Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Аналіз програмно-алгоритмічних засобів для оптимізації процесу збору даних потенційних клієнтів компанії Just Export

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Якуб'як Ю. П.
(підпис) (прізвище та ініціали)

Керівник _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ _____
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_29_» _травня_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Якуб'як Юліяні Павлівні
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз програмно-алгоритмічних засобів для оптимізації процесу збору даних потенційних клієнтів компанії Just Export

Керівник роботи Млинко Богдана Богданівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_29_» _листопада_ 2024 року № _4/7-1139_____

2. Термін подання студентом завершеної роботи 30 травня 2025 р.

3. Вихідні дані до роботи Наукові публікації про розробку та реалізацію алгоритму оптимізації збору даних потенційних клієнтів

4. Зміст роботи (перелік питань, які потрібно розробити)
1 Аналітичний огляд програмно-алгоритмічних засобів для оптимізації потоків
2 Аналіз та вибір інструментів для оптимізації логістичних потоків
3 Реалізація скрипта для оптимізації логістичних потоків
4 Охорона праці та безпека в надзвичайних ситуаціях. Висновки. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, Мета, Об'єкт, Предмет дослідження, Завдання дослідження. 3 Актуальність теми. 4 Аналітичний огляд програмно-алгоритмічних засобів для оптимізації потоків. 5 Огляд Octoparse. 6 Огляд Netpeak Checker. 7 Огляд ParseHub. 8 Порівняльна таблиця програмно-алгоритмічних засобів 9. Аналіз та вибір інструментів для оптимізації логістичних потоків. 10 Розробка алгоритму формування цільової бази для клієнта. 11 Розробка алгоритму Скрипта для оптимізації логістичних потоків. 12 Реалізація та тестування скрипта для оптимізації логістичних потоків. 13 Висновки. 14 Завершальний слайд

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Сенчишин В.С., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викладач		

7. Дата видачі завдання 29 листопада 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	29.11.2024	*Виконано*
2.	Підбір та опрацювання наукових публікацій, збір	02.12.2024-10.01.2025	*Виконано*
	даних по темі роботи		
3.	Виконання дослідження згідно теми кваліфікаційної	13.01.2025-14.03.2025	*Виконано*
	роботи		
4.	Оформлення розділу «Аналітичний огляд програмно-	17.03.2025-28.03.2025	*Виконано*
	алгоритмічних засобів для оптимізації потоків»		
5.	Оформлення розділу «Аналіз та вибір інструментів для	31.03.2025-11.04.2025	*Виконано*
	оптимізації логістичних потоків»		
6.	Оформлення розділу «Реалізація скрипта для	14.04.2025-25.04.2025	*Виконано*
	оптимізації логістичних потоків»		
7.	Виконання завдання до підрозділу «Охорона праці»	28.04.2025-02.05.2025	*Виконано*
8.	Виконання завдання до підрозділу «Безпека в	28.04.2025-02.05.2025	*Виконано*
	надзвичайних ситуаціях»		
9.	Оформлення кваліфікаційної роботи	05.05.2025-09.05.2025	*Виконано*
10.	Нормоконтроль	12.05.2025-15.05.2025	*Виконано*
11.	Перевірка на плагіат	16.05.2024	*Виконано*
12.	Попередній захист кваліфікаційної роботи	19.05.2025	*Виконано*
13.	Захист кваліфікаційної роботи	30.05.2025	

Студент _____ Якуб’як Ю.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Млинко Б.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Аналіз програмно-алгоритмічних засобів для оптимізації процесу збору даних потенційних клієнтів компанії Just Export // Кваліфікаційна робота освітнього рівня «Магістр» // Якуб'як Юліяна Павлівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль 2025 // С. 102, рис. – 25 , табл. – 4, кресл. – 0, додат. – 3, бібліогр. – 52.

Ключові слова: алгоритм, оптимізація, цільові країни, парсинг, вигрузка, конверсія, клієнтська база.

Кваліфікаційна робота присвячена дослідженню існуючих рішень для розуміння алгоритму роботи, який є основою скрипта для парсингу даних за заданими критеріями. В першому розділі кваліфікаційної роботи ознайомлено з основними поняттями парсингу даних. Проведено аналіз доступних рішень. Ознайомлено з принципами та компонентами існуючих систем.

В другому розділі кваліфікаційної роботи проведено аналіз інструментів для розробки та роботи зі скриптом. Обрано платформу для розробки, середовище розробки та програму для роботи зі скриптом. Також проведено дослідження легальності парсингу даних в Україні.

В третьому розділі кваліфікаційної роботи розроблено алгоритм формування цільової бази для клієнта та алгоритм роботи скрипта. Наведено лістинги та описи основного функціоналу скрипта. Проведено тестування готового продукту.

ANNOTATION

Analysis of Software and Algorithmic Tools to Optimize the Data Collection Process of Potential Customers of the JUST EXPORT Company // The educational level "Master" qualification work // Yakubiak Yuliiana // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2025 // P. 102, fig. – 25, tables – 4, posters – 0, annexes – 3, ref. – 52.

Keywords: algorithm, optimization, target countries, parsing, unloading, conversion, customer base.

Thesis is devoted to the study of existing solutions for understanding the algorithm of work, which is the basis of the script for parsing data according to the specified criteria. In the first section of the qualification work, the basic concepts of data parsing are introduced. An analysis of available solutions is carried out. The principles and components of existing systems are introduced.

In the second section of the qualification work, an analysis of tools for developing and working with the script is carried out. A platform for development, a development environment and a program for working with the script are selected. A study of the legality of data parsing in Ukraine is also carried out.

In the third section of the qualification work, an algorithm for forming a target base for the client and an algorithm for working with the script are developed. Listings and descriptions of the main functionality of the script are provided. The finished product is tested.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Парсер – програма або скрипт для автоматизованого збору та обробки інформації з веб-сторінок.

Парсинг — автоматизований процес збору та обробки інформації з веб-сторінок.

API – application programming interface.

CSV – файловий формат, котрий є відмежовувальним форматом для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок.

HTML – стандартизована мова розмітки документів для перегляду веб-сторінок у браузері.

ЗМІСТ

ВСТУП

Актуальність теми: Інформаційні системи на сьогоднішній день мають великі перспективи практично в кожній сфері сучасного життя, зокрема в логістиці. Якщо розглядати підприємство з точки зору організованої системи, то всі її складові частини потребують безперебійного зв'язку між собою для створення інтегрованої та складної структури. Впровадження інформаційних технологій в логістиці та бізнес стратегіях, безпосередньо для збору даних про потенційних покупців з метою розширення клієнтської бази є актуальною задачею.

Аналіз предметної області показав, що більшість готових рішень надають доступ до потрібного функціоналу тільки за оплату за ширший функціонал. При цьому, окрім потрібного функціоналу відкривається багато зайвого, за що також потрібна оплата. Тому було вирішено зробити простіший скрипт, який буде задовольняти саме наші потреби.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Магістр» є оптимізація логістичних потоків шляхом розробки скрипта для пошуку та збору бази даних з потенційними клієнтами. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

1. Проаналізувати принципи роботи та компоненти програмно-алгоритмічних засобів для оптимізації потоків даних;
2. Провести аналіз інструментів для розробки скрипта для оптимізації логістичних потоків;
3. Дослідити легальність парсингу в Україні;
4. Розробити алгоритм, на основі якого буде реалізовано скрипт;
5. Розробити скрипт для парсингу даних, який буде мати за мету вирішення задачі парсингу гугл карт за допомогою ключових слів та країн;
6. Провести тестування готового продукту.

Об'єкт дослідження: процес оптимізації логістичних потоків.

Предмет дослідження: методи реалізації скрипта для збору даних за заданими критеріями з метою оптимізації логістичних потоків.

Наукова новизна одержаних результатів кваліфікаційної роботи полягає у тому, що отримав подальший розвиток метод оптимізації логістичних потоків.

Практичне значення одержаних результатів: розроблено скрипт для пошуку та витягування даних з Google Maps, який дозволяє зменшити час для обробки великого масиву даних про потенційних клієнтів та збільшити конверсію даних, тобто оптимізувати логістичні потоки.

Апробація результатів магістерської роботи. Основні результати проведених досліджень обговорювались на XIII Міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 2024 р.).

Публікації. Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додаток Б).

Структура й обсяг кваліфікаційної роботи. Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 51 найменувань та 3 додатків. Загальний обсяг кваліфікаційної роботи складає 90 сторінки, з них 61 сторінка основного тексту, який містить 25 рисунків та 4 таблиці.

1 АНАЛІТИЧНИЙ ОГЛЯД ПРОГРАМНО-АЛГОРИТМІЧНИХ ЗАСОБІВ ДЛЯ ОПТИМІЗАЦІЇ ПОТОКІВ

1.1 Поняття парсингу даних

Будь-який сучасний бізнес сьогодні доволі важко уявити без даних, але збирати дані вручну є доволі часозатратним заняттям. Тому для цього використовують парсинг – автоматизований процес збору необхідних даних з інтернету.

Парсер – програма, за допомогою якої відбувається збір даних з веб-сторінок. Він витягує дані з відповідних сайтів та форматує їх у таблицю Excel або файл CSV. Таким чином надаючи можливість користувачу моніторити процес та коригувати його за потребою.

Парсинг даних – процес, який надає можливість отримувати швидкий доступ до доволі великих масивів інформації.

Існують доволі різні підходи та види парсингу даних з мережі Інтернет. Вони можуть відрізнятися тим, як саме користувач отримує дані з ресурсу:

HTML-парсинг – це один із найпоширеніших методів витягування даних. Програма надсилає запит, у відповідь отримує сирий код сторінки, після чого проводить його «розбір» та знаходить потрібні елементи відповідно до заданих правил чи структур.

API-парсинг. Інколи сайти надають можливість доступу до їх даних за допомогою API. Такий метод парсингу є надійним та вважається «легальним», адже програма отримує доступ до відкритих даних сайту.

Google Sheets – доволі простий метод парсингу даних, адже має функцію IMPORTXML, що надає можливість витягувати потрібні дані з сайтів.

1.2 Аналіз Octoparse

Octoparse є одним із інструментів для парсингу даних, який надає можливість автоматизувати процес збору інформації з веб-ресурсів. Octoparse створено для користувачів, які мають різні рівні підготовки. Завдяки функції "drug and drop", створення завдань є доволі простим і не потребує знання кодингу. Це робить інструмент доступним навіть для новачків.

Octoparse має змогу автоматично розпізнавати структуру даних на веб-сторінках, працювати з динамічним контентом (наприклад, JavaScript) та виконувати завдання без втручання користувача. Це дозволяє швидко отримувати дані навіть зі складних сайтів.

Інструмент дозволяє виконання завдань на хмарних серверах, що забезпечує:

- Можливість одночасної обробки великих обсягів інформації;
- Резервне копіювання даних. Ця функція є доволі корисною для бізнесів, які працюють з масштабним збором даних.

Отримані дані експортуються в різні формати:

- CSV;
- Excel;
- Бази даних;
- API.

Octoparse також дозволяє налаштовувати поля даних, наприклад, перейменовувати їх, об'єднувати або розділяти.

Для розробників є передбачена інтеграція Octoparse з найпопулярнішими мовами програмування, такими як Python, Node.js. Окрім цього, інструмент надає API для автоматизації процесів. Octoparse дозволяє використання проксі-серверів для обходу CAPTCHA та інших обмежень на веб-сайтах. Це доволі важливо при роботі з сайтами, на яких передбачено механізми захисту.

Інструмент проводить збір даних із сайтів із пагінацією, нескінченним прокручуванням або кнопками "завантажити більше". Це надає можливість отримувати дані навіть з динамічних веб-ресурсів.

Octoparse може взаємодіяти зі сторінками:, авторизуватися, вводити ключові слова в пошукові форми, вибирати опції з спадних меню. Це робить його універсальним інструментом для складних завдань.

Для виконання завдання необхідно ввести посилання на ресурс, з якого користувач хоче отримати дані. Відбудеться завантаження ресурсу (рисунок 1.1).

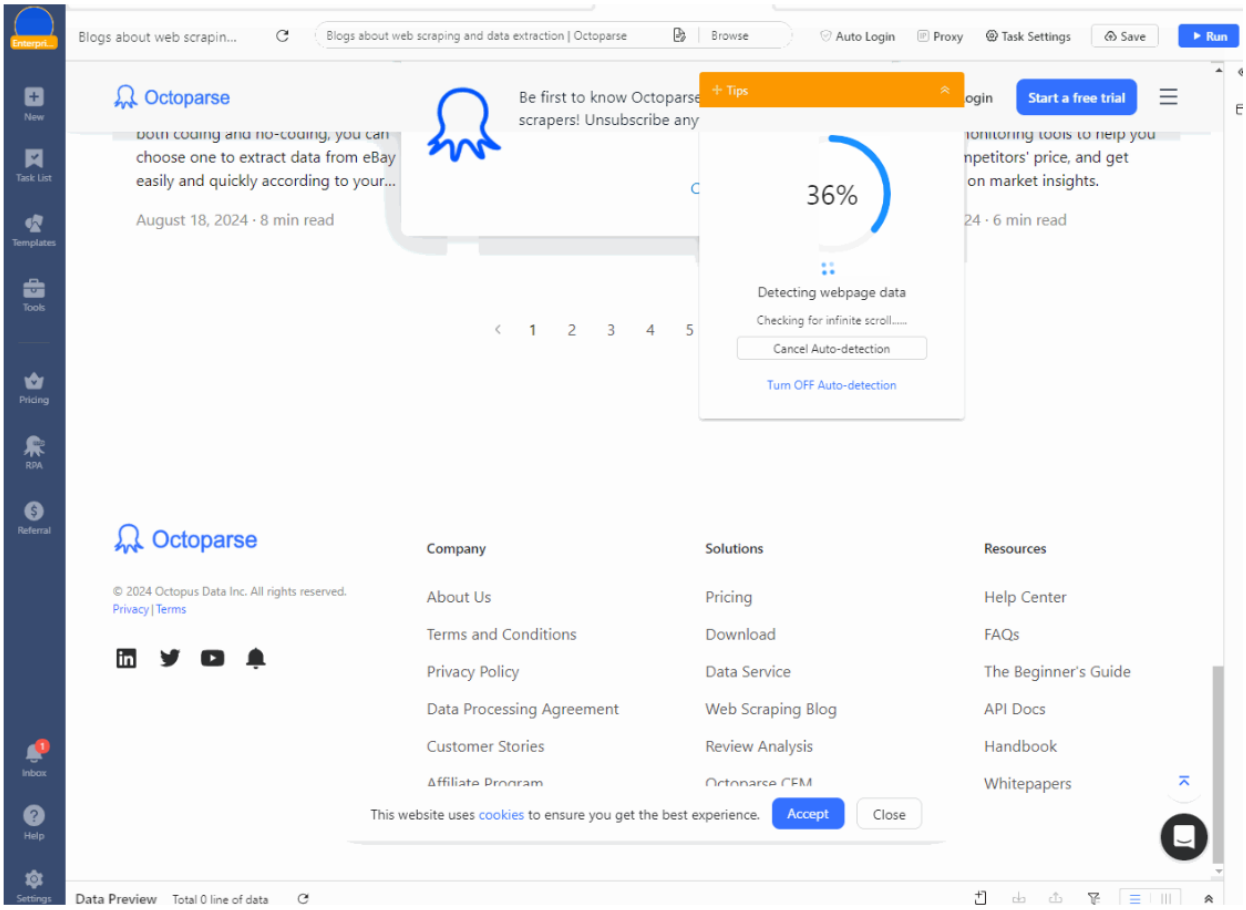


Рисунок 1.1 – Завантаження ресурсу

Як тільки сторінка завантажиться у вбудованому браузері, можна буде обрати «Auto-detect web page data» для автоматичного виділення всіх даних на сторінці. Парсер одразу виділить всі дані (рисунок 1.2). Секція «Data preview» надає можливість відсіювати непотрібні дані.

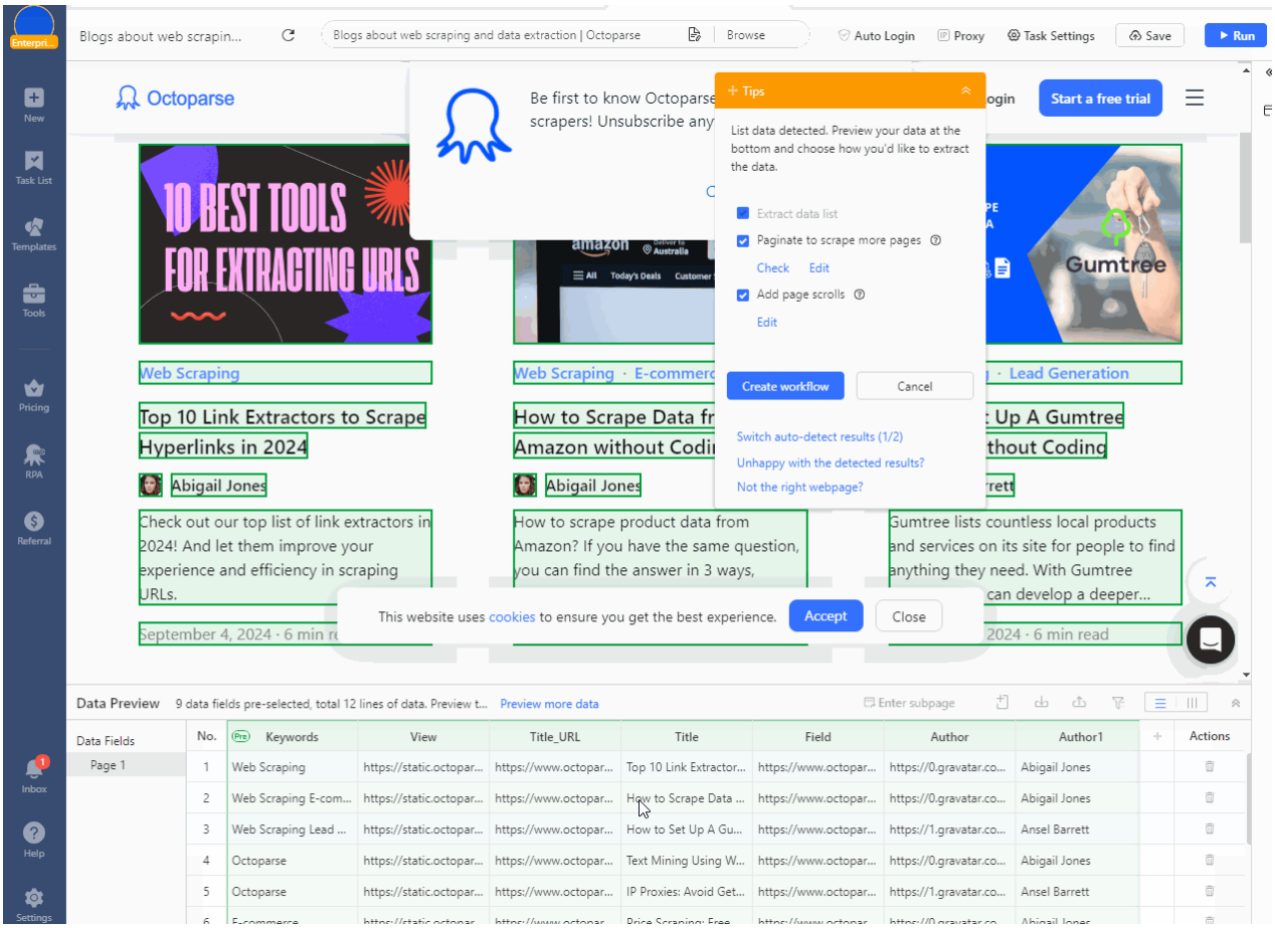


Рисунок 1.2 – Автоматичне виділення всіх даних на сторінці

Користувач може створювати та редагувати налаштування автоматичного виділення даних. Для цього треба використати «Tips panel» та перевірити всі вказані там налаштування. Тут можна змінити скрол сторінки, додати витягування даних з кількох сторінок, перевірити чи правильно була вибрана кнопка для переходу від однієї сторінки до іншої, створювати робочі потоки для кращого розуміння структури сторінки.

Окрім автоматичного виділення всіх даних на завантаженій сторінці, ресурс надає можливість виділяти необхідні дані вручну (рисунок 1.3).

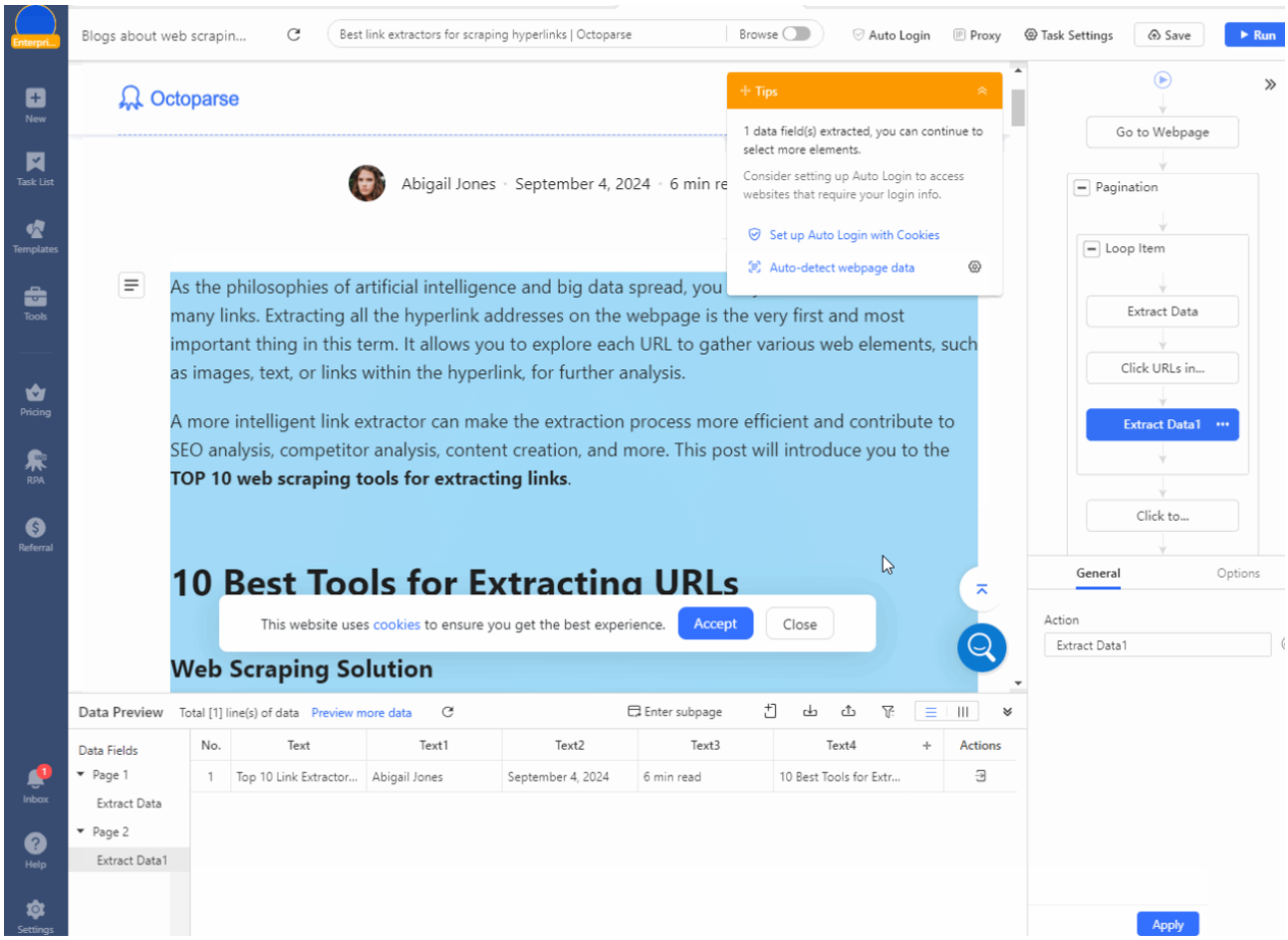


Рисунок 1.3 – Процес збору інформації вручну

Надається можливість змінювати та відфільтровувати отриману інформацію за допомогою «Clean Data». Цей функціонал надає можливість змінювати видачу даних відповідно до потреби користувача. Наприклад, користувачу потрібна видача дати та часу в конкретному форматі, відмінному від видачі дати та часу на сторінці. Цей інструмент дозволить змінити формат дати та часу всього за кілька кліків (рисунок 1.4).

