records, acoustics and interior items. (2024). URL: https://playvinyl.com.ua/
20. Rao, N. K., Naseeba, B., Challa, N. P., & Chakrvarthi, S. (2022). Web scraping (imdb) using Python. *Telematique*, *21*(1), 235–247. URL: https://www.researchgate.net/publication/368575445_WEB_SCRAPING_IMDB_USING_PYTHON
21. ROZETKA™ online store: the official website of the Rozetka online hypermarket in Ukraine. (2024). Recommendations Based on Your Views. URL: https://rozetka.com.ua
22. UN, S. M., & Mishra, S. P. (2023). Improving CAPTCHA recognition for enhanced web scraping. *International Journal for Research in Applied Science and Engineering Technology*, *11*(9), 881–884. https://doi.org/10.22214/ijraset.2023.55608
23. Vinyl Club Lviv. Vinyl Record Store. Vinyl club. (2024). Feel the real rhythm of your favorite music. [In Ukrainian]. URL: https://vinylclub.com.ua/
24. Vinyl records. Fonoteka. (2024). Hits for the winter holidays. [In Ukrainian]. URL: https://fonoteka.com.ua/
25. Vinyla. Vinyl records, vinyl players, speaker systems. (2024). Upcoming releases. [In Ukrainian]. URL: https://vinyla.com/

***A. A. Dzendzia, Ye. V. Levus, A. S. Vovk***

*Lviv Polytechnic National University, Lviv, Ukraine*

## ANALYSIS OF THE EFFECTIVE USE OF THE METHODS OF AUTOMATED DATA COLLECTION FROM WEBSITES

The task of automating data collection from the Internet is relevant to e-commerce, social media research, journalism, and any domain that requires fast collection of various types of data (structured and unstructured) in large volumes. Two key methods of automated data collection from websites are web scraping and Application Programming Interface (API). This study presents the results of data collection using the Harvester system, designed for automated data extracting through web scraping and API methods. Software implementation was done using Microsoft.NET technologies, AngleSharp, JSONpath, and React.js. Web scraping was performed using PuppeteerSharp to simulate user actions and AngleSharp for data parsing, while API access was structured through REST. The methods were evaluated based on their ability to process data on vinyl records, including price, artist, release title, and barcode. Experiments were conducted on five internet sources. From each source, ranging from 500 to 4000 records were retrieved, with a total of 14,995 records obtained. The experiments were carried out on an isolated server to ensure proper performance and reduce potential delays in the system components operation. Two following types of sources were used in the computational experiments: 1) Mass sources – where only the catalog page listing products was processed, containing the main product parameters; 2) Single sources – where additional data was collected from individual product pages after processing the catalog page to obtain more detailed information about each product. To analyze the performance of the methods, three main stages of website page processing were distinguished, in particular, loading, extraction, and idle time. The advantages and disadvantages of each method were analyzed. API requests showed significantly shorter result retrieval times compared to web scraping – 5 to 10 times faster. During the page loading stage, web scraping had greater variability in time compared to API, ranging from tenths of seconds to several tens of seconds. A combined method of automated data collection from websites was developed, which takes into account the characteristics of both web scraping and API methods, ensuring the necessary completeness of the retrieved data and speed of operation. A comparison of the combined method with web scraping and API methods was carried out. The future of web scraping lies in the implementation of intelligent algorithms for processing dynamic content, while APIs will continue to improve through support for new formats, such as GraphQL.

**Keywords:** data processing; webscraping; application programming interface (API); unstructured data; data extraction; downtime.