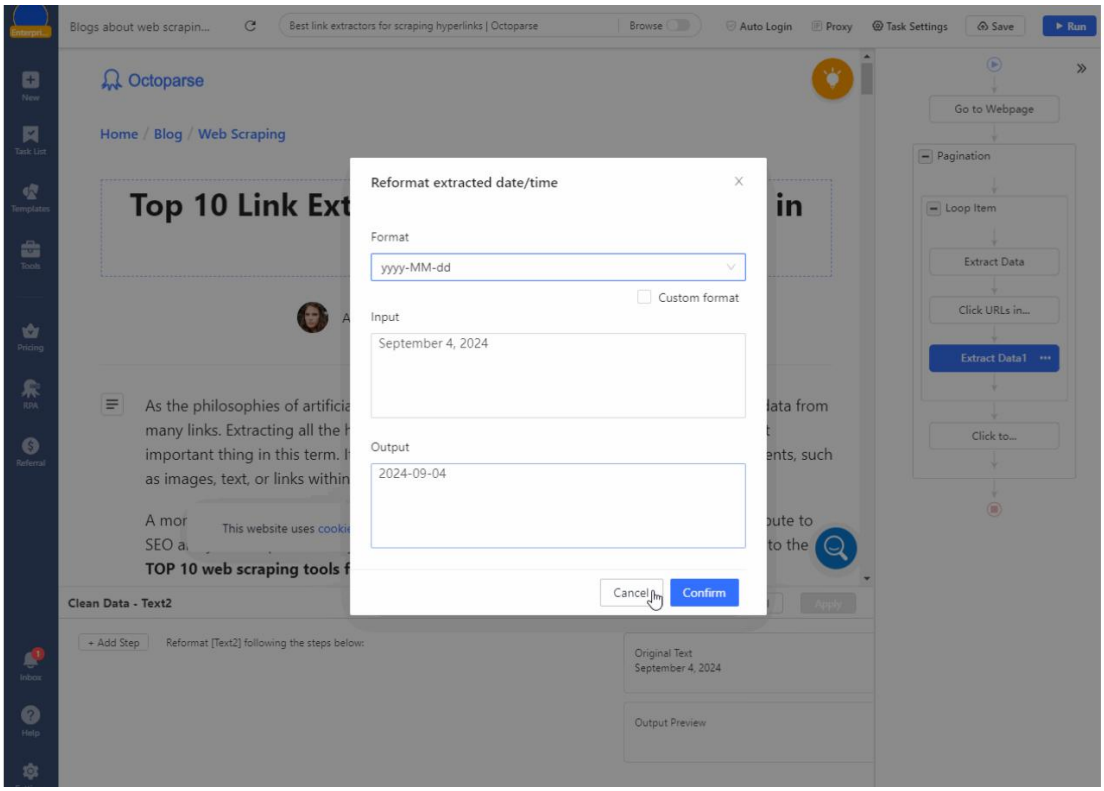


Рисунок 1.4 – Зміна формату видачі даних

Функціонал програми передбачає також експорт даних в різних форматах. Для цього необхідно в «Task List» натиснути «Export Data» та обрати формат, зручний для користувача (рисунок 1.5).

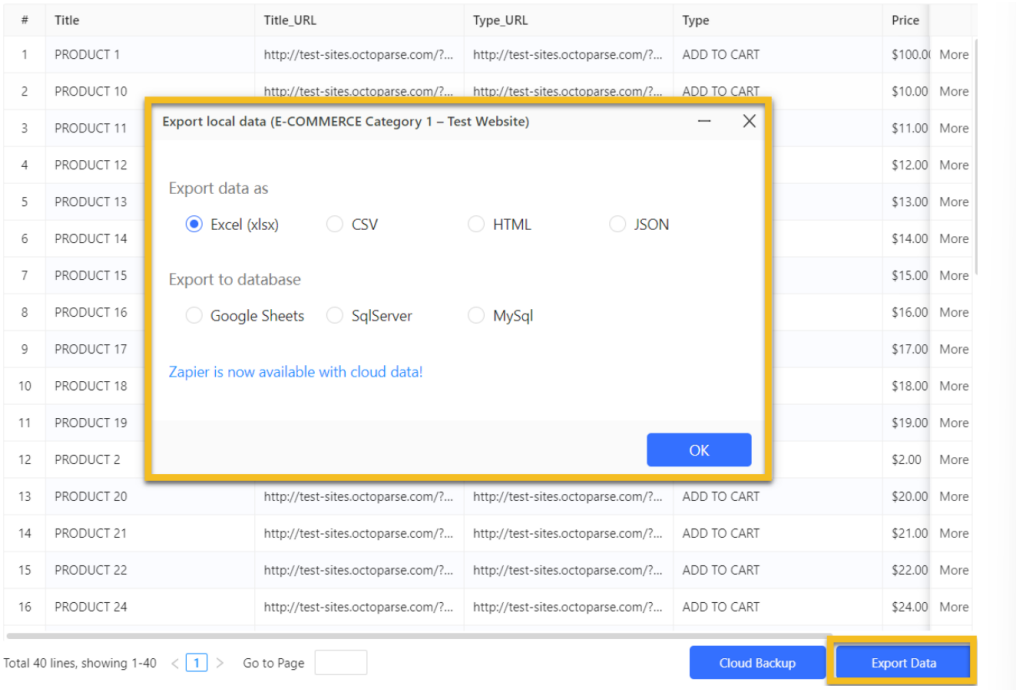


Рисунок 1.5 – Вікно вибору формату експорту даних

Octoparse є хорошим інструментом для автоматизації збору даних з веб-ресурсів. Він надає широкий набір функцій, що робить його однаково корисним як для новачків, так і для професіоналів. Завдяки гнучкості й хмарній обробці, цей застосунок може бути інтегрований у будь-який бізнес-процес, який потребує регулярного збору інформації [1].

1.3 Огляд Netpeak Checker

Netpeak Checker є комплексом інструментів, які призначені для аналізу SEO-метрик веб-сайтів. Ця система допомагає користувачам отримати потрібну інформацію про власний сайт та сайт конкурентів, що дозволяє приймати обґрунтовані рішення для покращення їхньої онлайн-присутності та залучення більшої кількості клієнтів.

Netpeak Checker є інструментом, який створений для допомоги користувачам під час дослідження важливих показників веб-сайтів і для отримання цінних даних для SEO-оптимізації. Він надає доволі різноманітну інформацію, яка включає:

a) Інформацію про стан On-Page сайту;
b) Кількість слів та символів на сторінках сайту;
c) Парсинг контактних даних;
d) Дані DNS;
e) Інформацію про домен з даними Whois;
f) Дані відомих SEO-сервісів;
g) Дані про трафік сайту;
h) Дані результатів пошукових систем та індексації.

Всі вищезгадані метрики досить зручно зібрані праворуч в інтерфейсі програми. Для отримання даних треба просто відзначити потрібні пункти. Вікно параметрів застосунку Netpeak Checker зображено на рисунку 1.6.

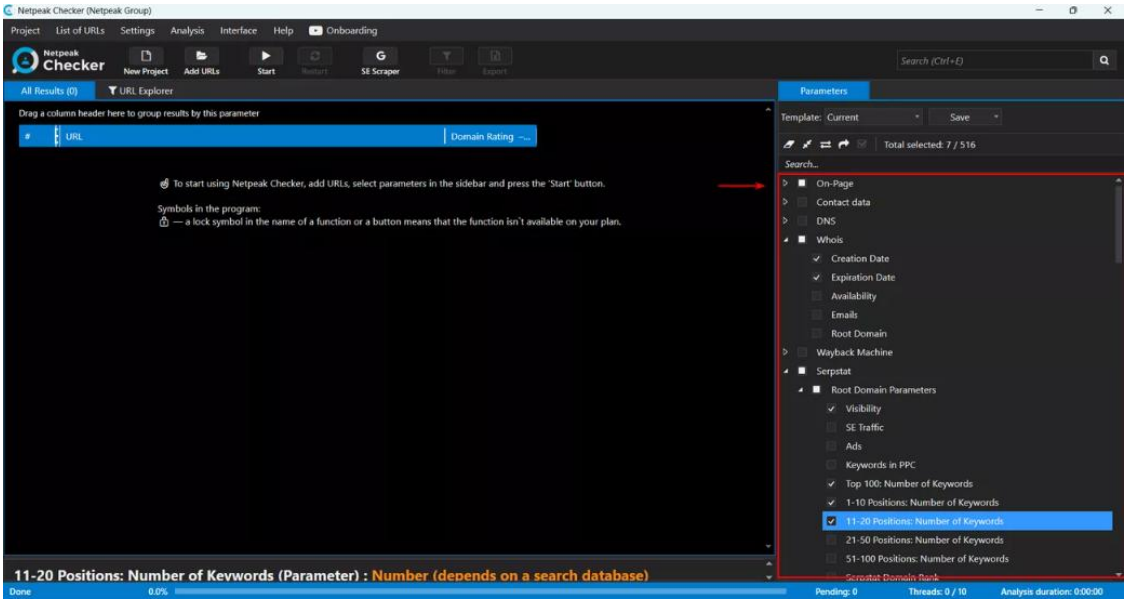


Рисунок 1.6 – Вікно параметрів Netpeak Checker

Інтерфейс Netpeak Checker інтуїтивно зрозумілий. Інтерфейс застосунку зображено на рисунку 1.7.

1. У верхній частині вікна знаходиться командна панель з основними операційними командами.
2. Всі дані поділені на вкладки з можливістю фільтрування;
3. Результати представлені у табличному вигляді;
4. Праворуч у сайдбарі у списку перераховані всі доступні для аналізу метрики;
5. Над сайдбаром знаходиться рядок пошуку.

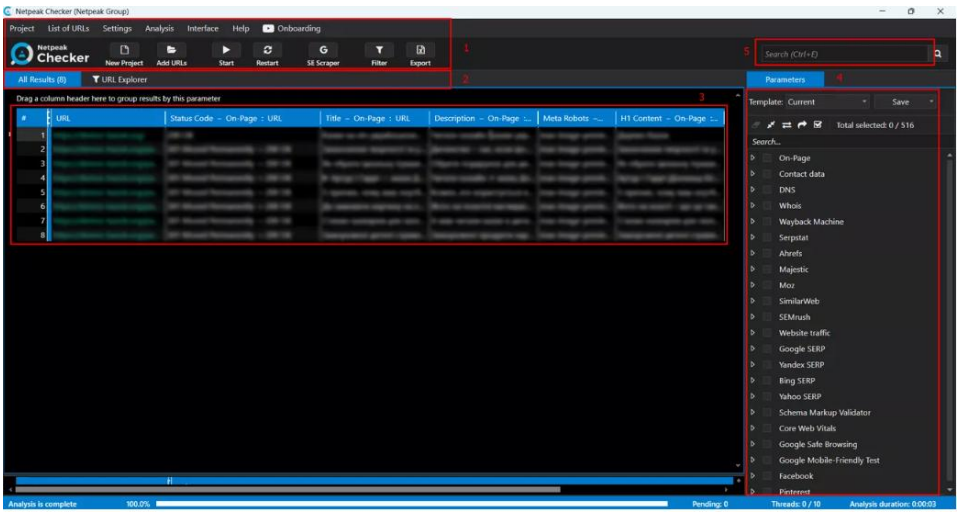


Рисунок 1.7 – Інтерфейс застосунку

Робота з даними відбувається за принципом проєкту. Для кожного окремого набору даних створюється окремий проєкт, який можна зберігати, експортувати та імпортувати. Можна відкривати Netpeak Checker у кількох вікнах, що надає можливість працювати з кількома проектами одночасно. Усі операції з проектами доступні у вкладці «Project» командної панелі (зображеної на рисунку 1.8):

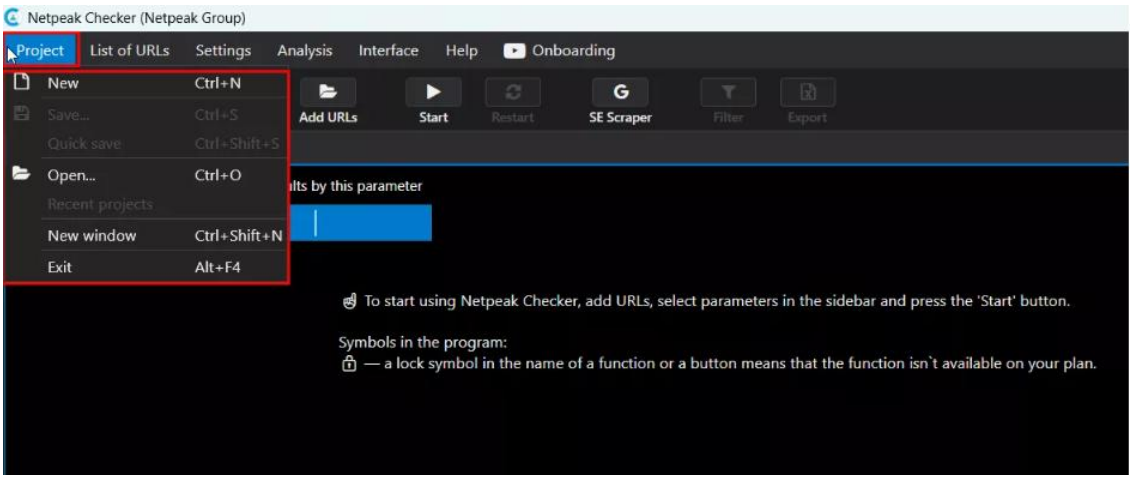


Рисунок 1.8 – Вкладка «Project»

Всі дані можна відфільтровувати за допомогою операторів «Ввімкнути» та «Вимкнути», вибравши необхідні метрики та вказавши умови фільтрування. Для швидкої фільтрації даних можна використовувати рядок пошуку. Приклад фільтрування даних наведено на рисунку 1.9. Достатньо ввести в рядок пошуку потрібну метрику/елемент і всі дані по ній автоматично відфільтруються (пошук відбувається по всій таблиці із метриками).

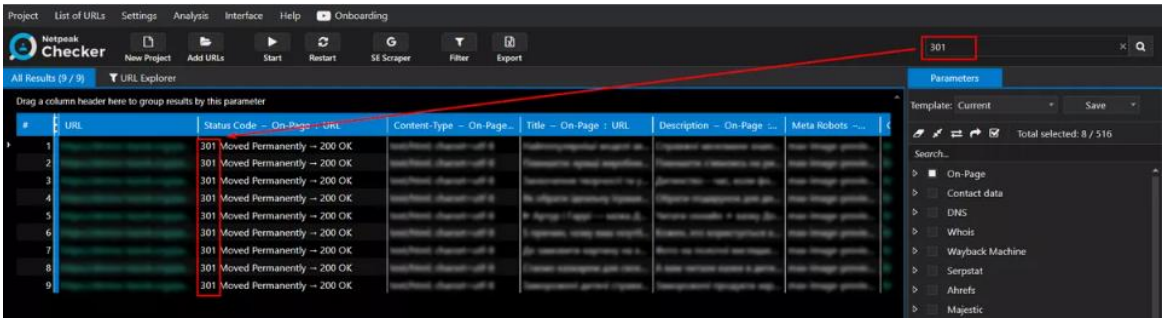


Рисунок 1.9 – Фільтрування даних

Слід окремо відмітити функціонал експорту даних до Google Drive, що надає можливість зберігати результати аналізу у форматі Google Sheets. Для цього потрібно авторизуватися у своєму обліковому записі Google і відмітити необхідний метод експорту в налаштуваннях програми. Налаштування експорту даних до Google Drive зображено на рисунку 1.10.

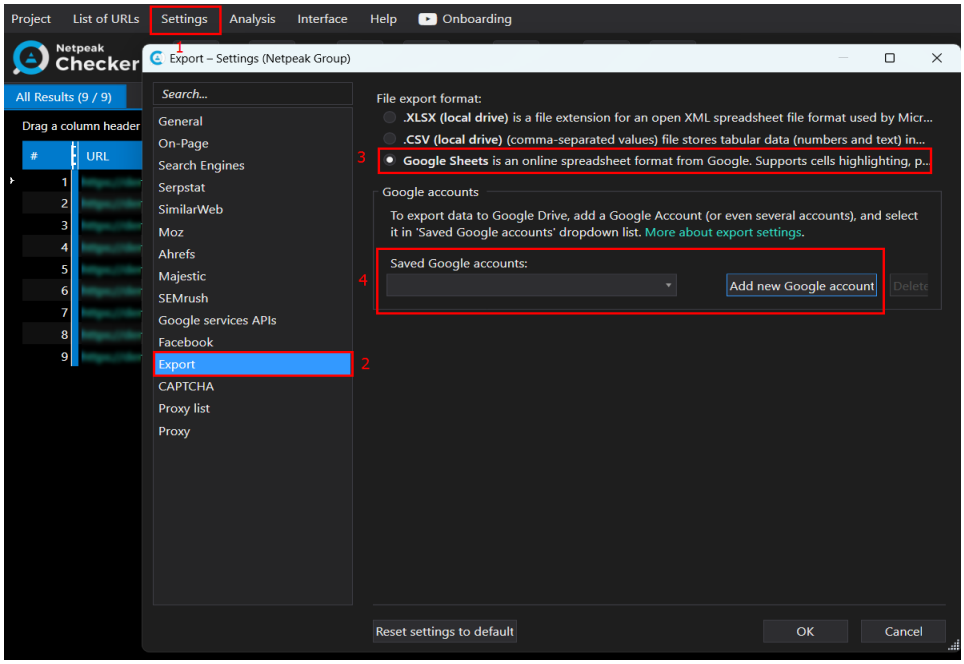


Рисунок 1.10 – Налаштування експорту даних до Google Drive

Ще однією важливою функцією програми є можливість парсингу пошукової видачі, що надає можливість за переліком ключових фраз спарсити результати топ-1, топ-3, топ-10 та топ-50 видачі пошукових систем Google, Bing та Yahoo. Для цього треба вставити (або імпортувати) потрібні ключові слова та налаштувати пошукові параметри системи. Усі результати пошукової видачі користувач отримає у табличному форматі, який можна експортувати для подальшого аналізу. Налаштування парсингу пошукової видачі зображено на рисунку 1.11.

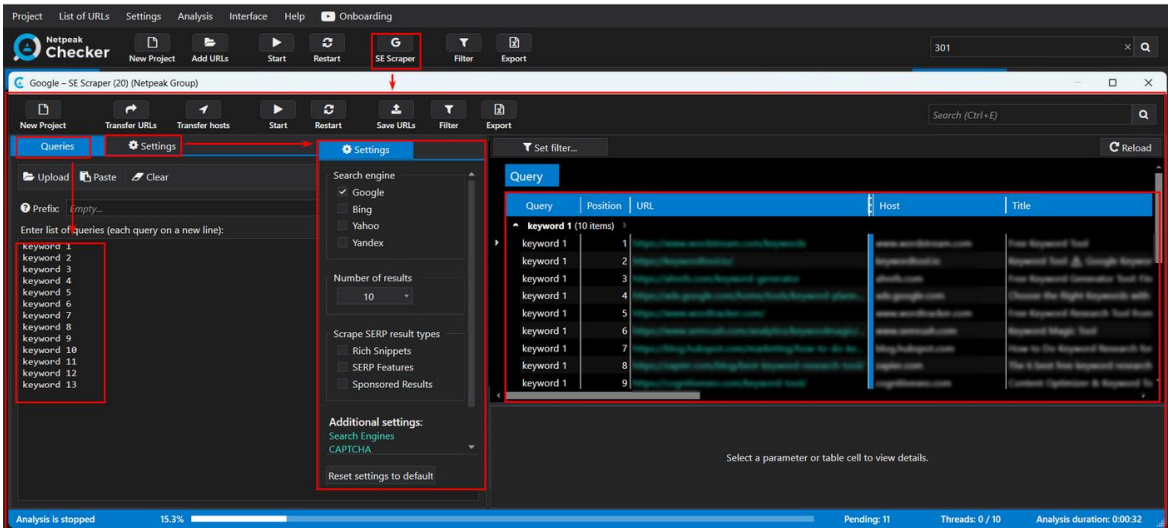


Рисунок 1.11 – Налаштування парсингу пошукової видачі

Для тривалого парсингу даних із сайтів та результатів видачі пошукових систем без збоїв програма надає можливість використовувати проксі-сервери та API відомих сервісів для автоматичного розгадування перевірки на робота. Для цього треба заповнити відповідні поля у налаштуваннях програми. Поля, необхідні для налаштування подано на рисунку 1.12.

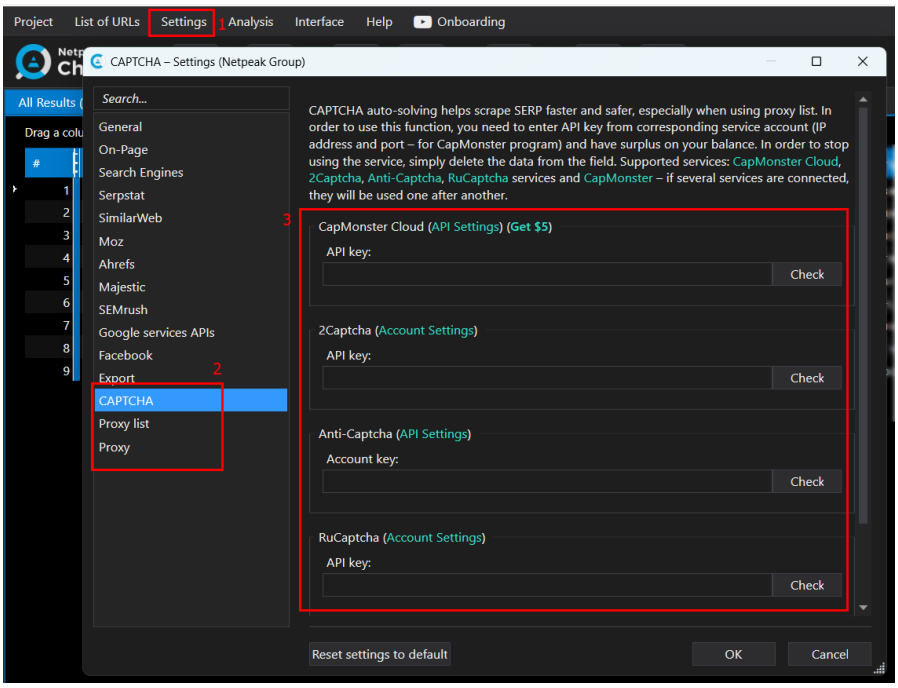


Рисунок 1.12 – Налаштування проксі

Netpeak Checker надає можливість передавати дані в Netpeak Spider, доповнюючи його звіти новими відомостями, що робить сайт більш комплексним та повним. Для цього треба експортувати дані Netpeak Checker у CSV форматі та вибрати пункт «Upload parameters to enrich data…» у Netpeak Spider. Всі дані будуть додані до метрик, що вже містяться в таблиці. Експорт даних відображено на рисунку 1.13.

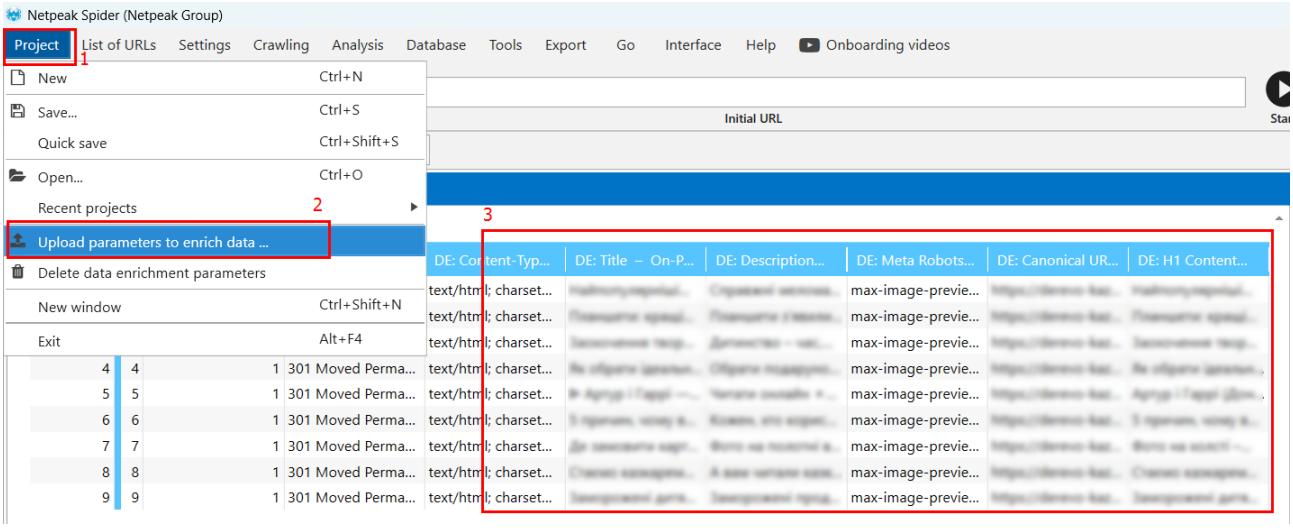


Рисунок 1.13 – Експорт даних з Netpeak Checker

Netpeak Checker є багатофункціональним інструментом для аналізу веб-сайтів, що дозволяє SEO-фахівцям, інтернет-маркетологам та веб-майстрам аналізувати великі обсяги даних, важливих для просування своїх сайтів в органічному пошуку. Завдяки інтуїтивно зрозумілому інтерфейсу, розширеним функціям та інтеграції з іншими SEO-інструментами, він цілком може стати помічником для тих, хто прагне покращити ефективність свого веб-сайту [2].

1.4 Ознайомлення з ParseHub

Parsehub – це інструмент для веб-парсингу, який забезпечує ефективне витягування даних із веб-сайтів без попередніх знань у сфері програмування. Цей інструмент застосовує передові підходи машинного навчання для аналізу та тлумачення веб-сайтів, що змінюються динамічно, включаю ті, що

використовують технології JavaScript і AJAX. Parsehub дозволяє налаштовувати проєкти для парсингу: він підлаштовується під різні типи даних та забезпечує роботу навіть із тими сайтами, які вимагають автентифікації користувача або введення спеціальних даних для доступу до інформації.

Parsehub набув доволі широкого застосування в різних галузях завдяки своїй здатності підлаштовуватись під складні завдання та умови:

- Аналітики та маркетологи використовують цей інструмент для спостереження за цінами та аналізу поведінки споживачів, що сприяє покращенню стратегій ціноутворення і популяризації товарів.
- У сфері фінансів Parsehub використовують для збору фінансових показників і аналізу тенденцій ринку, що допомагає приймати обґрунтовані рішення стосовно інвестицій.
- Дослідники та академічні установи використовують його для автоматизації збору даних із різноманітних наукових публікацій і баз даних, прискорюючи процес наукових досліджень.

Проте, застосування цього парсера можна знайти і в інших сферах, наприклад, SEO, електронна комерція, репутаційний менеджмент.

Парсер має широкий набір різних параметрів і дає змогу реалізувати иайже будь-які завдання з парсингу. Також варто виділити алгоритми машинного навчання, які призначені для розпізнавання шаблонів у даних і структурах сторінок, що полегшує процес налаштування парсингу і підвищує точність вилучення даних. Окрім цього, користувачі можуть створювати і налаштовувати проєкти за допомогою інтерфейсу.

До процесу автоматизації в Parsehub можна віднести два компоненти: API і планувальник завдань.

API надає змогу автоматизувати процеси збору даних, інтегруючи зібрані дані в зовнішні системи та застосунки. Розробники можуть використовувати API для запуску й керування проєктами вигрузки, отримання результатів в режимі реального часу та їхнього експорту в потрібному форматі. Це надає можливість безшовної інтегрувати зібрані дані у бізнес-процеси, мінімізуючи необхідність

втручання людини. На сайті розробника можна знайти детальну документацію стосовно інтеграції та застосування API.

Планувальник завдань дає можливість налаштовувати автоматичне виконання завдань парсингу згідно із вказаним графіком. Це містить в собі щоденне, щотижневе або щомісячне виконання завдань, а також запуск процесів парсингу в задані дати та час. Планувальник спрощує керування даними, даючи гарантію, що інформація завжди буде оновлена і доступна в потрібний час без необхідності постійно контролювати і вручну запускати проекти.

Ці інструменти разом формують сильну систему автоматизації Parsehub, дозволяючи користувачам масштабувати й покращувати процеси збору даних.

Parsehub має сучасні інструменти для масштабованого й ефективного збору даних з великої кількості пов'язаних веб-сторінок. З його допомогою, користувачі можуть керувати проектами парсингу таким чином, щоб автоматично переходити за внутрішніми посиланнями сайту, періодично витягувати дані з кожної відкритої сторінки та вигружати їх у централізовану базу даних. Платформа підтримує роботу з веб-сторінками генерованими динамічно, використовуючи JavaScript і AJAX, що дає змогу витягувати дані навіть із найскладніших веб-сайтів.

Можливість конфігурації дій на сайті поширюється не тільки на переходи за посиланнями, а й на заповнення форм введення, авторизацію на сайтах і обробку пагінації. Ці компоненти автоматизації сприяють точному і глибокому аналізу структур інформації, що забезпечує не тільки витягування вмісту, але і подальшу його структуризацію та класифікацію.

Платформа підтримує експорт даних у популярних форматах(Excel, JSON та через API)

Експорт в Excel відбувається у вигляді таблиць. Цей формат підходить для тих, кому потрібне візуальне представлення даних для розрахунків або складання звітів.

Експорт у JSON забезпечує гнучкість в керуванні даними, спрощуючи роботу з веб-додатками та підтримку безлічі програмних мов. Формат підходить для веб-розробників, яким потрібне зручне передавання даних між системами.

Використання API дозволяє розширити можливості автоматизації, забезпечуючи доступ до інформації у реальному часі та даючи змогу використовувати їх у корпоративних або зовнішніх додатках. Це критично важливо для систем, яким потрібна актуальність даних, і дає змогу розробникам налаштовувати обробку даних під різні специфічні завдання. Ці механізми експорту значно спрощують процес керування та аналізу даних.

Інтерфейс Parsehub досить простий і спрямований на полегшений менеджмент і запуск проектів. Усі елементи керування розташовані на лівій панелі.

У вкладці Projects (рисунок 1.14) користувачеві доступно кілька варіантів взаємодії, а саме:

- Створення нового проекту;
- Імпорт уже готового;
- Вивантаження всіх активних.

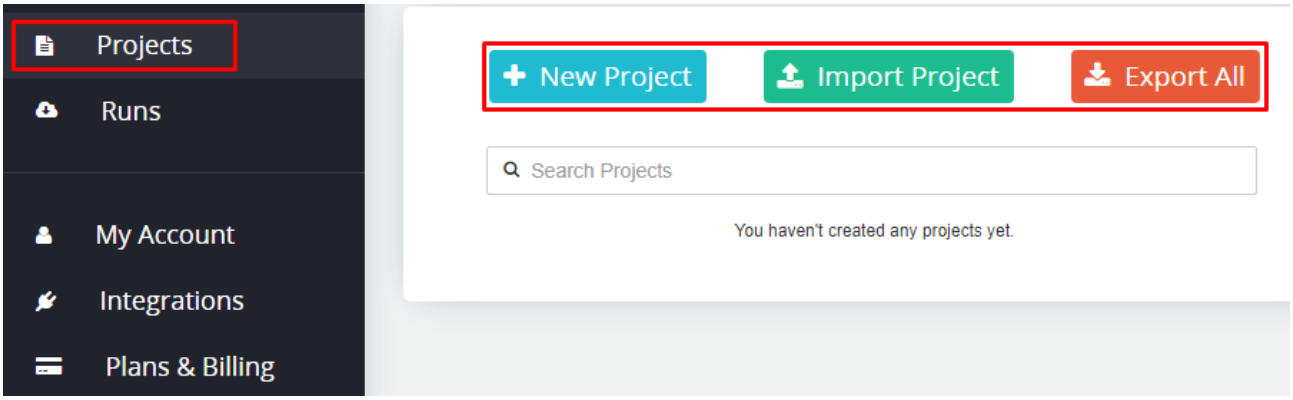
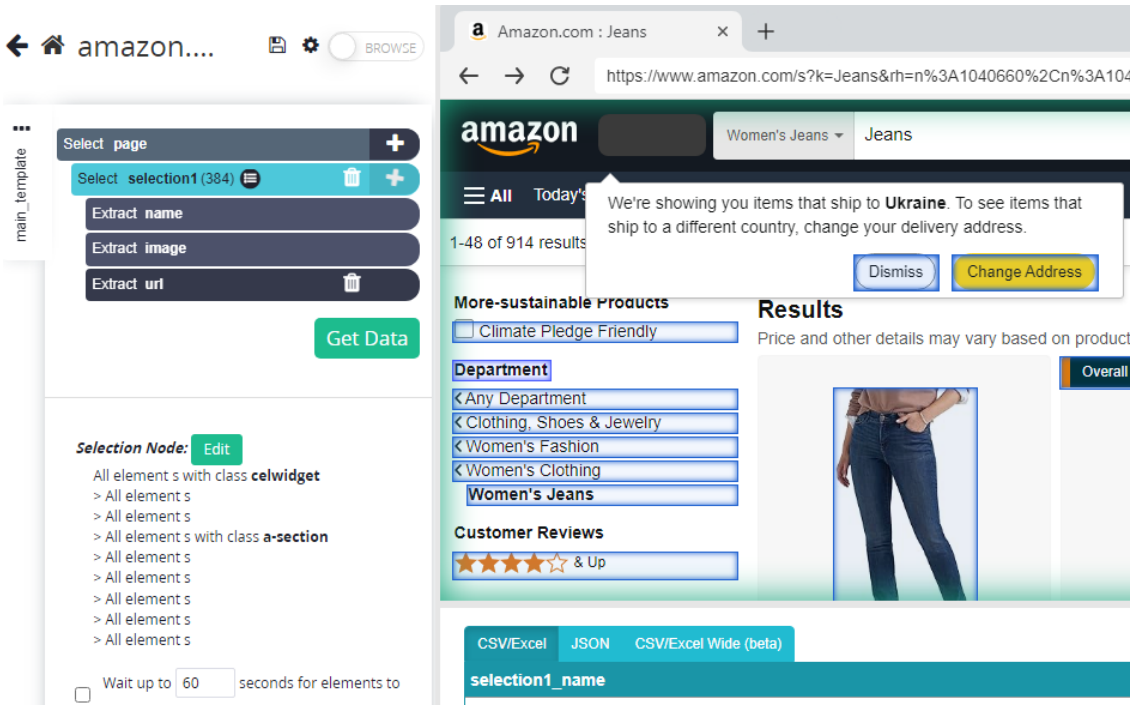


Рисунок 1.14 – Вкладка Projects

Після натискання на "New Project" відкриється нове робоче вікно (рисунок 1.15). Тут є можливість вставити посилання цільового сайту і запустити створення проєкту.



Вкладка Runs надає можливість моніторингу статусу виконання проєктів, включаючи з кількість запущених і успішно завершених. Вікно вкладки Runs зображено на рисунку 1.16.

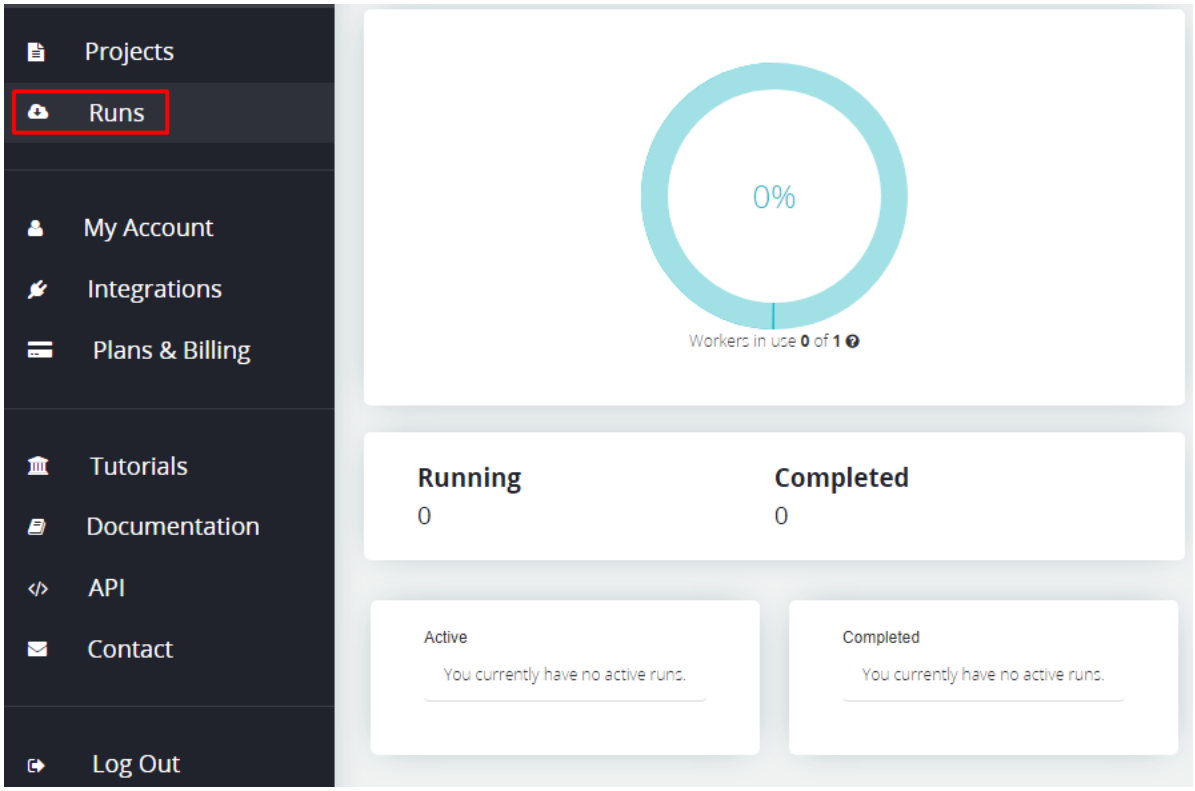


Рис.1.16 – Вікно вкладки Runs

Використання проксі-серверів під час процесу парсингу даних із сайтів доволі важливе з кількох причин:

Перше, проксі-сервери надають можливість замаскувати вихідну IP-адресу користувача, що дає змогу вибрати проксі з країни, де цільовий сервіс не заблоковано.

Друга важлива функція - ротація IP-адрес, яку забезпечено через проксі менеджер. Це означає, що кожен новий запит до сайту може виходити з іншої IP-адреси, що допомагає обійти обмеження на кількість запитів до веб-сайтів і запобігає блокуванню за IP.

Для роботи з парсерами рекомендується використовувати тільки приватні проксі-сервери, оскільки вони працюють стабільно, і забезпечують високий рівень довіри з боку цільових ресурсів [3] .

Проведемо порівняльний аналіз розглянутих ресурсів, результати якого наведено в таблиці 1.1:

Таблиця 1.1 – Порівняння ресурсів

Характеристики	**Octoparse**	**Netpeak Checker**	**ParseHub**
1	2	3	4
Опис	Інструмент для автоматизації збору веб-даних.	Інструмент для аналізу SEO та збору даних з веб-сторінок.	Інструмент для збору динамічного контенту.
Функції	Автоматизація, хмарна обробка, експорт у CSV/Excel.	SEO-аналіз, перевірка доменів, інтеграція з іншими інструментами.	Робота з динамічним контентом, інтеграція з API.
Переваги	Простий інтерфейс, підходить для новачків.	Широкий функціонал для SEO-аудиту, зручність порівняння з конкурентами	Складність для новачків, обмеження у безкоштовній версії.
Недоліки	Висока вартість, обмеження у безкоштовній версії.	Обмежена безкоштовна версія, платна підписка для повного доступу	Обмежена можливість налаштування складних сценаріїв, платна підписка
Ціна	Безкоштовна версія, платні плани від $75/місяць.	Безкоштовна версія, платні плани від $19/місяць.	Безкоштовна версія, платні плани від $89/місяць.
Тип інструменту	Веб-скрапінг	SEO аналіз та аудит	Веб-скрапінг

Продовження таблиці 1.1

1	2	3	4
Налаштування	Інтуїтивний інтерфейс, налаштування через API.	Просте налаштування через інтерфейс.	Візуальний інтерфейс, налаштування через API.
Види даних	HTML, XML, динамічний контент.	SEO-дані, веб-метрики.	HTML, JavaScript.
Рівень складності	Низький, підходить для новачків.	Низький, підходить для новачків.	Середній, потребує базових знань.
Можливість безкоштовного використання	Так, з обмеженнями.	Так, з обмеженнями.	Так, з обмеженнями.
Підтримка та оновлення	Регулярні оновлення, підтримка через документацію.	Регулярні оновлення, підтримка через документацію.	Регулярні оновлення, підтримка через документацію.

З результатів порівняльного аналізу видно, що дані засоби мають доволі широкий функціонал, проте він є обмеженим в безкоштовній версії. Тому прийнято рішення розробити власний скрипт, який надасть можливість витягувати дані потенційних клієнтів.

1.5 Висновки до розділу 1

В першому розділі кваліфікаційної роботи було проведено аналітичний огляд програмно-алгоритмічних засобів, зокрема, порівняння функціоналу парсерів (результати представлено в таблиці 1.1). Виходячи з розглянутих прикладів було сформовано розуміння основного функціоналу та завдань, які має виконувати майбутній скрипт, а саме:

1. Формування масиву запитів;
2. Прогін масиву в Google Maps;
3. Запис результатів в кінцевий файл.

Було поставлено задачу дослідити специфіку та особливості роботи існуючих парсингових систем, проаналізувати принципи їх роботи та дослідити їх структуру. А також розробити скрипт для парсингу Google Maps.

2 АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТІВ ДЛЯ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ ПОТОКІВ

2.1 Оптимізація логістичних потоків

Застосування інформаційних систем у логістиці має великі перспективи, оскільки підприємство, як система, потребує взаємозв'язку між її складовими частинами для створення інтегрованої та складної структури. Запровадження інформаційних технологій у бізнес стратегіях та логістиці, а саме для збору даних про компанії та постачальників з метою розширення клієнтської бази є доволі актуальною задачею.

Just Export допомагає українському бізнесу виходити на міжнародний ринок, а пошук контактних даних потенційних клієнтів є важливим етапом у процесі експорту. Парсери, або автоматизовані програми для збору інформації, дозволяють швидко й ефективно знаходити дані, необхідні для комунікації з партнерами по всьому світу.

Використання парсерів для автоматизації збору даних:

Парсинг веб-сайтів та бізнес-каталогів. Парсери можуть автоматично збирати контактну інформацію з відкритих джерел, таких як LinkedIn, Crunchbase, Yellow Pages, Alibaba та інші бізнес-реєстри. Вони шукають номери телефонів, email-адреси, соціальні мережі, щоб компанія могла швидко налагодити комунікацію.

Збір даних із відкритих баз та форумів. У деяких країнах існують публічні бізнес-реєстри, де можна знайти офіційні контакти компаній. Парсери можуть збирати цю інформацію та автоматично формувати базу даних для експорту Just Export.

Аналіз конкурентів та постачальників. завдяки парсингу можна знаходити партнерів та клієнтів, які співпрацюють із конкурентами, аналізувати їхню структуру бізнесу та виходити на нові ринки з найкращими пропозиціями.

Фільтрація та перевірка контактів. Зібрані дані проходять внутрішню перевірку – парсери можуть перевіряти валідність email-адрес, аналізувати активність профілів у соцмережах та виключати неактуальні контакти.

Інтеграція з CRM та автоматизація комунікації. Дані з парсерів можуть автоматично імпортуватися в CRM-систему, наприклад, HubSpot, Salesforce або Pipedrive, що дозволяє Just Export ефективно керувати контактами та запускати email-кампанії.

Впровадження парсерів у систему Just Export:

Розробка парсерів для бізнес-каталогів, LinkedIn та торгових платформ.

Інтеграція з CRM для автоматичного імпорту контактів.

Впровадження механізмів перевірки даних (валідація email, активність клієнтів).

Оптимізація бізнес-аналітики для виходу на нові ринки.

Завдяки парсерам Just Export може значно прискорити процес пошуку клієнтів та налагодження міжнародних партнерств!

2.2 Порівняння платформ для виконання

Проведемо порівняльний аналіз платформ для виконання.

Node.js.

Node.js використовує движок V8 від Google, який компілює JavaScript в машинний код, що забезпечує доволі високу швидкість виконання. Його асинхронна, неблокуюча модель вводу-виводу заснована на подіях та колбеках, що робить його ідеальним для роботи з великою кількістю одночасних запитів, наприклад у веб-серверах чи потоковій обробці даних.

Переваги:

Єдиний стек – надає можливість писати і фронт-енд, і бек-енд на JavaScript. Це знижує поріг входу і спрощує взаємодію між частинами проєкту.

Висока продуктивність – завдяки ефективному движку V8 та неблокуючій I/O модель ефективна для обробки тисяч запитів одночасно.

npm (Node Package Manager) – найбільший менеджер пакетів із понад мільйоном бібліотек, що допомагає швидко розширювати можливості Node.js.

Гнучкість – ідеально підходить для створення мікросервісів, API, реального часу додатків (чат, WebSocket, стрімінг).

Недоліки та виклики:

Однопотокова модель – підходить для задач з високою кількістю запитів, але неефективна для CPU-інтенсивних операцій (наприклад, обробка складних алгоритмів або великих обчислень).

Callback Hell – велика кількість вкладених колбеків ускладнює підтримку коду. Це значно покращилось завдяки async/await та Promises, але все ще потребує правильного проєктування.

Динамічна типізація – через JavaScript складніше підтримувати великі проєкти. Це вирішується використанням TypeScript, який додає строгі типи та підвищує стабільність.

Високе споживання пам’яті – порівняно з деякими іншими бекенд-технологіями, Node.js може використовувати більше ресурсів.

Швидкий старт – легкий для новачків та має активну спільноту.

Де Node.js застосовувати найкраще: Чати, стрімінгові сервіси – завдяки WebSocket та подієво-орієнтованій моделі. API та мікросервіси – доволі легка інтеграція з іншими сервісами та масштабованість. Обробка масивних обсягів запитів – хмарні сервіси, сервери додатків. DevOps та автоматизація – багато інструментів для автоматизації та роботи з серверами.

.NET (C#).

.NET – це потужна платформа від Microsoft, яка дозволяє розробляти комп’ютерні, мобільні, веб та хмарні додатки. Вона використовує C# як основну мову програмування, що надає статичну типізацію, високу безпеку та ефективність.

Архітектура та основні особливості:

Движок CLR (Common Language Runtime) – забезпечує автоматичне управління пам’яттю та виконання коду, що покращує стабільність та продуктивність.

Гнучкість платформи – можна створювати додатки для Windows, Linux, macOS, використовуючи .NET Core або .NET 7/8.

Підтримка мов – крім C#, підтримує F#, VB.NET та інші.

ASP.NET Core – фреймворк для створення веб-додатків та API, що працює швидко та ефективно.

XAML та Blazor – для створення UI (Blazor дозволяє писати веб- фронтенд на C#).

Переваги .NET та C#:

Статична типізація – знижує ризик помилок, код більш надійний.

Безпека – завдяки вбудованим механізмам, як Garbage Collection та Code Access Security.

Потужна IDE – Visual Studio забезпечує зручне середовище розробки, автодоповнення та інтеграцію з Azure.

Підтримка корпоративних рішень – широко використовується в бізнесі та великих проектах.

Гнучкість – можна створювати десктопні, мобільні, веб, cloud, IoT додатки.

Продуктивність – ефективні асинхронні операції (async/await) та оптимізований код.

Недоліки та виклики:

Менше бібліотек у порівнянні з npm – хоч екосистема багата, вона менша за JavaScript.

Комерційні інструменти – Visual Studio у повній версії може коштувати грошей, так само як Azure.

Крива навчання – C# складніший для новачків через строгу типізацію та ООП принципи.

Платформоорієнтованість – хоча .NET став кросплатформним, Windows залишається основною платформою.

Де .NET працює найкраще: Корпоративні системи – банківські сервіси, CRM, ERP-системи. Web-додатки – ASP.NET Core дозволяє швидко створювати API та сайти. Cloud-рішення – інтеграція з Azure, висока масштабованість. GameDev – використовується у Unity для створення ігор. Десктопні програми – .NET MAUI, WPF для Windows-додатків.

Python (Django / Flask).

Python – одна з найпопулярніших мов програмування завдяки простому синтаксису, багатій екосистемі та широкому застосуванню від веб-розробки до штучного інтелекту.

Два найпопулярніші фреймворки для веб-розробки – Django та Flask – мають різні підходи: Django – "батарейки включені", повнофункціональний фреймворк із багатьма вбудованими можливостями. Flask – мінімалістичний фреймворк, що дає більше свободи у виборі архітектури.

Основні переваги Python для веб-розробки:

Простий синтаксис – легкий для навчання та швидкого прототипування.

Швидкий старт – мінімум конфігурацій, можливість швидко розробляти MVP (мінімально життєздатний продукт).

Велика спільнота – багато документації, готових рішень та підтримка від розробників.

Масштабованість – хоча Python не найшвидша мова, він чудово підходить для мікросервісної архітектури.

Широке застосування – не лише веб-розробка, а й data science, AI, машинне навчання.

Багато фреймворків – Django для складних проєктів, Flask для гнучких рішень.

Недоліки Python та його фреймворків:

Менша продуктивність у порівнянні з Node.js або .NET – Python не завжди ефективний для великих навантажень.

Динамічна типізація – легше зробити помилку, ніж у C# або TypeScript, вирішується Type Hints та Pydantic.

Складність у масштабуванні – через GIL (Global Interpreter Lock) Python обмежений в багатопотоковості.

Менше можливостей для high-performance – хоча існують FastAPI, Nginx та Gunicorn, для швидкісних серверів краще розглянути Go або .NET.

Java (Spring Boot).

Java – одна з найстабільніших мов програмування, яка чудово підходить для великих enterprise-систем завдяки об'єктно-орієнтованому підходу, надійності та безпеці.

Spring Boot – це спрощений фреймворк для швидкого створення веб-додатків та мікросервісів, який зменшує складність конфігурації у порівнянні зі звичайним Spring Framework.

Основні переваги Java:

Стабільність та масштабованість – використовується у великих компаніях, таких як банки, телеком, фінансові сервіси.

Висока безпека – строгий контроль пам’яті, exception-handling, вбудовані механізми шифрування.

Кросплатформність – можна запускати на будь-якій ОС завдяки JVM (Java Virtual Machine).

Широкий набір інструментів – багато фреймворків (Spring, Hibernate, Micronaut).

Активна спільнота – велика підтримка розробників, регулярні оновлення мови.

Недоліки Java та Spring Boot:

Більший "шумний" синтаксис – більше коду для реалізації порівняно з Python або JavaScript.

Важчий стек – потрібне розуміння JVM, багатопоточності, колекцій, що ускладнює старт для новачків.

Більше конфігурацій – хоча Spring Boot спрощує процес, все ж налаштування можуть бути складнішими, ніж у Flask або Express.js.

Витрати ресурсів – Java потребує більше оперативної пам’яті та CPU у порівнянні з легкими фреймворками.

Результати проведеного аналізу та порівняння можна переглянути 1 таблиці 2.1.

Таблиця 2.1 – Порівняння платформ для виконання

Характеристика	**Node.js**	**.NET(C#)**	**Python (Django/Flask)**	**Java (Spring Boot)**
Продуктивність	Висока	Висока	Середня	Висока
Простота	Легка	Середня	Легка	Складна
Навчання	Швидке	Довге	Швидке	Довге
Масштабованість	Хороша	Відмінна	Можлива	Відмінна
CPU-інтенсивні задачі	Погано	Добре	Погано	Добре
Community/Бібліотеки	Дуже багато	Середньо	Великий обсяг	Великий обсяг

Отже, внаслідок порівняння платформ для виконання застосунків, було прийнято рішення обрати для розробки платформу Node.js через її високу продуктивність, простоту в використанні, швидке навчання та доступ до великої кількості бібліотек.

Node.js – це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Вона дає змогу виконувати JavaScript-скрипти на сервері та надсилати результати їх виконання користувачеві. Node.js перетворила JavaScript з мови, яка раніше використовувалась переважно в браузері, на загальноприйняту мову програмування з великою спільнотою розробників.

Основні властивості Node.js включають:

- асинхронна однопотокова модель виконання запитів, що дозволяє ефективно обробляти багатопотокові запити;
- неблокуючий ввід/вивід, що дозволяє ефективно працювати з мережевими операціями та операціями вводу-виводу;

- система модулів CommonJS, яка дозволяє організовувати код у модулі та повторно використовувати його;
- використання рушія JavaScript Google V8, що забезпечує швидке виконання коду JavaScript.

Для керування модулями використовується пакетний менеджер npm (node package manager).

Платформа Node.js призначена для виконання високопродуктивних мережевих застосунків, написаних мовою програмування JavaScript. Платформа окрім роботи із серверними скриптами для веб-запитів, також використовується для створення клієнтських та серверних програм.

В платформі використовується розроблений компанією Google рушій V8.

Для забезпечення обробки великої кількості паралельних запитів у Node.js використовується асинхронна модель запуску коду, заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback). Як способи мультиплексування з'єднань підтримується epoll, kqueue, /dev/poll і select. Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу нитей (thread pool) задіяна бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Всі системні виклики, що спричиняють блокування, виконуються всередині пулу потоків і потім, як і обробники сигналів, передають результат своєї роботи назад через неіменовані канали (pipe).

За своєю суттю Node.js схожий на фреймворки Perl AnyEvent, Ruby Event Machine і Python Twisted, але цикл обробки подій (event loop) у Node.js прихований від розробника і нагадує обробку подій у веб застосунку, що працює в браузері. При написанні програм для Node.js необхідно враховувати специфіку подієво-орієнтованого програмування, наприклад, замість виконання запиту з очікуванням завершення роботи і наступною обробкою результатів, в Node.js використовує принцип асинхронного виконання.

При цьому управління миттєво перейде до коду який слідує після виклику функції db.query, а результат запиту буде оброблений як тільки будуть оброблені

дані. Жодна функція в Node.js не повинна безпосередньо виконувати (блокуючі) операції вводу/виводу — для отримання даних з диска, від іншого процесу або з мережі потрібна установка callback-обробника.

Для розширення функціональності застосунків на базі Node.js підготовлена велика колекція модулів, в якій можна знайти реалізацію HTTP, SMTP, XMPP, DNS, FTP, IMAP, POP3 серверів і клієнтів, модулі для інтеграції з різними вебфреймворками, обробники WebSocket і AJAX, конектори до СКБД (MySQL, PostgreSQL, SQLite, MongoDB), шаблонізатори, CSS-рушії, реалізації криптоалгоритмів і систем авторизації (наприклад, OAuth), XML-парсери.

Стандартна поставка node.js включає в себе кількадесят модулів, у яких реалізовані типові операції для взаємодії з операційною системою, файловою системою, мережею і протоколами, утиліти для обробки даних. Крім того є доступними безліч модулів від незалежних розробників, програмісти можуть отримати їх з відкритих репозиторіїв і використовувати у своїх проектах. [4].

2.3 Вибір середовища для оптимізації

Проведемо аналіз середовищ розробки та оберемо найоптимальніше для даної роботи.

Atom.

Atom був одним із найпопулярніших відкритих текстових редакторів, створених GitHub і пізніше підтримуваних Microsoft. Він вирізнявся гнучкістю, розширюваністю та простотою налаштування, але, на жаль, офіційна підтримка припинена у грудні 2022 року.

Основні можливості Atom:

Висока розширюваність – понад 8000 плагінів (packages), які дозволяли додавати нові функції, покращувати навігацію та інтегрувати редактор з іншими інструментами.

Вбудований менеджер пакетів (apm) – спрощував пошук і встановлення розширень без додаткових налаштувань.

Гнучке налаштування – можна змінювати інтерфейс за допомогою HTML, CSS та JavaScript.

GitHub-інтеграція – зручні функції для роботи з репозиторіями, які значно полегшували життя розробникам (до припинення підтримки).

Кросплатформність – працював на Windows, macOS, Linux без обмежень.

Підходить для фронтенд-розробки – завдяки легкості та можливості встановлення плагінів для HTML, CSS, JS.

Недоліки та виклики

Повільна продуктивність на великих проектах – через використання Electron, Atom міг бути ресурсозатратним.

Офіційна підтримка припинена – немає нових оновлень чи виправлення вразливостей, що робить його менш безпечним.

Не повноцінна IDE – хоч Atom і був чудовим редактором, він не мав засобів для рефакторингу, глибокого налагодження чи автоматичного аналізу коду, як у Visual Studio або IntelliJ IDEA.

Visual Studio Code.

Visual Studio Code (VS Code) – легкий, швидкий та розширюваний текстовий редактор, який став стандартом для багатьох розробників. Він підтримує широкий спектр мов, має потужний набір інструментів і чудово інтегрується з Git та хмарними середовищами.

Основні переваги VS Code:

Висока продуктивність – працює швидко навіть на великих проєктах.

Легка вага – значно легший за повноцінні IDE, такі як IntelliJ IDEA або Visual Studio.

Велика бібліотека розширень – понад 30 000 плагінів, які розширюють функціональність редактора.

Інтеграція з Git – вбудовані інструменти для роботи з репозиторіями, комітами та злиттям змін.

IntelliSense та автодоповнення – покращує продуктивність завдяки підсвічуванню, автозавершенню коду та підказкам.

Підтримка TypeScript, Python, C++, Docker – редактор чудово адаптується до різних мов і технологій.

Вбудований дебагер – дозволяє налагоджувати код без встановлення додаткових інструментів.

Кросплатформність – працює на Windows, macOS, Linux.

Недоліки VS Code:

Деякі функції потребують встановлення плагінів – наприклад, розширене налагодження чи робота з базами даних.

Частковий збір телеметрії – редактор надсилає анонімні дані про використання (можна вимкнути в налаштуваннях).

Менше інструментів для рефакторингу у порівнянні з повноцінними IDE – хоча є плагіни, вони не завжди настільки глибокі, як у JetBrains.

JetBrains IntelliJ IDEA / WebStorm / PyCharm.

JetBrains створює потужні кросплатформні IDE, які відзначаються глибоким автодоповненням, рефакторингом та вбудованими інструментами для розробки. Вони особливо популярні серед професійних розробників, оскільки мають розширені можливості порівняно з редакторами на кшталт VS Code.

Основні переваги JetBrains IDE:

Потужне автодоповнення та рефакторинг – аналіз коду, розумне підказування, швидке переформування структури.

Вбудований налагоджувач – ефективний дебагер для Java, Python, JavaScript, TypeScript та інших мов.

Тестування та профайлінг – глибокий аналіз продуктивності коду та перевірка функціональності.

Сильна підтримка фреймворків – працює чудово з Spring, Django, React, Angular, Vue, FastAPI тощо.

Інтеграція з Docker, Git, CI/CD – полегшує DevOps-процеси та автоматизацію.

Розширене управління проєктами – підтримка Gradle, Maven, зручне навігаційне дерево, швидкі переходи.

Розширюваність – сотні плагінів для доповнення функціоналу.

Недоліки JetBrains IDE:

Комерційні версії платні – хоча є безкоштовні версії (Community), вони мають обмежений функціонал.

Важчі й повільніші за VS Code – через велику кількість вбудованих функцій займають більше ресурсів.

Високі системні вимоги – потребують більше RAM та CPU, особливо при роботі з великими проєктами.

Sublime Text.

Sublime Text – один із найшвидших редакторів коду, який відзначається мінімалістичним дизайном, високою продуктивністю та простотою. Його використовують розробники, яким потрібний легкий, але потужний інструмент для редагування коду без зайвих функцій.

Основні переваги Sublime Text:

Надзвичайно швидкий – завдяки оптимізованій архітектурі працює плавно навіть на великих файлах.

Легкий – займає мало ресурсів, тому добре працює навіть на слабких комп'ютерах.

Мінімалістичний інтерфейс – немає зайвих елементів, зосереджений на редагуванні коду.

Широка підтримка мов – працює з Python, JavaScript, HTML, CSS, C++, Go та багатьма іншими мовами.

Package Control – потужна система розширень, яка дозволяє додавати нові функції.

Функція “Goto Anything” – швидкий перехід між файлами та кодовими фрагментами.

Розширене редагування – підтримка мульти-курсора, автодоповнення та форматування.

Кросплатформність – працює на Windows, macOS, Linux.

Недоліки Sublime Text:

Платний (безкоштовна версія має нагадування) – ліцензія коштує 99$ для персонального використання.

Менш розширюваний, ніж VS Code – хоча є розширення, немає такої великої бібліотеки як у Microsoft VS Code.

Не повноцінна IDE – немає вбудованого дебагера, рефакторингу, інтеграції з CI/CD, тому не підходить для складних проектів.

Всі результати порівняння середовищ наведені у таблиці 2.2.

Таблиця 2.2 – Порівняльна таблиця середовищ розробки

Характеристика	**Atom**	**VS Code**	**JetBrains IDE**	**Sublime Text**
Продуктивність	Низька	Висока	Висока	Дуже висока
Розширюваність	Висока	Висока	Висока	Середня
Підтримка	Припинена	Активна	Активна	Активна
Підходить для новачків	Так	Так	Не зовсім	Так
Повноцінна IDE	Ні	Частково	Так	Ні
Вартість	Безкоштовно	Безкоштовно	Частково платна	Платна

Отже, Atom — хороший редактор свого часу, але зараз застарілий і не рекомендується для нових проєктів. JetBrains IDE — ідеальний вибір для професійної розробки в конкретній мові (Java, Python тощо). Sublime Text — блискавичний редактор, проте без розширеного функціоналу IDE. VS Code — найкращий баланс між швидкістю, функціональністю та підтримкою.

Тому для розробки проєкту було вибрано середовище VS Code.

Visual Studio Code, який також зазвичай називають VS Code — це редактор початкового коду, створений Microsoft із Electron Framework для Windows, Linux і macOS. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш, параметри та встановлювати розширення, які додають функціональність.

В опитуванні розробників Stack Overflow 2022 серед 71 010 респондентів Visual Studio Code назвали найпопулярнішим інструментом середовища розробника, при цьому 74,48 % повідомили, що вони ним користуються.

Visual Studio Code — це редактор вихідного коду, який можна використовувати з різними мовами програмування, включаючи C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust. Він базується на структурі Electron, яка використовується для розробки вебдодатків Node.js, які працюють на механізмі компонування Blink. Visual Studio Code використовує той самий компонент редактора (під кодовою назвою «Monaco»), який використовується в Azure DevOps (раніше називався Visual Studio Online і Visual Studio Team Services).

З коробки Visual Studio Code містить базову підтримку для більшості поширених мов програмування. Ця базова підтримка включає підсвічування синтаксису, зіставлення дужок, згортання коду та настроювані фрагменти. Visual Studio Code також постачається з IntelliSense для JavaScript, TypeScript, JSON, CSS і HTML, а також підтримує налагодження Node.js. Підтримка додаткових мов може бути забезпечена безоплатними розширеннями на VS Code Marketplace.

Замість системи проєктів вона дозволяє користувачам відкривати один або кілька каталогів, які потім можна зберегти в робочих областях для подальшого використання. Це дозволяє йому працювати як мовно-агностичний редактор коду для будь-якої мови. Він підтримує багато мов програмування та набір функцій, який відрізняється для кожної мови. Небажані файли та папки можна виключити з дерева проєкту за допомогою налаштувань. Багато функцій Visual Studio Code не доступні через меню чи інтерфейс користувача, але доступ до них можна отримати через палітру команд.

Код Visual Studio можна розширити за допомогою розширень, доступних через центральне сховище. Це включає доповнення до редактора та підтримку мови. Примітною функцією є можливість створювати розширення, які додають підтримку нових мов, тем, налагоджувачів, налагоджувачів подорожей у часі,

виконують статичний аналіз коду та додають лінтери коду за допомогою протоколу Language Server.

Керування джерелом є вбудованою функцією Visual Studio Code. Він має спеціальну вкладку всередині панелі меню, де користувачі можуть отримати доступ до налаштувань керування версіями та переглянути зміни, внесені до поточного проєкту. Щоб використовувати цю функцію, Visual Studio Code має бути зв’язано з будь-якою підтримуваною системою керування версіями (Git, Apache Subversion, Perforce тощо). Це дозволяє користувачам створювати репозиторії, а також робити запити push і pull безпосередньо з програми Visual Studio Code.

Visual Studio Code містить кілька розширень для FTP, що дозволяє використовувати програмне забезпечення як безплатну альтернативу для веброзробки. Код можна синхронізувати між редактором і сервером без завантаження додаткового програмного забезпечення.

Код Visual Studio дозволяє користувачам встановлювати кодову сторінку, на якій буде збережено активний документ, символ нового рядка та мову програмування активного документа. Це дозволяє використовувати його на будь-якій платформі, у будь-якій локальній мережі та для будь-якої заданої мови програмування [promotional language].

Visual Studio Code збирає дані про використання та надсилає їх до Microsoft, хоча це можна вимкнути. Через відкритий вихідний код програми, код телеметрії доступний для громадськості, яка може бачити, що саме збирається[5].

Розглянемо деякі просунуті JavaScript функції, які підтримує Visual Studio Code. Використовуючи мовний сервіс TypeScript, VS Code може забезпечити розумне доповнення (IntelliSense), а також перевірку типів для JavaScript.

JavaScript IntelliSense від Visual Studio Code забезпечує інтелектуальне доповнення коду, інформацію про параметри, пошук посилань та багато інших розширених функцій мови. JavaScript IntelliSense працює на основі служби мов JavaScript, розробленої командою TypeScript. У той час як IntelliSense повинен

працювати лише для більшості проектів JavaScript без будь-якої конфігурації, можна зробити IntelliSense ще кориснішим за допомогою JSDoc або налаштувавши проект jsconfig.json.

2.4 Вибір програми для роботи зі скриптом

Проведемо порівняльний аналіз доступних програм.

Git Bash.

Git Bash – це термінал для Windows, який дозволяє використовувати Bash-команди, як у Linux. Він поставляється разом із Git for Windows і є чудовим рішенням для розробників, які працюють з Git у Windows-середовищі.

Основні переваги Git Bash:

Linux-подібні команди – підтримує ls, cat, grep, ssh, chmod, що робить роботу зі скриптами схожою на Linux-середовище.

Ідеально для Git – містить повний набір команд Git, що дозволяє працювати з репозиторіями та керувати версіями коду.

Проста установка – входить до складу Git for Windows, тому налаштування мінімальне.

Легкий інтерфейс – не потребує додаткових ресурсів, працює швидко навіть на старих машинах.

Інтеграція з GitHub – дозволяє легко взаємодіяти з хостингом репозиторіїв.

Можливість виконання Bash-скриптів – працює майже як Linux, що зручно для автоматизації.

Підходить для розробників, які використовують Git та Bash-команди в Windows-середовищі.

Недоліки та обмеження Git Bash:

Менше можливостей, ніж повноцінні емулятори Linux – хоча підтримує більшість команд Bash, це не справжній Linux-середовище.

Не підтримує вкладки – у порівнянні з Windows Terminal або Cmder, у Git Bash немає мультивкладок.

Відсутній кастомний UI – немає кольорового оформлення або розширеної конфігурації, як у PowerShell чи iTerm.

Лише для Windows – немає версій для macOS або Linux (хоча Linux уже має Bash за замовчуванням).

Windows Terminal.

Windows Terminal – це потужний, сучасний інтерфейс для командного рядка у Windows, який підтримує PowerShell, CMD, Git Bash та WSL в одному вікні. Його головна особливість – розширена кастомізація, вкладки та GPU-рендеринг, що робить роботу із терміналом зручнішою.

Основні переваги Windows Terminal:

Об'єднує всі shell-и – можна працювати з PowerShell, CMD, Git Bash, WSL без відкриття окремих терміналів.

Теми та кастомізація – змінюваний шрифт, колірна схема, прозорість, а також підтримка Unicode та емодзі.

Вкладки та панелі – можливість відкривати кілька терміналів у одному вікні та розділяти їх у горизонтальних або вертикальних панелях.

Продуктивність – завдяки GPU-рендерингу працює швидше та плавніше.

Кросплатформний формат – працює на Windows 10 та 11, а також підтримує WSL для емуляції Linux-середовища.

Швидкий запуск – працює значно швидше, ніж класичний CMD чи PowerShell.

JSON-конфігурації – дозволяють тонко налаштовувати вигляд і поведінку терміналу.

Недоліки та виклики Windows Terminal:

Налаштування через settings.json – кастомізація потребує редагування JSON-файлу, що може бути складним для новачків.

Не є shell сам по собі – це інтерфейс над іншими shell, а не самостійний командний інтерпретатор.

Вимагає Windows 10 або новіше – недоступний на старих версіях Windows.

PowerShell.

PowerShell – це розширений командний інтерпретатор, який прийшов на зміну CMD, надаючи більше можливостей для автоматизації, роботи з .NET та управління системою.

Основні переваги PowerShell:

Підтримка об'єктів .NET – дозволяє працювати з повноцінними об'єктами, а не просто текстовими рядками, як у CMD.

Cmdlets – покращені команди – кожна команда (Get-Process, Get-Service, Stop-Computer) працює за принципом об'єктної моделі.

Скрипти .ps1 – створення повноцінних автоматизованих скриптів для адміністрування Windows-систем.

Глибока інтеграція з Windows – доступ до служб, файлової системи, мережі та інших системних ресурсів.

Робота з JSON, XML, REST API – дозволяє легко отримувати та обробляти дані у складних форматах.

Розширена система pipeline – передача об'єктів між командами для більш ефективної роботи.

Недоліки та обмеження PowerShell:

Інша логіка синтаксису (не POSIX-стандарт) – команди працюють інакше, ніж у Bash, що може бути незвичним.

Менше крос-платформеного досвіду – хоча PowerShell Core доступний для Linux та macOS, він не такий популярний, як Bash.

Не завжди сумісний з Bash-скриптами – стандартні команди Linux можуть не працювати без змін.

CMD (Command Prompt).

Опис: Класичний термінал Windows, дуже базовий.

Недоліки: Застарілий, без підтримки скриптів bash або git команд.

WSL (Windows Subsystem for Linux).

WSL дозволяє запускати повноцінне Linux-середовище без необхідності встановлення віртуальної машини чи використання окремого комп’ютера. Це

чудове рішення для розробників, які хочуть поєднувати можливості Windows та Linux в одному середовищі.

Основні переваги WSL:

Запуск справжньої Linux-оболонки – підтримка Ubuntu, Debian, Fedora, openSUSE та інших дистрибутивів.

Linux-інструменти у Windows – можна використовувати apt, yum, ssh, cron, grep, awk, sed, systemctl та інші Linux-команди.

Інтеграція з Windows Terminal – працює у вкладках поряд із CMD, PowerShell, Git Bash.

Висока продуктивність у порівнянні з класичними віртуальними машинами – WSL працює швидше, ніж VirtualBox або VMware, оскільки не вимагає повної емуляції.

Доступ до Windows файлової системи – можна безпосередньо працювати з файлами Windows у Linux (/mnt/c/Users/...).

Docker та контейнери – WSL 2 дозволяє запускати Docker-контейнери безпосередньо у Windows, що значно спрощує роботу DevOps.

Кросплатформна розробка – зручно для тестування програм у Linux, навіть якщо основне середовище – Windows.

Ідеально підходить для розробників, DevOps, системних адміністраторів.

Недоліки та виклики WSL:

Потрібне налаштування – початкове налаштування WSL 2 може вимагати встановлення додаткових компонентів (Hyper-V).

Шар віртуалізації – хоча продуктивність висока, WSL 2 працює через віртуалізацію, що може використовувати більше оперативної пам’яті.

Іноді складно інтегрувати з Windows FS – хоча доступ до /mnt/c/ є, деякі програми можуть некоректно працювати з Windows API.

Відсутність Linux GUI-програм за замовчуванням – потрібно додатково налаштовувати підтримку X-сервера або Wayland для запуску Linux GUI додатків.

Результати порівняльного аналізу програм наведено в таблиці 2.3.

Таблиця 2.3 – Порівняльна таблиця програм

Функціональність	**Git Bash**	**PowerShell**	**CMD**	**Windows Terminal**	**WSL (Ubuntu, etc.)**
Linux-команди	частково	ні	ні	ні	повністю
Вкладки	ні	ні	ні	так	Так, через WT
Bash-скрипти	так	ні	ні	ні	так
Git-інтеграція	Так, відразу	Можлива через Git	Можлива	Можлива	Можлива
Візуальна кастомізація	ні	ні	ні	так	можлива
Платформа	Windows	Windows/Linux	Windows	Windows	Windows

Отже, Git Bash — ідеальний для швидких Git-операцій, Bash-скриптів на Windows. Windows Terminal — найкраще середовище для тих, хто хоче поєднати WSL, PowerShell, Git Bash в одному. WSL — для розробників, які хочуть повноцінний Linux-досвід у Windows. PowerShell — для адміністраторів Windows і .NET-розробників. CMD — лише для старих застосунків або крайньої потреби.

Для роботи зі скриптом буде використано програму Git Bash.

Git - це набір утиліт командного рядка, які використовуються для управління версіями програмного коду. В основному, Git використовується через командний рядок в середовищі Unix-подібних операційних систем, таких як Linux і macOS. На цих операційних системах доступні вбудовані термінали командного рядка Unix, що дозволяє легко використовувати Git.

Однак, на платформі Microsoft Windows, яка має відмінне від Unix термінальне середовище, Git поширено використовувати разом з графічними інтерфейсами вищого рівня. Ці графічні інтерфейси намагаються спростити використання Git і приховати його основні команди. Вони можуть бути

корисними для початківців, які швидко хочуть почати використовувати Git у своїх проектах.

Проте, коли потрібно співпрацювати з іншими членами команди або коли вимоги до управління версіями стають складнішими, важливо мати розуміння базових команд Git. Для цього можна використовувати інструменти командного рядка, наприклад Git Bash, який забезпечує термінальний досвід для використання Git на платформі Windows.

Таким чином, Git можна використовувати як через командний рядок в середовищі Unix, так і через графічні інтерфейси на різних операційних системах. Вибір інструменту залежить від потреб та рівня зручності для користувача. [6].

Git - це система контролю версій, яка дозволяє відстежувати зміни в колекції файлів за допомогою фіксацій і повертатися до попередніх станів проекту. Git можна використовувати як локально на своєму комп'ютері, так і на онлайн-хостінгових платформах, таких як GitHub або Bitbucket. Однак, в основному Git використовується через утиліту командного рядка в стилі Unix.

Git є надзвичайно популярною системою контролю версій завдяки своїм потужним функціям. Він дозволяє створювати гілки, що дозволяють створювати незалежні версії кодової бази, які потім можна об'єднати. Це дозволяє програмістам легко співпрацювати та обмінюватися своїми змінами у вихідному коді.

Git є відкритим вихідним кодом, безкоштовним у використанні та досить простим для вивчення. Він надає можливості для ефективного керування версіями проектів будь-якої складності.

Bash - це процес. Це програма, як і будь -яка інша. Він читає від STDIN, інтерпретує те, що він знаходить, і виконує дію у відповідь. Це не відрізняється від, наприклад, перекладача Python.

Bash - це конкретний вид програми, це оболонка. Він призначений для інтерпретації вашого введення як назви та параметрів інших процесів для запуску. Він також має купу спеціального синтаксису для виконання

примітивного контрольного потоку. Існує багато інших надбудов (CSH, KSH, Fish тощо). Платформа також надає можливість створювати власні надбудови. Bash розроблений так, щоб бути схожим на старшу оболонку Unix, під назвою Bourne Shell (вона ж SH).

BASH викликає функції інтерфейсу операційних систем для створення та запуску цих процесів. Багато програм роблять це, тому BASH в цьому плані не є дуже особливий, окрім факту, що він створює багато інших процесів.

BASH розроблений для розташування в операційній системі GNU/Linux, оскільки вона використовує функції інтерфейсу ОС, які існують лише на Linux, наприклад, Fork (). ОС Linux також зазвичай налаштована для запуску Bash під час входу через текстовий інтерфейс, наприклад, через SSH або віртуальний термінал, але змінити оболонку входу можна на зручну для розробника.

Git Bash є інструментом, спеціально призначеним для користувачів Microsoft Windows. Він поєднує в собі Git та оболонку командного рядка Bash, надаючи користувачам Windows доступ до функціональності Git і командного середовища, яке є родинним для користувачів macOS і Linux.

Отже, Git - це система контролю версій, а Bash - оболонка командного рядка. Git Bash є спеціальним інструментом для користувачів Windows, який поєднує в собі обидва цих компонента. [7].

2.5 Легальність парсингу

В сьогоднішньому світі дані є цінним ресурсом, а їхнє збирання — важливою складовою частиною бізнесу, науки та державного управління. Одним із інструментів, який дозволяє автоматизовано збирати великі обсяги даних з інтернет-ресурсів, є парсинг. Однак виникає питання: наскільки законним є цей процес в Україні?

Парсинг як явище не має прямого регулювання в українському законодавстві. Але до нього можна застосовувати положення таких нормативних актів:

- Закон України "Про захист персональних даних";
- Закон України "Про інформацію";
- Цивільний кодекс України;
- Закон України "Про авторське право і суміжні права";
- Закон України "Про доступ до публічної інформації".

Загалом, якщо дані є відкритими, доступними для будь-якого користувача, і не містять персональної або авторсько захищеної інформації — їх збір не порушує закон.

Парсинг не є кримінальним правопорушенням сам по собі. Він може бути реалізований шляхом простого HTTP-запиту без втручання в систему. Проте при парсингу можуть порушуватись умови користувацької угоди сайту, що має юридичне значення. Також деякі ресурси мають обмеження через файл robots.txt або API-умови, які варто враховувати.

Парсинг, який супроводжується DDoS-атаками, обходом авторизації або іншими втручаннями в систему — уже є незаконним.

Україна підтримує політику відкритості даних. Законодавчо це закріплено в Постанові КМУ №835 від 21.10.2015 року "Про затвердження Положення про набори даних, що підлягають оприлюдненню у формі відкритих даних" та Законі України "Про доступ до публічної інформації".

Дані, що оприлюднені на data.gov.ua та інших офіційних порталах у форматі open data, дозволено використовувати без обмежень. Відповідно, їх парсинг є цілком законним.

Інформація сама по собі не захищається авторським правом, однак її структура, добір, розміщення можуть бути об’єктами охорони. Тому парсинг сайтів, що містять бази даних або структуровану інформацію, може становити порушення авторських прав у випадку копіювання суттєвої частини бази даних.

Відповідно до Цивільного кодексу України (ст. 433), бази даних визнаються об’єктом інтелектуальної власності.

Якщо парсинг передбачає обробку персональних даних (наприклад, імен, адрес, телефонів), то застосовується Закон "Про захист персональних даних". Такі дані не можуть бути оброблені без згоди суб’єкта.

Крім того, для компаній, які працюють з громадянами ЄС, діє GDPR — Загальний регламент про захист даних. Порушення правил обробки персональних даних тягне за собою серйозні санкції.

В Україні мало судових справ, прямо пов’язаних із парсингом. Проте існують прецеденти, коли компанії скаржились на несанкціоноване копіювання даних, зокрема оголошень з маркетплейсів, каталогів.

На міжнародному рівні відомі кейси:

HiQ Labs v. LinkedIn (США) — суд визнав, що публічна інформація може бути спарсена, якщо вона не захищена технічними засобами.

Ryanair v. PR Aviation — Суд ЄС підтвердив, що умови сайту можуть забороняти автоматичний збір даних, навіть якщо інформація не захищена авторським правом.

У країнах ЄС, США, Канаді парсинг розглядається в контексті порушення авторських прав, конфіденційності, договорів користування. Ключовими є:

- чи були дані публічними;
- чи було порушено умови сайту;
- чи використовувалися персональні дані;
- чи завдано шкоди власнику ресурсу.

У деяких юрисдикціях парсинг є законним, якщо не порушено технічних або правових обмежень.

Парсинг в Україні не є забороненим автоматично. Якщо він:

- здійснюється щодо відкритих даних;
- не порушує умови використання ресурсу;
- не включає персональні або захищені дані;
- не втручається в роботу системи — він є легальним. [8][9]

2.6 Висновки до розділу 2

В другому розділі даної роботи проведено порівняльний аналіз найпопулярніших платформ для виконання коду. Беручи до уваги результати проведеного аналізу сформовано таблицю порівнянь та прийняте рішення використовувати Node.js як платформу для виконання коду.

Для вибору середовища розробки проведено порівняння відомих середовищ розробки, виділено їх переваги і недоліки, сформовано таблицю з результатами, на основі якої було обрано найкраще середовище для розробки.

Також проведено ознайомлення та аналіз доступних програм для роботи вже безпосередньо зі скриптом. Було сформовано порівняльну таблицю та обрано найкраще рішення.

Під час виконання роботи також було підняте питання легальності парсингу даних в Україні, тому було проведено дослідження, в результаті якого було з’ясовано, що чіткого законодавчого регулювання парсингу в Україні немає, але, щоб уникнути будь-яких претензій зі сторони власників компаній чи веб-ресурсів, потрібно поважати авторське право, а також користуватись тільки відкритими ресурсами та інформацією, яка є доступною у відкритому доступі.

3 РЕАЛІЗАЦІЯ СКРИПТА ДЛЯ ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ ПОТОКІВ

3.1 Формування алгоритму скрипта для пошуку

Для кращого розуміння алгоритму вигрузки даних для клієнтської бази відповідно до запиту замовника, розроблено блок-схему, зображену на рисунку 3.1. Процес відбору відбувається наступним чином:

a) Спеціаліст отримує інформацію від клієнта на формування бази даних в певній області чи з певним продуктом;

b) Беручи до уваги сегмент ринку, до якого відноситься даний запит, обираються ключові слова та перевіряються на актуальність в Google Maps;

c) Складається приблизний перелік цільових країн (наприклад, країни Шенгенської зони);

d) Запускається скрипт, з використанням сформованих ключових слів та обраних цільових країн;

e) Після завершення роботи, на основі отриманих результатів створюється таблиця;

f) Якщо витягнутих даних достатньо – зібрана база надсилається замовнику;

g) Якщо ж даних замало або база має малу конверсію (власники не піднімають слухавки, інтернет-сторінки не містять потрібні товари або категорії товарів) – визначаються нові ключові слова або обираються нові цільові країни (інколи обидва варіанти). Після цього знову запускається скрипт, формується база даних та перевіряється зібрані дані на повноту та задовільну конверсію

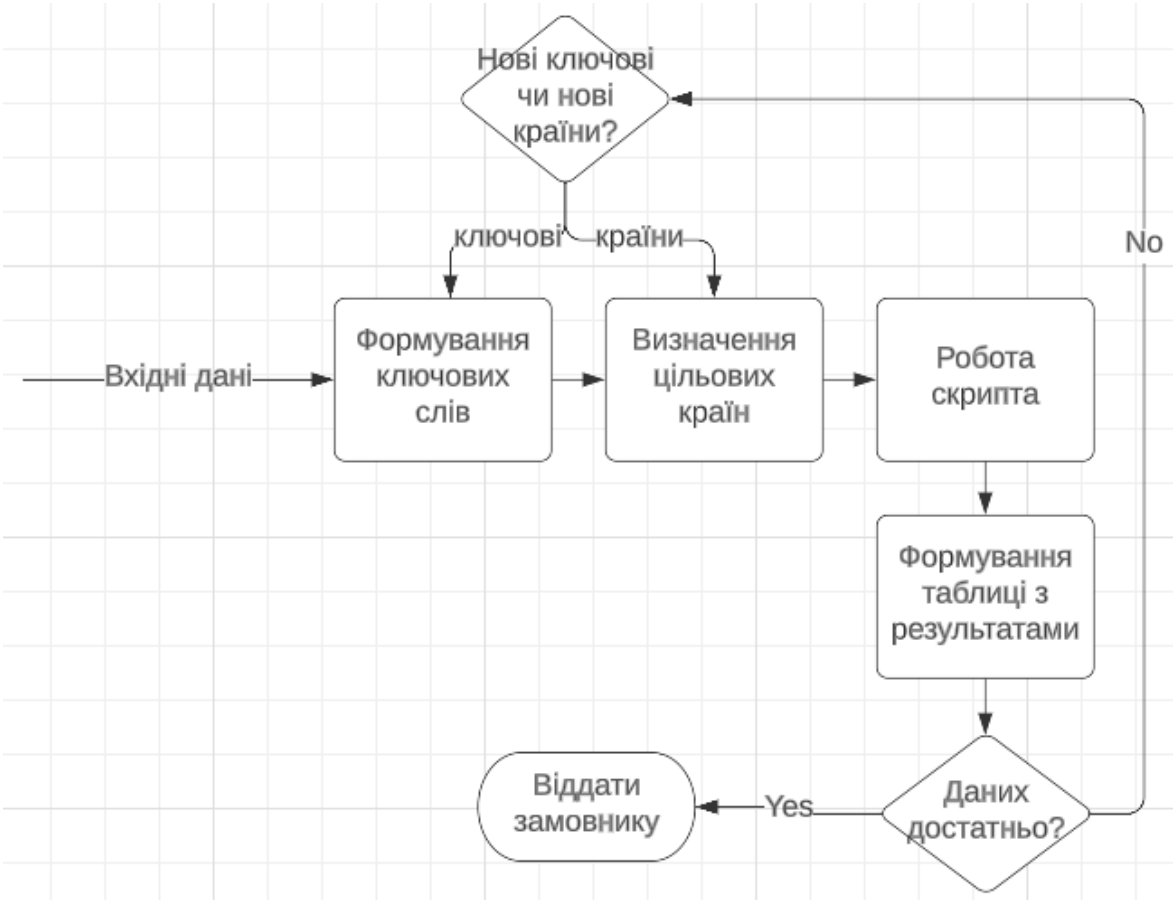


Рисунок 3.1 – Алгоритм формування цільової бази для клієнта

Щоб мати краще розуміння алгоритму роботи самого скрипта для парсингу розроблено блок-схему, подану на рисунку 3.2.

Основний алгоритм роботи скрипта складається з наступних кроків:

a) Початковим кроком є запуск консолі;

b) В консолі вводиться команда для запуску, цільова країна, ключові слова, а також кількість карток, які будуть опрацьовані за один запит (кількість карток не є обов’язковою);

c) Скрипт створює масив запитів з міст відповідної цільової країни та обраних ключових слів;

d) Запити з масиву почергово вводяться в пошуку в Google Maps;

e) Проводиться перевірка чи запит містить звичайні чи короткі картки;

f) Якщо видача на запит містить звичайні картки, їх записується в окремий файл, посилання з якого потім будуть опрацьовані після чого отримана інформація буде записана в кінцевому файлі;

g) Якщо запит містить короткі картки, дані з них одразу записуються в кінцевий файл[30].

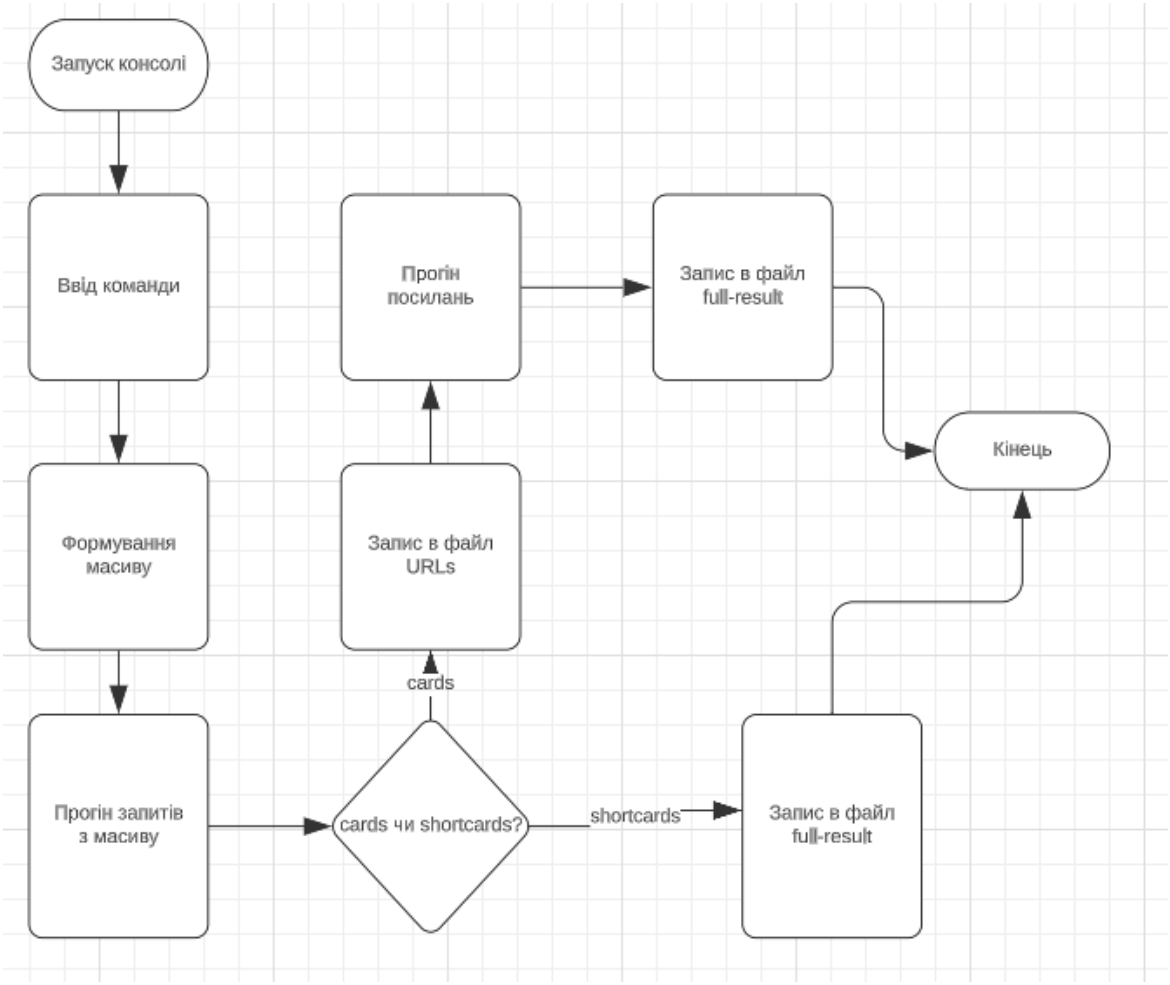


Рисунок 3.2 – Алгоритм роботи скрипта

3.2 Основний функціонал скрипта

Розробляємо скрипт, який буде реалізовувати пошук компаній по заданих ключових словах та країні. Тобто користувач вводить команду node parser.js, після цього в лапках вводить ключові слова, за якими буде проводитися пошук компаній, а потім буквений код країни, по якій буде проведено пошук. Дані формують масив запитів, які будуть введені в графі пошуку.

```
const queriesArray = places.makeQueriesArray(arguments)
log(queriesArray)
```

Реалізуємо перевірку даних на валідність (лістинг 3.1):

Отримання аргументів командного рядка:

```
const arguments = process.argv.slice(2);
```

process.argv — це масив, який містить аргументи, передані в програму через командний рядок.

Використовуючи slice(2), ми видаляємо перші два аргументи, що відповідають шляху до Node.js і виконуваного скрипта, залишаючи тільки те, що ввів користувач.

Перевірка кількості аргументів:

```
if (arguments.length == 0 || arguments.length > 3) {
  log.e('arguments empty or has a wrong format');
  exit();}
```

Перевіряється, чи кількість аргументів знаходиться в межах допустимого діапазону (1–3 аргументи).

Якщо аргументів немає або їх більше трьох, програма виводить повідомлення про помилку і завершується (exit()).

Перевірка валідності країни:

```
    if
(!places.countriesList.hasOwnProperty(arguments[1].toUpperCase()))
{
      log.e('you have entered nonexistent country shortname');
      exit();}
```

Аргумент arguments[1] має бути скороченням країни (наприклад, "US", "UA").

arguments[1].toUpperCase() конвертує текст у верхній регістр.

Функція hasOwnProperty перевіряє, чи існує передана країна в об'єкті places.countriesList.

Якщо країна не знайдена, виводиться помилка і програма завершується.

Перевірка третього аргументу:

```
if (arguments[2] && !Number.isInteger(+arguments[2])) {
  log.e('third argument is not a number');
  exit();}
```

Перевіряється наявність третього аргументу.

Якщо він існує, перевіряється, чи є він числом за допомогою Number.isInteger(+arguments[2]).

Якщо третій аргумент не є числом, виводиться повідомлення про помилку.

Визначення значення scrollCardsAmount

```
const scrollCardsAmount = arguments[2] || 40;
```

Змінна scrollCardsAmount встановлюється в значення третього аргументу, якщо він існує і є валідним.

Якщо третього аргументу немає, за замовчуванням використовується значення 40.

Лістинг 3.1 – Перевірка даних на валідність

```
const arguments = process.argv.slice(2)
if (arguments.length == 0 || arguments.length > 3) {
  log.e('arguments empty or has a wrong format');
  exit();
}
if
(!places.countriesList.hasOwnProperty(arguments[1].toUpperCase()))
{
  log.e('you have entered nonexistent country shortname');
  exit();
}

if (arguments[2] && !Number.isInteger(+arguments[2])) {
  log.e('third argument is not a number');
  exit();
}
const scrollCardsAmount = arguments[2] || 40;
```

Реалізуємо запуск скрипта (лістинг 3.2).

Ініціалізація браузера:

```
const browser = await puppeteer.launch({ headless: false });
const page = await browser.newPage();
```

```
page.setViewport({ width: 1280, height: 600 });
```

Puppeteer запускає новий браузер, налаштований на режим headless: false (видимий браузер).

Відкривається нова сторінка браузера, а її розмір встановлюється на ширину 1280px та висоту 600px.

Навігація до сторінки Google Maps:

```
await page.goto("https://www.google.com.ua/maps/?hl=en");
await page.waitForSelector('input#searchboxinput', { timeout: 5000 });
```

page.goto здійснює перехід на веб-сторінку Google Maps із вказаним параметром мови hl=en.

page.waitForSelector очікує появу елемента пошуку (input#searchboxinput) протягом 5 секунд, щоб гарантувати готовність сторінки для взаємодії.

Цикл обробки запитів:

```
for (const currentQuery of queriesArray) {
  try {
    await handleSingleQuery(page, currentQuery);
  } catch (error) {
    log.error(`throuble on ${currentQuery},\n`, error);
  }
  log(currentQuery);}
```

queriesArray містить список запитів для обробки.

У циклі для кожного запиту викликається функція handleSingleQuery, яка виконує обробку сторінки. У випадку помилки викликається catch, щоб зареєструвати помилку через log.error.

Закриття браузера:

```
log.debug('done all scrolling');
await browser.close();
```

Після завершення всіх операцій браузер закривається, а в лог додається запис про завершення скролінгу.

Обробка посилань:

```
    parseLinks(placesLinksIterator(parsedPlacesLinks));
    log.start(`lines   total:   ${parsedPlacesLinks.length}\nlines
done [%s]`, 1);
```

parseLinks обробляє зібрані посилання (parsedPlacesLinks) за допомогою ітератора.

В лог додається інформація про загальну кількість оброблених рядків.

Лістинг 3.2 – Запуск роботи скрипта

```
async function startParse() {
  const browser = await puppeteer.launch({ headless: false });
  const page = await browser.newPage();
  page.setViewport({ width: 1280, height: 600 });

  await page.goto("https://www.google.com.ua/maps/?hl=en");
  await     page.waitForSelector('input#searchboxinput',     {
timeout: 5000 })

  for (const currentQuery of queriesArray) {
    try {
      await handleSingleQuery(page, currentQuery);
    } catch (error) {
      log.error(`throuble on ${currentQuery},\n`, error)
    }
    log(currentQuery)
  }
  log.debug('done all scrolling');
  await browser.close();

  parseLinks(placesLinksIterator(parsedPlacesLinks))
  log.start(`lines  total:  ${parsedPlacesLinks.length}\nlines
done [%s]`, 1);}
```

Реалізуємо функцію, яка обробляє один запит для автоматизації роботи з веб-сторінкою Google Maps, витягаючи необхідні дані (наприклад, назву, активність, адресу, телефон і сайт)(Лістинг3.3 Додаток А).

```
async function handleSingleQuery(page, item)
```

Оголошується асинхронна функція handleSingleQuery, яка отримує два аргументи:

page — об'єкт Puppeteer для взаємодії з веб-сторінкою.

item — конкретний запит для виконання.

Обробка введення через клавіатуру:

```
await page.focus('input#searchboxinput');
```

Наводиться фокус на поле вводу input#searchboxinput (лістинг 3.4) на сторінці (рядок пошуку).

Лістинг 3.4 – Наведення фокусу на поле вводу

```
await page.keyboard.down('Control');
await page.keyboard.press('A');
await page.keyboard.up('Control');
await page.keyboard.press('Backspace');
```

Виконується комбінацію клавіш Ctrl + A, щоб вибрати весь текст у полі.

Натискається клавішу Backspace, щоб видалити наявний текст.

```
await page.type('input#searchboxinput', item);
await page.keyboard.press('Enter');
```

Вводиться новий текст (значення item) у поле вводу.

Натискається Enter, щоб виконати пошук.

```
await page.waitForTimeout(3000);
```

Очікується 3 секунди, щоб дати сторінці час на завантаження результатів.

Перевірка завантаження результатів:

```
let load = await
page.waitForSelector('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd', {
timeout: 60000 });
log(load);
```

Очікується появу елемента результатів, визначеного селектором .m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd, протягом 60 секунд.

Зберігається знайдений елемент у змінну load і записується його в лог.

Обробка результатів:

```
await
page.waitForSelector('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd', {
timeout: 60000 }).then(async () => {
```

Знову очікується наявність елемента результатів, після чого виконується блок обробки.

Автоматичний скролінг:

```
try { await autoScroll(page); log.debug('autoScroll single
done'); } catch (error) { log.error('scroll error \n', error);
}
```

Викликається функцію autoScroll(page) для автоматичного прокручування сторінки вниз (щоб завантажити більше результатів).

Якщо виникає помилка, її записується в лог.

Перевірка коротких карток

```
if (await page.$(".lcr4fd.S9kvJb") !== null) { log.debug('has
shortcards');
```

Перевіряється, чи на сторінці присутні короткі картки (.lcr4fd.S9kvJb).

Якщо вони існують, додається відповідний запис в лог.

Витягування даних із карток:

```
const cardsData = await page.evaluate(() => {
```

Виконується page.evaluate, щоб виконати JavaScript-код на сторінці та отримати дані з карток.

```
let cardsArray = [];
document.querySelectorAll('.Nv2PK.THOPZb').forEach(function
(elem) { const output = {};
```

Створюється порожній масив cardsArray для збереження даних карток.

Виконуємо цикл для кожного елемента з класом .Nv2PK.THOPZb (картка результату).

Витягування конкретних даних:

```
output.name =
elem.querySelector('.NrDZNb').textContent.replace(/,/g, ".");
```

Витягується назва об'єкта (компанії чи місця) з елемента .NrDZNb, замінюючи коми на крапки (лістинг 3.5).

Лістинг 3.5 – Витягування назви об’єкта

```
let activity = elem.querySelector(".Z8fK3b .W4Efsd .W4Efsd:nth-child(1) span:nth-child(1) span"); if (activity !== null) { output.activity = activity.textContent.replace(/,/g, "."); } else { output.activity = "null"; }
```

Перевіряється наявність інформації про діяльність та зберігається дана діяльність. Якщо дані відсутні, встановлюється значення "null".

```
let company_address = elem.querySelector(".Z8fK3b .W4Efsd .W4Efsd:nth-child(1) span:nth-child(2)"); if (company_address !== null) { output.address = company_address.textContent.replace(/,/g, "."); } else { output.address = "null"; }
```

Аналогічно витягується адреса компанії.

```
let phone = elem.querySelector(".Z8fK3b .UaQhfb .W4Efsd .W4Efsd .UsdlK"); if (phone !== null) { output.phone = phone.innerText; } else { output.phone = "null"; }
```

Витягується номер телефону, якщо він є.

```
let site_link = elem.querySelector(".Rwjeuc a"); if (site_link
!== null) { output.site = site_link.href; } else { output.site
= "null"; }
```

Витягується посилання на веб-сайт.

```
cardsArray.push(output);
```

Додається отримані дані картки в масив cardsArray.

```
return cardsArray;
```

Повертається масив зібраних даних.

Запис результатів у файл:

```
await parseResultWriter.writeRecords(cardsData);
log.warn(cardsData);
log.debug('done single shortcard');
```

Записується зібрані дані карток у файл через parseResultWriter.writeRecords (Лістинг 3.6).

Логується зібрані дані та завершення обробки коротких карток.

Стандартний потік (якщо карток немає).

Лістинг 3.6 – Запис зібраних даних

```
const URIs = await page.evaluate(() => {
  const links = document.querySelectorAll(".hfpxzc");
  let linksURIs = [];
  links.forEach((el) => {
    linksURIs.push({ "URL": el.href });
  });
  return linksURIs;
});
```

Якщо короткі картки відсутні, збирається список посилань (URIs) із елементів hfpxzc.

```
URIs.forEach(el => {
  parsedPlacesLinks.push(el["URL"]);});
await csvWriter.writeRecords(URIs);
```

Додається кожне зібране посилання у масив parsedPlacesLinks та записується їх у CSV-файл.

Завершення обробки

```
log.info("done single");
```

Записується в лог завершення обробки поточного запиту.

Обробка помилок

```
}).catch(e => {
  log('FAIL');
  log.error(`fail on ${item} query,\n`, e);});
```

У разі виникнення помилки під час виконання handleSingleQuery, записується інформація про помилку в лог.

Реалізуємо функцію автоскролінгу (лістинг 3.8).

Запуск автоскролінгу:

```
log('start autoscrolling');
```

Додається запис в лог про початок процесу автоскролінгу.

Перевіряється чи закінчився список(Лістинг 3.7).

Лістинг 3.7 – Перевірка кінця списку:

```
let endOfListSelector = await page.$('.PbZDve');
if (!!endOfListSelector) {
```

```
  log('end selector');
  return true;}
```

Шукається елемент .PbZDve, який може позначати кінець списку.

Якщо елемент знайдений (!!endOfListSelector), записується в лог інформацію та завершується функція, повертаючи true.

Проводиться перевірка кількості карток.

Лістинг 3.7 – Перевірка кількості карток

```
let placesCards = await page.$$('.hfpxzc');
if (placesCards.length > scrollCardsAmount) {
  log('too much places cards');
  return true;}
```

Отримується список карток з класом .hfpxzc.

Якщо кількість карток перевищує значення scrollCardsAmount, це записується в лог та завершується функцію, повертаючи true.

Затримка перед прокручуванням:

```
new Promise(r => setTimeout(r, 500));
```

Додається затримка у 500 мілісекунд для забезпечення плавності роботи.

Прокручування сторінки вверх:

```
await page.evaluate(() => {
  const scrollElem =
document.querySelectorAll('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd
')[1];
  scrollElem.scrollTop = scrollElem.scrollTop - 300;
});
```

Виконується прокручування елемента з класом .m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd на 300 пікселів вверх (наприклад, для видимості попередньої частини списку).

Очікування завершення завантаження мережі:

```
await page.waitForNetworkIdle();
```

Очікується стан "мережа без активності", щоб гарантувати завершення завантаження додаткових елементів.

Прокручування сторінки вниз:

```
await page.evaluate(() => {
document.querySelectorAll('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd
')[1].scrollTop = 100000;});
```

Прокручується той самий елемент вниз до максимально можливого значення (scrollTop = 100000).

Очікування завершення завантаження мережі (повторно):

```
await page.waitForNetworkIdle();
```

Знову очікується завершення активності мережі.

Рекурсивний виклик функції:

```
await autoScroll(page);
```

Викликається функція autoScroll повторно для продовження процесу скролінгу, якщо список ще не закінчився і умов для завершення немає.

Лістинг 3.8 – Функція автоскролінгу

```
async function autoScroll(page) {
  log('start autoscrolling');

  let endOfListSelector = await page.$('.PbZDve');
  if (!!endOfListSelector) {
    log('end selector')
    return true;
  }


  let placesCards = await page.$$('.hfpxzc');

  if (placesCards.length > scrollCardsAmount) {
    log('too much places cards')
```

```
        return true;
      }

      // log.warn(placesCards.length)

      new Promise(r => setTimeout(r, 500));

      await page.evaluate(() => {
        const                    scrollElem                    =
document.querySelectorAll('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd')[1
];
        scrollElem.scrollTop = scrollElem.scrollTop - 300;
      })

      await page.waitForNetworkIdle();

      await page.evaluate(() => {

document.querySelectorAll('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd')[1
].scrollTop = 100000
      })

      await page.waitForNetworkIdle();

      await autoScroll(page)

    }
```

Функція placesLinksIterator (лістинг 3.9) використовується для ітерації масиву array, обробляючи його елементи один за одним. Це простий ітератор, який працює покроково.

Перевірка завершення ітерації:

```
if (currentIteratorStep === array.length) {
  log.warn('process finished succesfully');
  return false;}
```

Перевіряється, чи досяг поточний крок currentIteratorStep кінця масиву (array.length).

Якщо всі елементи вже оброблені:- У лог виводиться повідомлення про успішне завершення процесу.

Функція повертає false, сигналізуючи про завершення ітерації.

Збільшення кроку ітерації:

```
++currentIteratorStep;
```

Значення глобальної змінної currentIteratorStep збільшується на 1, переходячи до наступного елемента масиву.

Логування кроку:

```
log.step(1);
```

У лог виводиться повідомлення про виконання ще одного кроку ітерації, що є доволі корисним для відстеження прогресу.

Повернення поточного елемента:

```
return array[currentIteratorStep - 1];
```

Функція повертає поточний елемент масиву.

Оскільки індексація починається з 0, currentIteratorStep - 1 дозволяє повернути правильний елемент масиву відповідно до кроку.

Лістинг 3.9 – Функція placesLinksIterator

```
function placesLinksIterator(array) {
  if (currentIteratorStep === array.length) {
    log.warn('process finished succesfully');
    return false;
  }

  ++currentIteratorStep

  log.step(1);
  return array[currentIteratorStep - 1]
}
```

3.3 Тестування скрипта

Запускаємо скрипт. Для цього відкриваємо консоль (відкриту програму подано на рисунку 3.3) та вводимо команду запуску скрипта (вводимо почаок

команди, вказуємо назву файлу зі скриптом, вказуємо ключові слова, потім вводимо короткий код країни та задаємо кількість карток на один запит).

Рисунок 3.3 – Відкрита програма Git Bash

Після початку роботи скрипта перед аналітиком з’явиться готовий масив даних, які будуть вводитися в рядок пошуку під час роботи скрипта. Масив даних наведений на рисунку 3.4.

Рисунок 3.4 – Масив даних

Перед користувачем відкривається веб-переглядач, в якому він може моніторити процес роботи скрипта. Веб-переглядач під час роботи зображено на рисунку 3.5.

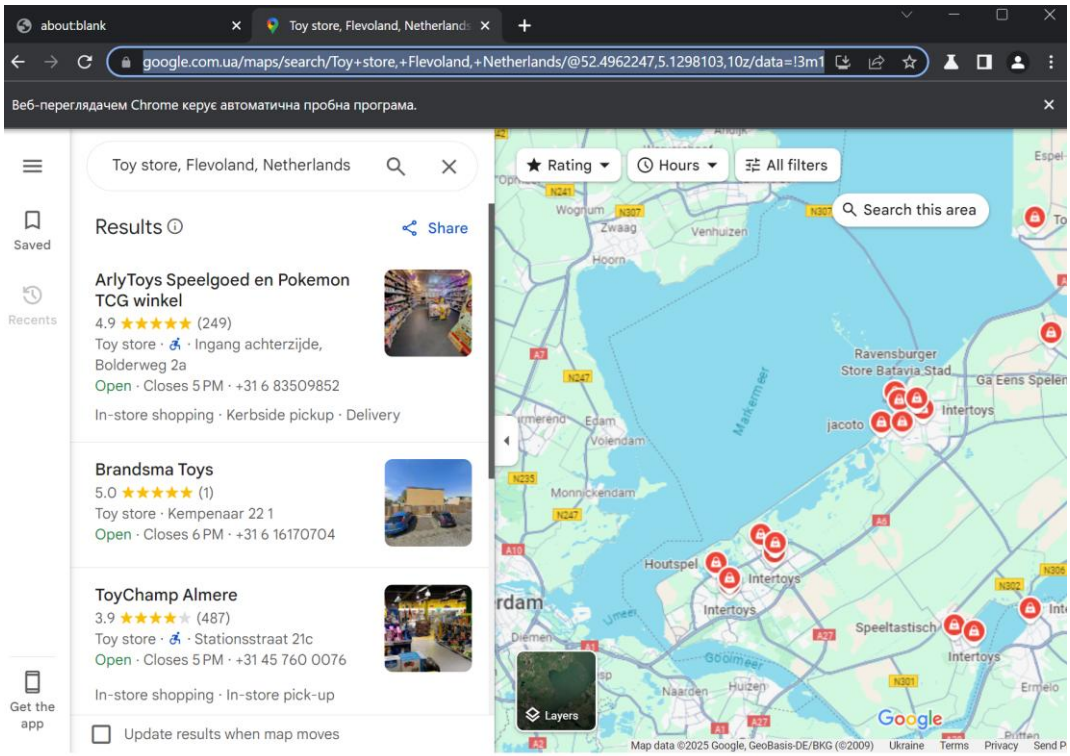


Рисунок 3.5 –Веб-переглядач під час роботи

Після обробки кожного запиту в консолі виводиться короткий звіт відповідно до виконаних дій з запитом, як на рисунку 3.6.

Рисунок 3.6 – Повідомлення в консолі

Як тільки буде перебрано всі запити і зібрано всі посилання, почнеться вигрузка інформації, вказаної в картках компаній, тобто назви компаній, статус, адреса, телефон та посилання сайт. Кількість зібраних результатів та кількість пройдених сайтів також виводяться в консолі, приклад наведено на рисунку 3.7.

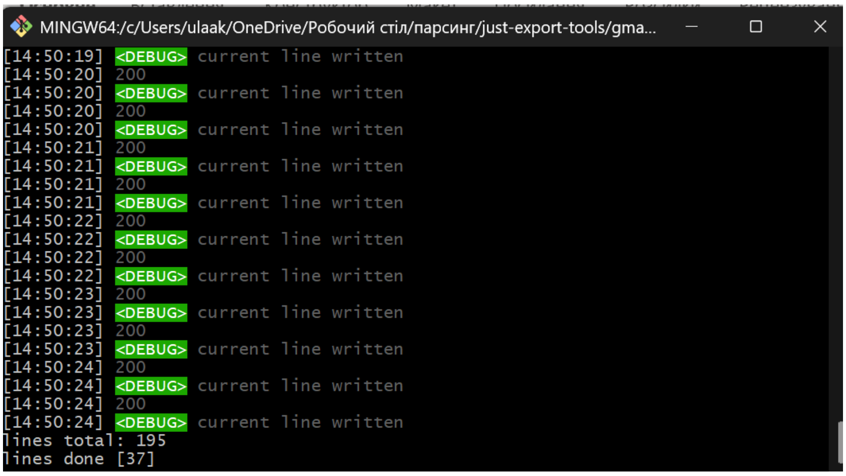


Рисунок 3.7 – Процес збору даних

Після завершення роботи скрипта в консолі буде виведено повідомлення про успішне завершення роботи.

Якщо запит не вдалось виконати повністю через певні проблеми (наприклад перебої з інтернетом), буде виведено повідомлення про помилку (рисунок 3.8).

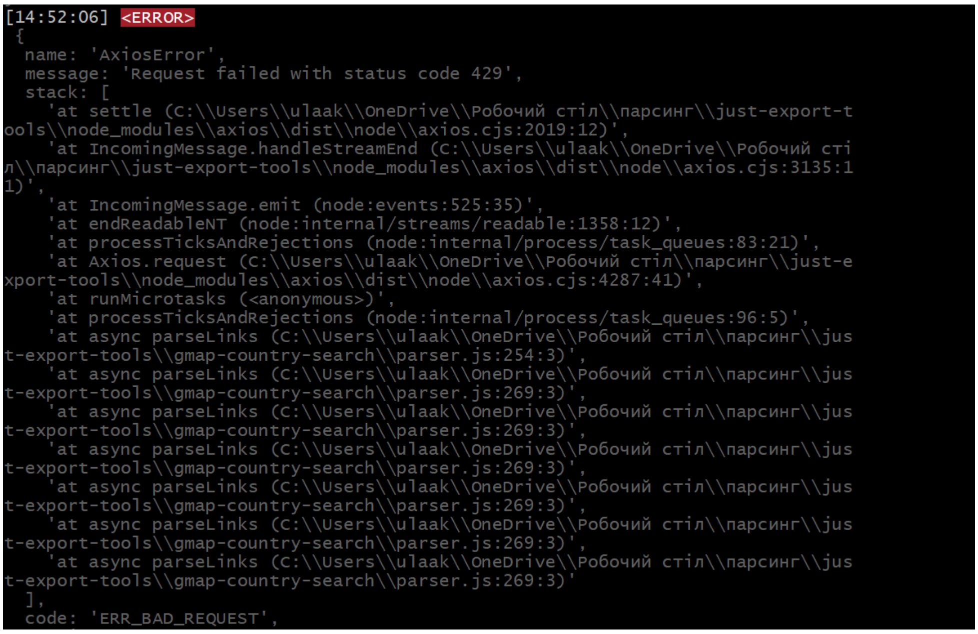
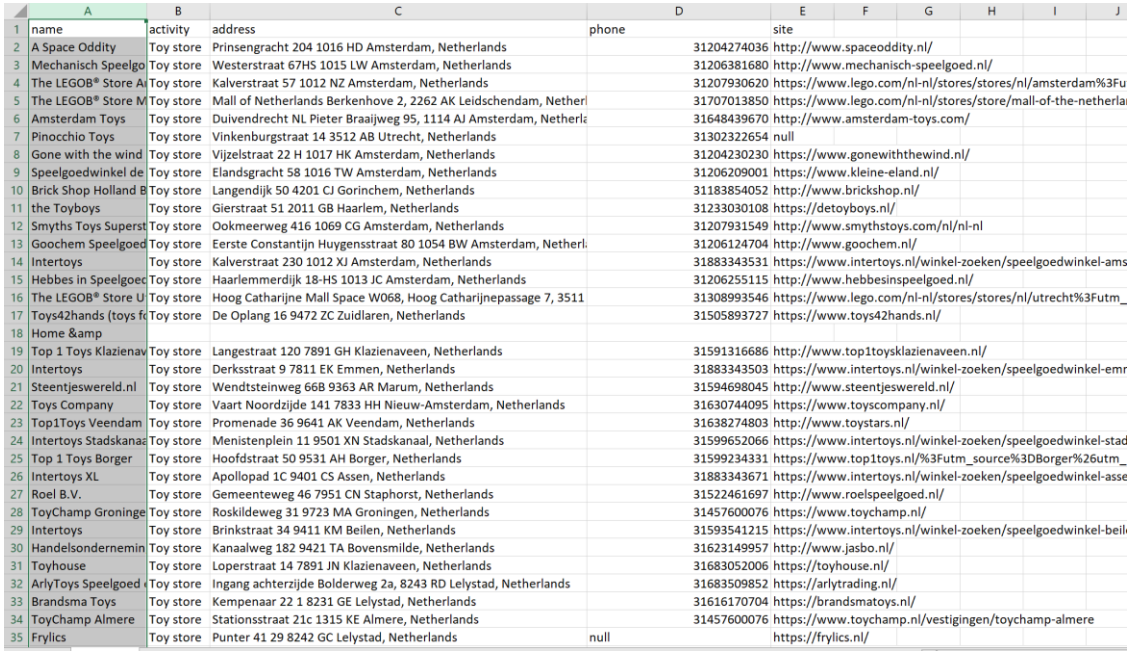


Рисунок 3.8 – Повідомлення про помилку

Зібрана інформація зберігається в CSV-файлі. Результати вигрузки даних можна побачити на рисунку 3.9.



	A	B	C	D	E	F	G	H	I	J
1	name	activity	address	phone	site					
2	A Space Oddity	Toy store	Prinsengracht 204 1016 HD Amsterdam, Netherlands	31204274036	http://www.spaceoddity.nl/					
3	Mechanisch Speelgo	Toy store	Westerstraat 67HS 1015 LW Amsterdam, Netherlands	31206381680	http://www.mechanisch-speelgoed.nl/					
4	The LEGOB® Store A	Toy store	Kalverstraat 57 1012 NZ Amsterdam, Netherlands	31207930620	https://www.lego.com/nl-nl/stores/stores/nl/amsterdam%3Fu					
5	The LEGOB® Store M	Toy store	Mall of Netherlands Berkenhove 2, 2262 AK Leidschendam, Nether	31707013850	https://www.lego.com/nl-nl/stores/store/mall-of-the-netherla					
6	Amsterdam Toys	Toy store	Duivendrecht NL Pieter Braaijweg 95, 1114 AJ Amsterdam, Netherla	31648439670	http://www.amsterdam-toys.com/					
7	Pinocchio Toys	Toy store	Vinkenburgstraat 14 3512 AB Utrecht, Netherlands	31302322654	null					
8	Gone with the wind	Toy store	Vijzelstraat 22 H 1017 HK Amsterdam, Netherlands	31204230230	https://www.gonewiththewind.nl/					
9	Speelgoedwinkel de	Toy store	Elandsgracht 58 1016 TW Amsterdam, Netherlands	31206209001	https://www.kleine-eland.nl/					
10	Brick Shop Holland B	Toy store	Langendijk 50 4201 CJ Gorinchem, Netherlands	31183854052	http://www.brickshop.nl/					
11	the Toyboys	Toy store	Gierstraat 51 2011 GB Haarlem, Netherlands	31233030108	https://detoyboys.nl/					
12	Smyths Toys Superst	Toy store	Ookmeerweg 416 1069 CG Amsterdam, Netherlands	31207931549	http://www.smythstoys.com/nl/nl-nl					
13	Goochem Speelgoed	Toy store	Eerste Constantijn Huygensstraat 80 1054 BW Amsterdam, Netherl	31206124704	http://www.goochem.nl/					
14	Intertoys	Toy store	Kalverstraat 230 1012 XJ Amsterdam, Netherlands	31883343531	https://www.intertoys.nl/winkel-zoeken/speelgoedwinkel-ams					
15	Hebbes in Speelgoec	Toy store	Haarlemmerdijk 18-HS 1013 JC Amsterdam, Netherlands	31206255115	http://www.hebbesinspeelgoed.nl/					
16	The LEGOB® Store U	Toy store	Hoog Catharijne Mall Space W068, Hoog Catharijnepassage 7, 3511	31308993546	https://www.lego.com/nl-nl/stores/stores/nl/utrecht%3Futm_					
17	Toys42hands (toys f	Toy store	De Oplang 16 9472 ZC Zuidlaren, Netherlands	31505893727	https://www.toys42hands.nl/					
18	Home &amp									
19	Top 1 Toys Klazienav	Toy store	Langestraat 120 7891 GH Klazienaveen, Netherlands	31591316686	http://www.top1toysklazienaveen.nl/					
20	Intertoys	Toy store	Derksstraat 9 7811 EK Emmen, Netherlands	31883343503	https://www.intertoys.nl/winkel-zoeken/speelgoedwinkel-emr					
21	Steentjeswereld.nl	Toy store	Wendtsteinweg 66B 9363 AR Marum, Netherlands	31594698045	http://www.steentjeswereld.nl/					
22	Toys Company	Toy store	Vaart Noordzijde 141 7833 HH Nieuw-Amsterdam, Netherlands	31630744095	https://www.toyscompany.nl/					
23	Top1Toys Veendam	Toy store	Promenade 36 9641 AK Veendam, Netherlands	31638274803	http://www.toystars.nl/					
24	Intertoys Stadskanaa	Toy store	Menistenplein 11 9501 XN Stadskanaal, Netherlands	31599652066	https://www.intertoys.nl/winkel-zoeken/speelgoedwinkel-stad					
25	Top 1 Toys Borger	Toy store	Hoofdstraat 50 9531 AH Borger, Netherlands	31599234331	https://www.top1toys.nl/%3Futm_source%3DBorger%26utm_					
26	Intertoys XL	Toy store	Apollopad 1C 9401 CS Assen, Netherlands	31883343671	https://www.intertoys.nl/winkel-zoeken/speelgoedwinkel-asse					
27	Roel B.V.	Toy store	Gemeenteweg 46 7951 CN Staphorst, Netherlands	31522461697	http://www.roelspeelgoed.nl/					
28	ToyChamp Groninge	Toy store	Roskildeweg 31 9723 MA Groningen, Netherlands	31457600076	https://www.toychamp.nl/					
29	Intertoys	Toy store	Brinkstraat 34 9411 KM Beilen, Netherlands	31593541215	https://www.intertoys.nl/winkel-zoeken/speelgoedwinkel-beil					
30	Handelsondernemin	Toy store	Kanaalweg 182 9421 TA Bovensmilde, Netherlands	31623149957	http://www.jasbo.nl/					
31	Toyhouse	Toy store	Loperstraat 14 7891 JN Klazienaveen, Netherlands	31683052006	https://toyhouse.nl/					
32	ArlyToys Speelgoed	Toy store	Ingang achterzijde Bolderweg 2a, 8243 RD Lelystad, Netherlands	31683509852	https://arlytrading.nl/					
33	Brandsma Toys	Toy store	Kempenaar 22 1 8231 GE Lelystad, Netherlands	31616170704	https://brandsmatoys.nl/					
34	ToyChamp Almere	Toy store	Stationsstraat 21c 1315 KE Almere, Netherlands	31457600076	https://www.toychamp.nl/vestigingen/toychamp-almere					
35	Frylics	Toy store	Punter 41 29 8242 GC Lelystad, Netherlands	null	https://frylics.nl/					

Рисунок 3.9 – Інформація, зібрана під час парсингу

3.4 Висновки до розділу 3

В третьому розділі кваліфікаційної роботи було сформовано алгоритм формування цільової клієнтської бази для замовника та алгоритм безпосередньої роботи скрипта. Визначено основні аспекти роботи скрипта та пояснено процес формування відбору даних для клієнта.

Також було реалізовано основний функціонал скрипта за допомогою Node.js та текстового редактора Visual Studio Code.

Проведено тестування працездатності скрипта за допомогою запиту на пошук магазинів іграшок в Нідерландах. Після його виконання, було зроблено висновок, що розроблений скрипт вирішує основні завдання, які перед ним було поставлено:

1. Формує масив запитів для роботи;
2. Проганяє масив запитів в Google Maps;
3. Збирає результати в кінцевий файл.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

4.1.1 Дія електричного струму на організм людини, види електротравм

Вплив електричного струму на організм людини є складним і багатогранним. При проходженні струму через тіло він чинить термічний, електролітичний та біологічний вплив, що може спричиняти серйозні порушення функціонування життєво важливих органів—мозку, серця та легенів.

Термічний ефект струму призводить до опіків різних ділянок тіла. Електролітичний вплив викликає розклад крові та інших органічних рідин, що суттєво змінює їх фізико-хімічний склад. Біологічна дія електричного струму спричиняє подразнення та збудження живих тканин, призводячи до неконтрольованих судомних скорочень м’язів, зокрема легенів і серця, що може порушувати біологічні процеси.

Усі види ураження електричним струмом поділяються на дві основні групи: електричні травми та електричні удари. Електричні травми—це локальні ушкодження тканин і органів внаслідок дії струму або електричної дуги. До основних видів електротравм належать: електричний опік; електричні знаки металізація шкіри—проникнення дрібних частинок металу в поверхневі шари шкіри; електроофтальмія—ушкодження очей через інтенсивне електромагнітне випромінювання; механічні ушкодження.

Електричний струм може мати серйозні наслідки для здоров’я, тому важливо дотримуватися правил електробезпеки та уникати контакту з небезпечними електричними джерелами.

Електричний опік є найбільш поширеною електротравмою, яка може виникнути внаслідок впливу електричного струму на людський організм. Вони

бувають двох видів: струмові (контактні) та дугові. Струмовий опік виникає, коли струм силою понад 1А проходить через тіло людини в результаті контакту зі струмоведучими частинами, що призводить до перетворення електричної енергії в теплову. Розрізняють чотири ступені ураження шкіри: перший ступінь супроводжується почервонінням, другий – утворенням пухирів, третій – некрозом усіх шарів шкіри, а четвертий – обвугленням тканин. Однак ступінь небезпеки визначається не лише глибиною ураження, а й площею постраждалої поверхні. Дугові опіки виникають при високих напругах, коли між струмоведучою частиною та тілом людини утворюється електрична дуга, температура якої перевищує 3500°С. Такі опіки є вкрай важкими та зазвичай належать до третього або четвертого ступеня.

Окрім опіків, електрична травма може призвести до появи електричних знаків – чітко окреслених плям сірого або жовтого кольору в місці контакту з електродами. Вони можуть мати вигляд подряпин, порізів, ран, мозолів або бородавок, але є безболісними і зазвичай мають діаметр до 10 мм.

Ще одним видом електротравми є металізація шкіри – проникнення розплавлених частинок металу у її верхні шари. Це може статися через електричну дугу під час короткого замикання або вимкнення рубильника під навантаженням і часто супроводжується опіками [10].

Електроофтальмія – це ураження очей, викликане інтенсивним випромінюванням електричної дуги, яке містить шкідливі ультрафіолетові та інфрачервоні промені. Крім того, можливе потрапляння в очі розплавлених частинок металу. Захиститися від такого ураження можна за допомогою спеціальних захисних окулярів.

Механічні ушкодження виникають унаслідок неконтрольованих судомних скорочень м’язів під дією електричного струму. Це може призвести до розривів шкіри, кровоносних судин і нервових волокон, а також спричинити вивихи або переломи кісток.

Електричний удар – це вплив струму на організм, що викликає неконтрольовані судомні скорочення м’язів. Залежно від його наслідків

виділяють чотири ступені ураження: перший – судомне скорочення м’язів без втрати свідомості, другий – втрата свідомості при збереженні дихання та серцевої діяльності, третій – втрата свідомості з порушенням роботи серця або дихання, а четвертий – клінічна смерть, коли дихання та кровообіг припиняються.

Клінічна смерть є перехідним станом між життям і смертю. Її основні ознаки – зупинка серця, відсутність пульсу та дихання, розширення зіниць без реакції на світло, відсутність больових відчуттів. Тривалість такого стану може тривати від 5 до 10 хвилин, залежно від сили ураження та індивідуальних особливостей організму.

4.2.2 Вимоги безпеки до ПК та устаткування

Правила охорони праці під час роботи з ПК поширюються на всі підприємства, незалежно від форми власності, які використовують персональні комп’ютери з відеодисплейними терміналами та периферійними пристроями. Вони встановлюють вимоги безпеки для облаштування робочих місць операторів комп’ютерної техніки та регламентують правила її експлуатації. Дотримання цих вимог є обов’язковим для роботодавців і працівників, які працюють з ПК.

Вимоги не застосовуються до навчальних комп’ютерних класів, робочих місць в атомній енергетиці, транспортних системах, портативних пристроїв, касових апаратів, друкарських машинок із дисплеями, ігрових автоматів та інших систем для громадського використання. Однак їх повинні враховувати всі працівники, які займаються експлуатацією, технічним обслуговуванням, налаштуванням, ремонтом комп’ютерної техніки, а також при проєктуванні та реконструкції підприємств, де вона використовується.

У питаннях електробезпеки вся комп’ютерна техніка, включаючи апаратуру керування, вимірювальні прилади та освітлення, повинна відповідати встановленим нормам. Для уникнення загоряння через коротке замикання

необхідно використовувати негорючу ізоляцію та виключити можливість перевантаження електромережі. Під час ремонту електричних систем слід дотримуватися вимог протипожежної безпеки.

Електроживлення ПК має здійснюватися через окрему трипровідну мережу, що включає фазовий, нульовий робочий та нульовий захисний провідники. Останній використовується для заземлення, його не можна замінювати робочим нульовим провідником. У приміщеннях, де експлуатується понад п’ять ПК, необхідно встановити аварійний вимикач для повного відключення електроживлення, за винятком освітлення.

Підключення комп’ютерів допускається лише через заводські штепсельні з’єднання та розетки, які мають окремі контакти для нульового захисного провідника. Його приєднання повинно відбуватися перед з’єднанням інших провідників, а відключення — в зворотному порядку. Заборонено використовувати перехідники для підключення ПК до двопровідної електромережі. Лінії живлення потрібно прокладати у спеціальних металевих або пластмасових захисних оболонках, а якщо в приміщенні працює до п’яти ПК, допускається використання кабелів з негорючою ізоляцією без додаткових захисних елементів.

Всі електромережі повинні відповідати ПУЕ [12], параметрам навантаження, умовам експлуатації та вимогам безпеки. Дотримання цих правил гарантує безпечну роботу комп’ютерної техніки та запобігає аварійним ситуаціям.

4.2 Безпека в надзвичайних ситуаціях

Оцінка стійкості роботи об'єкту економіки до впливу вражаючих факторів ядерної зброї.

Сучасний світ стикається з новими викликами безпеки, серед яких загроза застосування ядерної зброї займає одне з провідних місць.[23] Хоча ймовірність широкомасштабної ядерної війни наразі залишається відносно низькою,

локальні конфлікти, зростання політичної напруги та поширення нових видів зброї масового ураження змушують переглядати стратегії захисту цивільних та економічних об’єктів. Об’єкти економіки — це ключові елементи, що забезпечують життєдіяльність держави: промислові підприємства, об’єкти енергетики, транспорту, логістики, зв’язку тощо. Їхня стійкість до ураження має вирішальне значення для збереження функціонування країни в умовах надзвичайної ситуації.[24]

Вражаючі фактори ядерної зброї мають багатовекторний вплив на об’єкти інфраструктури та персонал. Основні з них:

Ударна хвиля — основний механічний фактор, що спричиняє руйнування будівель, обладнання, комунікацій, транспортних засобів.

Світлове випромінювання — може викликати пожежі, ураження відкритих поверхонь конструкцій і опіки у людей.

Проникаюча радіація — небезпечна для людей і електроніки, призводить до миттєвих втрат серед персоналу.

Радіоактивне зараження місцевості (РЗМ) — довготривале забруднення місцевості, унеможливлює нормальне функціонування об’єкта[28].

Електромагнітний імпульс (ЕМІ) — виходить від ядерного вибуху на великій висоті, виводить з ладу електроніку, засоби зв’язку, управління, автоматику[29].

Вплив цих факторів залежить від типу вибуху (наземний, повітряний, підземний), потужності боєприпасу, відстані до епіцентру, географічного розташування об’єкта тощо.

Оцінка стійкості передбачає аналіз комплексу технічних, організаційних та інженерних характеристик об’єкта:

- Фізична вразливість конструкцій визначається їхньою міцністю та здатністю протистояти зовнішнім впливам, таким як ударна хвиля та світлове випромінювання. Чим вища міцність матеріалів, з яких зведені будівлі, тим краще вони витримують механічні навантаження та зберігають свою структурну цілісність. Крім того, конструкції можуть бути розраховані на поглинання або

відбиття світлового випромінювання, що впливає на їхню здатність протистояти термічним і оптичним ефектам. Використання спеціальних матеріалів і технологій проєктування допомагає підвищити стійкість будівель до екстремальних умов. [23].

- Системи захисту відіграють ключову роль у забезпеченні безпеки людей та об'єктів у разі надзвичайних ситуацій. Вони включають різноманітні укриття, герметичні приміщення та протипожежні системи, кожна з яких має свої особливості та призначення.

Укриття – це спеціально обладнані споруди, призначені для захисту людей від впливу вибухових хвиль, радіації, хімічного забруднення чи інших небезпек. Вони можуть бути як стаціонарними (бомбосховища, підземні укриття), так і мобільними (укриття, що швидко встановлюються у кризових ситуаціях). Їхня конструкція передбачає міцні матеріали та ефективну систему вентиляції, яка дозволяє підтримувати комфортний рівень кисню всередині.

Герметичні приміщення – це спеціальні ізольовані зони, створені для запобігання проникненню забрудненого повітря, шкідливих хімічних речовин або біологічних загроз. Вони оснащені системами очищення повітря, автономним енергопостачанням і запасами необхідних ресурсів, щоб забезпечити безпечне перебування людей у таких приміщеннях протягом тривалого часу.

Протипожежні системи – включають комплекс заходів та обладнання, яке запобігає виникненню пожеж, а також забезпечує їх швидку локалізацію та ліквідацію. До них належать системи автоматичного пожежогасіння, датчики диму, вогнегасники, спеціалізовані двері, що перешкоджають поширенню вогню, а також вентиляційні механізми, які ефективно видаляють токсичні гази під час займання.Рівень автоматизації — дозволяє дистанційно керувати процесами в умовах втрати персоналу.

- Залежність від зовнішніх мереж, таких як електропостачання, водопостачання та зв’язок, є важливим фактором, що впливає на загальну стійкість об’єкта чи системи. Чим більше автономності має будівля,

інфраструктура або організація, тим меншою є її вразливість перед зовнішніми збоями чи аваріями.

Автономне електропостачання може забезпечуватися за рахунок альтернативних джерел енергії, таких як сонячні батареї, вітрові турбіни або генератори. Використання накопичувальних систем, зокрема акумуляторів або резервних генераторів, дозволяє продовжити роботу у випадку відключення основних джерел живлення. [25].

- Запасні варіанти функціонування — дублюючі системи, резервні генератори, евакуаційні маршрути.

Ці критерії дозволяють об’єктивно оцінити, наскільки об’єкт зможе зберегти працездатність або оперативно відновити роботу після ураження.

Процес оцінки включає кілька етапів:

Збір вихідних даних: характеристика об’єкта, координати, типи конструкцій, основне обладнання, кількість персоналу.

Визначення параметрів уражаючих факторів: залежно від сценарію ядерного вибуху (потужність, відстань, тип вибуху)[27].

Моделювання впливу: розрахунок зон дії ударної хвилі, температур, доз радіації, рівня ЕМІ.

Аналіз ушкоджень: виявлення критичних елементів, визначення ступеня ураження основних систем.

Розрахунок коефіцієнта стійкості (Кс) — умовний показник, що демонструє здатність об’єкта продовжувати роботу після впливу[23]:

$$Кс = \frac{Пф}{Пн}$$

Де, Пф — фактичний обсяг функціонування після ураження, Пн — номінальний обсяг роботи.

Розглянемо умовний приклад: хлібопекарський завод, розташований за 5 км від умовного епіцентру ядерного вибуху потужністю 100 кт.

Ударна хвиля: значне пошкодження фасадів і часткове руйнування складів.

Світлове випромінювання: займання покрівлі, пошкодження обладнання, яке знаходилось на відкритій території.

Проникаюча радіація: ймовірне ураження до 30% персоналу.

РЗМ: забруднення території вище допустимих норм, робота без захисту неможлива[28].

ЕМІ: виведено з ладу системи керування печами та транспортерів[29].

Після аналізу та умовного розрахунку коефіцієнта стійкості отримаємо:

Кс=0.3 — об’єкт не може виконувати повну функцію, лише частково після аварійного ремонту та підключення резервних систем.

Підвищення стійкості можливе завдяки реалізації комплексу заходів:

Інженерний захист: будівництво захисних конструкцій, підземних сховищ, посилення будівель[24].

Розосередження: розміщення виробництва в кількох локаціях для уникнення повного паралічу.

Автономні джерела енергії: генератори, водозабірні системи, локальні мережі[25].

Резервування систем: дублювання критичних вузлів, електроніки, каналів зв’язку.

Навчання персоналу: тренування дій в умовах НС, евакуаційні навчання, наявність ЗІЗ (засобів індивідуального захисту)[27].

Системи оповіщення та моніторингу: встановлення датчиків радіації, температури, тиску[26].

У сучасних умовах наявність ядерної загрози вимагає глибокого аналізу та підготовки економічних об’єктів до надзвичайних ситуацій. Оцінка стійкості є ключовим елементом у системі цивільного захисту та національної безпеки. Вона дозволяє виявити вразливі місця, спланувати заходи щодо підвищення життєздатності об’єкта та мінімізувати втрати у випадку ядерного ураження.

Інтеграція інженерних, організаційних та технологічних рішень у поєднанні з ефективним управлінням персоналом значно підвищує здатність економіки функціонувати навіть в умовах найгірших сценаріїв.

4.3 Висновки до розділу 4

При експлуатації електронно-обчислювальних машин можна зробити висновок, що безпека працівників значною мірою залежить від правильного облаштування робочих місць, дотримання норм електробезпеки та використання відповідного обладнання. Вимоги щодо проєктування, монтажу та експлуатації електромережі спрямовані на запобігання аварійним ситуаціям та мінімізацію ризиків, пов’язаних із несправностями електрообладнання.

Окрему увагу слід приділити використанню трипровідної системи електроживлення, що забезпечує надійне заземлення і захист від струмових перевантажень. Правильне підключення ПК через заводські штепсельні розетки з окремими контактами для нульового захисного провідника гарантує безпеку користувачів, а наявність аварійного вимикача у великих приміщеннях дозволяє швидко знеструмити обладнання у випадку небезпеки.

Також важливим є правильне прокладання електромережі, що враховує навантаження, умови експлуатації та відповідність чинним стандартам безпеки. Використання негорючих матеріалів у розетках і кабелях допомагає запобігти можливому займання та забезпечує додатковий рівень захисту.

Загалом, впровадження всіх передбачених вимог охорони праці сприяє безпечній роботі з комп’ютерною технікою, захисту працівників та підтримці стабільності електромережі, що вкрай важливо для безперервного функціонування підприємств і установ. Дотримання правил дозволяє запобігти аваріям та забезпечити ефективне використання електронного обладнання

ВИСНОВКИ

Застосування інформаційних систем у логістиці має великі перспективи, оскільки підприємство, як система, потребує взаємозв'язку між її складовими частинами для створення інтегрованої та складної структури. Запровадження інформаційних технологій у бізнес стратегіях та логістиці, а саме для збору даних про компанії та постачальників з метою розширення клієнтської бази є доволі актуальною задачею.

Вивчаючи цей напрямок, аналізуючи велику кількість інформації та готових рішень, можна дійти до висновку, що даний напрям є перспективний в своєму розвитку. Готові рішення допомогли сформулювати чітке бачення майбутнього продукту, порівняння їх надало можливість визначити основні функції та алгоритми для майбутньої розробки.

Наступним кроком був вибір середовища та інструментів для розробки. За допомогою порівняльного аналізу середовищ та платформ для розробки, а також програм для виконання, було відібрано найзручніші для реалізації поставлених завдань.

Беручи за основу початковий аналіз існуючих рішень та базове бачення роботи скрипта було сформовано основні алгоритми для роботи скрипта. Запуск скрипта після реалізації всіх задуманих функцій показав, що скрипт виконує поставлені перед ним задачі та зручний у використанні.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lesson 7: Wrap-up! Build your first scraping task | Help Center [Електронний ресурс]. – Режим доступу: https://helpcenter.octoparse.com/en/articles/6470927-lesson-7-wrap-up-build-your-first-scraping-task – Дата звернення: 12.02.2025.

2. Wise A. Netpeak Checker review from 2023: a tool for analyzing site SEO metrics [Електронний ресурс]. – Режим доступу: https://netpeak.net/ru/blog/obzor-netpeak-checker-2023-multifunktsionalnogo-instrumenta-dlya-massovogo-analiza-i-sravneniya-saitov/ – Дата звернення: 15.11.2024.

3. Blog. Buy proxy – Private Socks5 & HTTPs proxies | Proxy-Seller [Електронний ресурс]. – Режим доступу: https://proxy-seller.com/ua/blog/oglyad-veb-skrapera-parsehub/#HAnch1 – Дата звернення: 15.11.2024.

4. Git bash: Definition, commands, & getting started | Atlassian [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/git/tutorials/git-bash, вільний. – Дата звернення: 01.03.2025.

5. Учасники проектів Вікімедіа. Visual Studio Code — Вікіпедія [Електронний ресурс]. – 30.04.2015. – Режим доступу: https://uk.wikipedia.org/wiki/Visual_Studio_Code – Дата звернення: 01.03.2025.

6. What Is Git Bash and How Do You Use It? MUO [Електронний ресурс]. – Режим доступу: https://www.makeuseof.com/git-bash-what-how-use/ – Дата звернення: 01.03.2025.

7. Git bash: Definition, commands, & getting started | Atlassian [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/git/tutorials/git-bash – Дата звернення: 10.03.2025.

8. Правове регулювання парсингу | LCF [Електронний ресурс]. – Режим доступу: https://lcf.ua/news/pravove-reguliuvannia-parsyngu/ – Дата звернення: 10.05.2025.

9. Парсинг даних з сайтів: що це та на чиєму боці закон [Електронний ресурс]. – Режим доступу: https://highload.tech/uk/parsing/ – Дата звернення: 10.05.2025.

10. Закон України «Про охорону праці» [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12

11. Санітарні правила і норми для роботи з персональними електронно-обчислювальними машинами [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0400282-93 - Дата звернення: 10.05.2025.

12. Міненерговугілля України. «Правила улаштування електроустановок» [Електронний ресурс]. – Режим доступу: https://zakon.isu.net.ua/sites/default/files/normdocs/pue.pdf - 10.05.2025

13. ISO/IEC 27001:2022 – Information Security Management Systems [Електронний ресурс]. – Режим доступу: https://www.iso.org/isoiec-27001-information-security.html - Дата звернення: 10.05.2025.

14. SAP SCM – Офіційний сайт SAP [Електронний ресурс]. – Режим доступу: https://www.sap.com/products/scm.html - Дата звернення: 10.05.2025.

15. Odoo Inventory [Електронний ресурс]. – Режим доступу: https://www.odoo.com - Дата звернення: 10.05.2025.

16. Route4Me Route Optimization Software [Електронний ресурс]. – Режим доступу: https://route4me.com - Дата звернення: 10.05.2025.

17. DHL Innovation Hub [Електронний ресурс]. – Режим доступу: https://www.dhl.com/global-en/home/insights-and-innovation.html - Дата звернення: 10.05.2025.

18. Automation and the Future of the Workforce in Logistics [Електронний ресурс] / Seegrid. – 2023. – Режим доступу: https://hub.seegrid.com/blog/automation-and-the-future-of-the-workforce-in-logistics – Дата звернення: 10.05.2025.

19. The Future of Warehouse Work: Technological Change in the U.S. [Електронний ресурс] / Berkeley Labor Center. – 2020. – Режим доступу:

https://laborcenter.berkeley.edu/future-of-warehouse-work – Дата звернення: 10.05.2025.

20. Enhancing operations management through smart sensors [Електронний ресурс] / arXiv. – 2021. – Режим доступу: https://arxiv.org/abs/2112.08213 – Дата звернення: 10.05.2025.

21. AI disempowers logistics workers while intensifying their work [Електронний ресурс] / ComputerWeekly. – 2023. – Режим доступу: https://www.computerweekly.com/feature/AI-disempowers-logistics-workers-while-intensifying-their-work – Дата звернення: 10.05.2025.

22. Occupational safety and health [Електронний ресурс] / Wikipedia. – Режим доступу: https://en.wikipedia.org/wiki/Occupational_safety_and_health – Дата звернення: 10.05.2025.

23. Бойко В. І., Ляшенко В. І. Цивільна оборона та основи охорони праці : підручник. – К. : Центр навчальної літератури, 2012. – 432 с.

24. Соколовський В. М. Безпека життєдіяльності та цивільний захист : навч. посіб. – Харків : НТУ «ХПІ», 2018. – 240 с.

25. ДСТУ ISO 22301:2015. Системи управління безперервністю бізнесу. Вимоги. – [Чинний від 2015-12-01]. – К. : ДП «УкрНДНЦ», 2015. – 34 с.

26. Національна стратегія забезпечення безпеки критичної інфраструктури України : Указ Президента України №56/2021 від 16.02.2021 [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/56/2021 – Дата звернення: 10.05.2025.

27. Державна служба України з надзвичайних ситуацій [Електронний ресурс]. – Режим доступу: https://dsns.gov.ua – Дата звернення: 10.05.2025.

28. International Atomic Energy Agency (IAEA). Official website [Електронний ресурс]. – Режим доступу: https://www.iaea.org – Назва з екрана.

29. UN Office for Disarmament Affairs (UNODA). United Nations Disarmament [Електронний ресурс]. – Режим доступу: https://www.un.org/disarmament - Дата звернення: 10.05.2025.

30. ELARTU – Інституційний репозитарій ТНТУ імені Івана Пулюя: Домівка. Режим доступу: https://elartu.tntu.edu.ua/bitstream/lib/41896/1/2023_KRB_SNs-41_Yakubyak_YP.pdf - Дата звернення: 27.05.2025)

31. Atom Review: Pricing, Pros, Cons & Features. CompareCamp.com [Електронний ресурс]. – Режим доступу: https://comparecamp.com/atom-review-pricing-pros-cons-features/ – Дата звернення: 10.05.2025.

32. Atom vs Atom-IDE | What are the differences? StackShare [Електронний ресурс]. – Режим доступу: https://stackshare.io/stackups/atom-vs-atom-ide – Дата звернення: 10.05.2025.

33. Visual Studio Code Vs Atom: Which Code Editor Is Better [Електронний ресурс]. – Режим доступу: https://www.softwaretestinghelp.com/visual-studio-code-vs-atom/ – Дата звернення: 10.05.2025.

34. Sunsetting Atom. The GitHub Blog [Електронний ресурс]. – Режим доступу: https://atom.io/ – Дата звернення: 10.05.2025.

35. Microsoft. Visual Studio Code – Code Editing. Redefined [Електронний ресурс]. – 03.11.2021. – Режим доступу: https://code.visualstudio.com/ – Дата звернення: 10.05.2025.

36. Microsoft. Documentation for Visual Studio Code [Електронний ресурс]. – 03.11.2021. – Режим доступу: https://code.visualstudio.com/docs – Дата звернення: 10.05.2025.

37. JetBrains: Essential tools for software developers and teams [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/ – Дата звернення: 10.05.2025.

38. JetBrains Product Documentation [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/help/ – Дата звернення: 10.05.2025.

39. Sublime Text – the sophisticated text editor for code, markup and prose [Електронний ресурс]. – Режим доступу: https://www.sublimetext.com/ – Дата звернення: 10.05.2025.

40. Documentation. Sublime Text – Text Editing, Done Right [Електронний ресурс]. – Режим доступу: https://www.sublimetext.com/docs/ – Дата звернення: 10.05.2025.

41. Package Control – the Sublime Text package manager [Електронний ресурс]. – Режим доступу: https://packagecontrol.io/ – Дата звернення: 10.05.2025.

42. Package Control - the Sublime Text package manager. Package Control - the Sublime Text package manager. [Електронний ресурс]. – Режим доступу: https://packagecontrol.io/ – Дата звернення: 10.05.2025.

43. atom/atom · Discussions. GitHub. [Електронний ресурс]. – Режим доступу: https://github.com/atom/atom/discussions – Дата звернення: 10.05.2025.

44. Stack Overflow Developer Survey 2023. Stack Overflow. Електронний ресурс]. – Режим доступу: https://survey.stackoverflow.co/2023/ – Дата звернення: 10.05.2025.

45. Fryz M., Mlynko B. Property Analysis of Conditional Linear Random Process as a Mathematical Model of Cyclostationary Signal. 2nd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2022). Ternopil, Ukraine: CEUR Workshop Proceedings, 2022. Vol. 3309. P. 77–82.

46. Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques. Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132.

47. M. Fryz, B. Mlynko, Property analysis of multivariate conditional linear random processes in the problems of mathematical modelling of signals, Technology Audit and Production Reserves, 3/2(65), 2022, pp. 29–32.

48. Бабак В. П., Марченко М. Є., Фриз. Б. Г. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.

49. Б.Б. Млинко, канд.техн.наук, доц.; Ю.П. Якуб’як. РОЗРОБКА АЛГОРИТМУ ОПТИМІЗАЦІЇ ПРОЦЕСУ ЗБОРУ ДАНИХ ПОТЕНЦІЙНИХ КЛІЄНТІВ. АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ : XIII Міжнар.

науково-практ. конф. молодих уч. та студентів, м. Тернопіль, 11 груд. 2024 р. Тернопіль, 2024. С. 215–216. URL: https://tntu.edu.ua/storage/pages/00001070/TNTU_Aktualni_zadachi_suchasnyh_tehnologiy_2024.pdf - Дата звернення: 10.05.2025.

50. Інформаційні системи в логістиці : Державний університет телекомунікацій. Онлайн. Головна. Режим доступу: https://dut.edu.ua/ua/lib/1/category/1137/view/1483. Дата звернення: 10.05.2025.

51. Парсинг сайтів: що це і навіщо він потрібен? Онлайн. Webpromo. Режим доступу: https://web-promo.ua/ua/blog/parsing-sajtov-chto-eto-i-zachem-nuzhen/ - Дата звернення: 10.05.2025.

52. Олександр Тартачний. Розширення для браузерів, хмарні сервіси та бібліотеки. robot_dreams - онлайн-курси для фахівців у сфері big data, machine learning, data science | Робот Дрімс. URL: https://robotdreams.cc/uk/blog/125-8-instrumentov-dlya-parsinga-saytov - Дата звернення: 14.05.2025.

ДОДАТКИ

Лістинг 3.3 – Функція для обробки запиту

```
async function handleSingleQuery(page, item) {
  // handle keyboard input
  await page.focus('input#searchboxinput');
  await page.keyboard.down('Control');
  await page.keyboard.press('A');
  await page.keyboard.up('Control');
  await page.keyboard.press('Backspace');
  await page.type('input#searchboxinput', item);
  await page.keyboard.press('Enter');
  await page.waitForTimeout(3000);

  let load = await page.waitForSelector('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd', { timeout: 60000 })
  log(load)

  await page.waitForSelector('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd', { timeout: 60000 }).then(async () => {

  try {
    await autoScroll(page);
    log.debug('autoScroll single done')

  } catch (error) {
    log.error('scroll error \n', error)
  }

  if (await page.$(".lcr4fd.S9kvJb") !== null) {
    log.debug('has shortcards')

    const cardsData = await page.evaluate(() => {

      let cardsArray = []

document.querySelectorAll('.Nv2PK.THOPZb').forEach(function (elem) {
        const output = {}

        output.name = elem.querySelector('.NrDZNb').textContent.replace(/,/g, ".")

        let activity = elem.querySelector(".Z8fK3b .W4Efsd .W4Efsd:nth-child(1) span:nth-child(1) span")
        if (activity !== null) {
          output.activity = activity.textContent.replace(/,/g, ".")
        } else {
          output.activity = "null"
```

```
            }

            let company_address = elem.querySelector(".Z8fK3b
.W4Efsd .W4Efsd:nth-child(1) span:nth-child(2)")
            if (company_address !== null) {
              output.address                                    =
company_address.textContent.replace(/,/g, ".")
            } else {
              output.address = "null"
            }

            let phone = elem.querySelector(".Z8fK3b .UaQhfb
.W4Efsd .W4Efsd .UsdlK")

            if (phone !== null) {

              // for (let i = 0; i < phone.length; i++) {
              //   const element = phone[i].innerText;
              //   if(element.indexOf("+") >= 0) {
                    output.phone = phone.innerText
                // }
              // }
            } else {
              output.phone = "null"
            }

            let site_link = elem.querySelector(".Rwjeuc a")
            if (site_link !== null) {
              output.site = site_link.href
            } else {
              output.site = "null"
            }

            cardsArray.push(output)
          })

          return cardsArray

        });

        await parseResultWriter.writeRecords(cardsData)
        log.warn(cardsData)
        log.debug('done single shortcard')

      }
      else {
        log('standart flow');

        const URIs = await page.evaluate(() => {
          const links = document.querySelectorAll(".hfpxzc");
          let linksURIs = [];

          links.forEach((el) => {
            linksURIs.push({ "URL": el.href });
```

```
      });
      return linksURIs;
    });

    URIs.forEach(el => {
      parsedPlacesLinks.push(el["URL"]);
    })

    await csvWriter.writeRecords(URIs)
  };



  log.info("done single")
}).catch(e => {
  log('FAIL');
  log.error(`fail on ${item} query,\n`, e)
});}
```

Додаток Б

Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедиміната (Литва)
Наукове товариство ім. Т.Шевченка

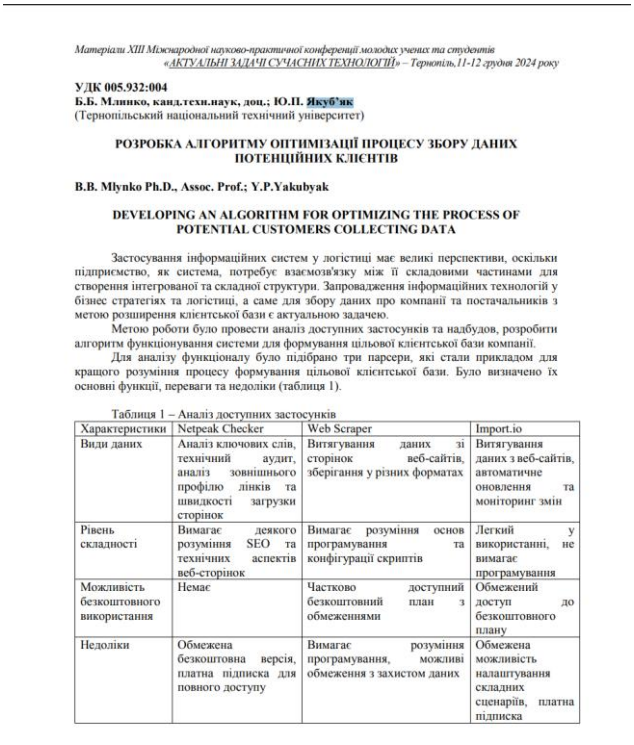
АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XIII Міжнародної науково-практичної конференції молодих учених та студентів
11-12 грудня 2024 року

УКРАЇНА
ТЕРНОПІЛЬ – 2024

Матеріали XIII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль,11-12 грудня 2024 року



Матеріали XIII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль,11-12 грудня 2024 року

УДК 005.932:004

Б.Б. Млинко, канд.техн.наук, доц.; Ю.П. Якуб'як
(Тернопільський національний технічний університет)

РОЗРОБКА АЛГОРИТМУ ОПТИМІЗАЦІЇ ПРОЦЕСУ ЗБОРУ ДАНИХ ПОТЕНЦІЙНИХ КЛІЄНТІВ

B.B. Mlynko Ph.D., Assoc. Prof.; Y.P.Yakubyak

DEVELOPING AN ALGORITHM FOR OPTIMIZING THE PROCESS OF POTENTIAL CUSTOMERS COLLECTING DATA

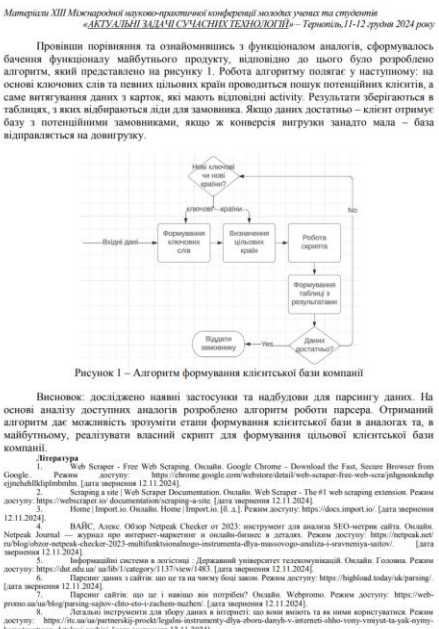
Застосування інформаційних систем у логістиці має великі перспективи, оскільки підприємство, як система, потребує взаємозв'язку між її складовими частинами для створення інтегрованої та складної структури. Запровадження інформаційних технологій у бізнес стратегіях та логістиці, а саме для збору даних про компанії та постачальників з метою розширення клієнтської бази є актуальною задачею.

Метою роботи було провести аналіз доступних застосунків та надбудов, розробити алгоритм функціонування системи для формування цільової клієнтської бази компанії.

Для аналізу функціоналу було підібрано три парсери, які стали прикладом для кращого розуміння процесу формування цільової клієнтської бази. Було визначено їх основні функції, переваги та недоліки (таблиця 1).

Таблиця 1 – Аналіз доступних застосунків

Характеристики	Netpeak Checker	Web Scraper	Import.io
Види даних	Аналіз ключових слів, технічний аудит, аналіз зовнішнього профілю лінків та швидкості загрузки сторінок	Витягування даних зі сторінок веб-сайтів, зберігання у різних форматах	Витягування даних з веб-сайтів, автоматичне оновлення та моніторинг змін
Рівень складності	Вимагає деякого розуміння SEO та технічних аспектів веб-сторінок	Вимагає розуміння основ програмування та конфігурації скриптів	Легкий у використанні, не вимагає програмування
Можливість безкоштовного використання	Немає	Частково доступний безкоштовний план з обмеженнями	Обмежений доступ до безкоштовного плану
Недоліки	Обмежена безкоштовна версія, платна підписка для повного доступу	Вимагає розуміння програмування, можливі обмеження з захистом даних	Обмежена можливість налаштування складних сценаріїв, платна підписка



Матеріали XIII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль,11-12 грудня 2024 року

Провівши порівняння та ознайомившись з функціоналом аналогів, сформувалось бачення функціоналу майбутнього продукту, відповідно до цього було розроблено алгоритм, який представлено на рисунку 1. Робота алгоритму полягає у наступному: на основі ключових слів та певних цільових країн проводиться пошук потенційних клієнтів, а саме витягування даних з карток, які мають відповідні activity. Результати зберігаються в таблицях, з яких відбираються ліди для замовника. Якщо даних достатньо – клієнт отримує базу з потенційними замовниками, якщо ж конверсія вигрузки занадто мала – база відправляється на довигрузку.

Рисунок 1 – Алгоритм формування клієнтської бази компанії

Висновок: досліджено наявні застосунки та надбудови для парсингу даних. На основі аналізу доступних аналогів розроблено алгоритм роботи парсера. Отриманий алгоритм дає можливість зрозуміти етапи формування клієнтської бази в аналогах та, в майбутньому, реалізувати власний скрипт для формування цільової клієнтської бази компанії.

Література

1. Web Scraper - Free Web Scraping. Онлайн. Google Chrome - Download the Fast, Secure Browser from Google. Режим доступу: https://chrome.google.com/webstore/detail/web-scraper-free-web-scra/jnhgnonknehpejjnehehllkliplmbmhn. [дата звернення 12.11.2024].
2. Scraping a site | Web Scraper Documentation. Онлайн. Web Scraper - The #1 web scraping extension. Режим доступу: https://webscraper.io/ documentation/scraping-a-site. [дата звернення 12.11.2024].
3. Home | Import.io. Онлайн. Home | Import.io. [б. д.]. Режим доступу: https://docs.import.io/. [дата звернення 12.11.2024].
4. ВАЙС, Алекс. Обзор Netpeak Checker от 2023: инструмент для анализа SEO-метрик сайта. Онлайн. Netpeak Journal — журнал про интернет-маркетинг и онлайн-бизнес в деталях. Режим доступу: https://netpeak.net/ru/blog/obzor-netpeak-checker-2023-multifunktsionalnogo-instrumenta-dlya-massovogo-analiza-i-sravneniya-saitov/. [дата звернення 12.11.2024].
5. Інформаційні системи в логістиці : Державний університет телекомунікацій. Онлайн. Головна. Режим доступу: https://dut.edu.ua/ ua/lib/1/category/1137/view/1483. [дата звернення 12.11.2024].
6. Парсинг данних з сайтів: що це та на чисму вони закон. Режим доступу: https://highload.today/uk/parsing/. [дата звернення 12.11.2024].
7. Парсинг сайтів: що це і навіщо він потрібен? Онлайн. Webpromo. Режим доступу: https://web-promo.ua/ua/blog/parsing-sajtov-chto-eto-i-zachem-nuzhen/. [дата звернення 12.11.2024].
8. Легальні інструменти для збору даних в інтернет: що вони вміють та як ними користуватися. Режим доступу: https://itc.ua/ua/partnerskij-proekt/legalni-instrumenty-dlya-zboru-danyh-v-interneti-shho-vony-vmiyut-ta-yak-nymy-korystuvatysya-detalnyj-rozbir/. [дата звернення 12.11.2024].

Рисунки Б.1-4 – Апробація роботи

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VIII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

24-25 квітня 2025 р.

(збірник тез конференції)

Тернопіль 2025

УДК 005.932:004

Якуб'як Ю. – ст. гр. СНнм-61
Тернопільський національний технічний університет ім. Івана Пулюя

РЕАЛІЗАЦІЯ АЛГОРИТМУ ОПТИМІЗАЦІЇ ПРОЦЕСУ ЗБОРУ ДАНИХ ПОТЕНЦІЙНИХ КЛІЄНТІВ

Науковий керівник: Б.Б. Млинко, канд.техн.наук, доц.

Yakubyak Y.
Ternopil Ivan Puluj National Technical University

REALISATION AN ALGORITHM FOR OPTIMIZING THE PROCESS OF POTENTIAL CUSTOMERS COLLECTING DATA

Supervisor: B.B. Mlynko Ph.D., Assoc. Prof.

Ключові слова: парсинг, оптимізація, алгоритм.
Keywords: parsing, optimization, algorithm.

Інформаційні системи на сьогоднішній день мають великі перспективи практично в кожній сфері сучасного життя, зокрема в логістиці. Якщо розглядати підприємство з точки зору організованої системи, то всі її складові частини потребують безперебійного зв'язку між собою для створення інтегрованої та складної структури. Впровадження інформаційних технологій в логістиці та бізнес стратегіях, безпосередньо для збору даних про потенційних покупців з метою розширення клієнтської бази є актуальною задачею.

Метою роботи було, на основі сформованого алгоритму[1] реалізувати скрипт, який надасть можливість парсити дані з застосунку Google Maps.

Для розробки скрипта за основу використаємо наступний алгоритм (рисунок 1):

1. Початковим кроком є запуск консолі;
2. Далі вводиться команда для запуску скрипта, ключові слова в лапках, цільова країна, а також кількість карток, які будуть витягнути за один запит;
3. Скрипт формує масив запитів з міст цільової країни та ключових слів;
4. Запити з масиву проганяються в Google Maps;
5. Перевіряється чи запит містить звичайні картки чи короткі картки;
6. Якщо запит містить звичайні картки, їх записується в окремий файл, посилання з якого потім будуть вигружені і прогнані в Google Maps, після чого отримані дані будуть записані в кінцевому файлі;
7. Якщо запит містить короткі картки, їх записується в кінцевий файл.

Розробляємо скрипт, який буде реалізовувати пошук компаній по заданих ключових словах та країні. Тобто користувач вводить команду node parser.js, після цього в лапках вводить ключові слова, за якими буде проводитися пошук компаній, а потім буквений код країни, по якій буде проведено пошук. Дані формують масив запитів, які будуть введені в графі пошуку.

На виході отримуємо скрипт, який вирішує наступні задачі (рисунок 2):

1. Створення масиву запитів для перебору;
2. Перевірка даних на валідність;
3. Запуск роботи скрипта;

4. Автопрокручування сторінки;
5. Обробка посилань.

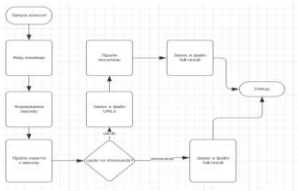


Рисунок 1 – Алгоритм роботи скрипта

Рисунки 2 – Процес роботи скрипта та зібрані дані

Рисунки Б.5-8 – Апробація роботи

Додаток В

```
const { exit } = require('process');
const log = require('cllc')();
const axios = require('axios');
const puppeteer = require('puppeteer');
const places = require('./countries');
const fs = require('fs');

const createCsvWriter = require('csv-
writer').createObjectCsvWriter;
const csvWriter = createCsvWriter({
  path: 'URLs.csv',
  header: ['URL']
});

const parseResultWriter = createCsvWriter({
  path: 'full-result.csv',
  header: [
    { id: 'name', title: 'name' },
    { id: 'activity', title: 'activity' },
    { id: 'address', title: 'address' },
    { id: 'phone', title: 'phone' },
    { id: 'site', title: 'site' },
  ]
});


const arguments = process.argv.slice(2)
if (arguments.length == 0 || arguments.length > 3) {
  log.e('arguments empty or has a wrong format');
  exit();
}
if
(!places.countriesList.hasOwnProperty(arguments[1].toUpperCase()))
{
  log.e('you have entered nonexistent country shortname');
  exit();
}

if (arguments[2] && !Number.isInteger(+arguments[2])) {
  log.e('third argument is not a number');
  exit();
}
const scrollCardsAmount = arguments[2] || 40;

// масив запитів для перебору
const queriesArray = places.makeQueriesArray(arguments)
log(queriesArray)

let parsedPlacesLinks = []
let currentIteratorStep = 0;
```

```
startParse()



async function startParse() {
  const browser = await puppeteer.launch({ headless: false });
  const page = await browser.newPage();
  page.setViewport({ width: 1280, height: 600 });

  await page.goto("https://www.google.com.ua/maps/?hl=en");
  await page.waitForSelector('input#searchboxinput', {
timeout: 5000 })

  for (const currentQuery of queriesArray) {
    try {
      await handleSingleQuery(page, currentQuery);
    } catch (error) {
      log.error(`throuble on ${currentQuery},\n`, error)
    }
    log(currentQuery)
  }
  log.debug('done all scrolling');
  await browser.close();

  parseLinks(placesLinksIterator(parsedPlacesLinks))
  log.start(`lines total: ${parsedPlacesLinks.length}\nlines
done [%s]`, 1);

}



async function handleSingleQuery(page, item) {
  // handle keyboard input
  await page.focus('input#searchboxinput');
  await page.keyboard.down('Control');
  await page.keyboard.press('A');
  await page.keyboard.up('Control');
  await page.keyboard.press('Backspace');
  await page.type('input#searchboxinput', item);
  await page.keyboard.press('Enter');
  await page.waitForTimeout(3000);


  let load = await
page.waitForSelector('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd', {
timeout: 60000 })
  log(load)

    await
page.waitForSelector('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd', {
timeout: 60000 }).then(async () => {
```

```
          try {
            await autoScroll(page);
            log.debug('autoScroll single done')

          } catch (error) {
            log.error('scroll error \n', error)
          }

          if (await page.$(".lcr4fd.S9kvJb") !== null) {
            log.debug('has shortcards')

            const cardsData = await page.evaluate(() => {

              let cardsArray = []

document.querySelectorAll('.Nv2PK.THOPZb').forEach(function (elem)
{
                const output = {}

                output.name =
elem.querySelector('.NrDZNb').textContent.replace(/,/g, ".")

                let activity = elem.querySelector(".Z8fK3b .W4Efsd
.W4Efsd:nth-child(1) span:nth-child(1) span")
                if (activity !== null) {
                  output.activity =
activity.textContent.replace(/,/g, ".")
                } else {
                  output.activity = "null"
                }

                let company_address = elem.querySelector(".Z8fK3b
.W4Efsd .W4Efsd:nth-child(1) span:nth-child(2)")
                if (company_address !== null) {
                  output.address =
company_address.textContent.replace(/,/g, ".")
                } else {
                  output.address = "null"
                }

                let phone = elem.querySelector(".Z8fK3b .UaQhfb
.W4Efsd .W4Efsd .UsdlK")

                if (phone !== null) {

                  // for (let i = 0; i < phone.length; i++) {
                  //   const element = phone[i].innerText;
                  //   if(element.indexOf("+") >= 0) {
                      output.phone = phone.innerText
                    // }
                  // }
                } else {
                  output.phone = "null"
```

```
            }

            let site_link = elem.querySelector(".Rwjeuc a")
            if (site_link !== null) {
              output.site = site_link.href
            } else {
              output.site = "null"
            }

            cardsArray.push(output)
          })

          return cardsArray

        });

        await parseResultWriter.writeRecords(cardsData)
        log.warn(cardsData)
        log.debug('done single shortcard')

      }
      else {
        log('standart flow');

        const URIs = await page.evaluate(() => {
          const links = document.querySelectorAll(".hfpxzc");
          let linksURIs = [];

          links.forEach((el) => {
            linksURIs.push({ "URL": el.href });
          });
          return linksURIs;
        });

        URIs.forEach(el => {
          parsedPlacesLinks.push(el["URL"]);
        })

        await csvWriter.writeRecords(URIs)
      };



      log.info("done single")
    }).catch(e => {
      log('FAIL');
      log.error(`fail on ${item} query,\n`, e)
    });

}



async function autoScroll(page) {
```

```
      log('start autoscrolling');

      let endOfListSelector = await page.$('.PbZDve');
      if (!!endOfListSelector) {
        log('end selector')
        return true;
      }


      let placesCards = await page.$$('.hfpxzc');

      if (placesCards.length > scrollCardsAmount) {
        log('too much places cards')
        return true;
      }

      // log.warn(placesCards.length)

      new Promise(r => setTimeout(r, 500));

      await page.evaluate(() => {
        const scrollElem =
document.querySelectorAll('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd')[1
];
        scrollElem.scrollTop = scrollElem.scrollTop - 300;
      })

      await page.waitForNetworkIdle();

      await page.evaluate(() => {

document.querySelectorAll('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd')[1
].scrollTop = 100000
      })

      await page.waitForNetworkIdle();

      await autoScroll(page)

    }



    function placesLinksIterator(array) {
      if (currentIteratorStep === array.length) {
        log.warn('process finished succesfully');
        return false;
      }

      ++currentIteratorStep

      log.step(1);
      return array[currentIteratorStep - 1]
    }
```

```
    async function parseLinks(url) {
      if (url === false) {
        log.info('exit');
        exit();
      }

      await axios(url)
        .then(async response => {
          log(response.status)

          let currentResult = [handleResponseData(response.data)];

          await parseResultWriter.writeRecords(currentResult)
          // log.info(currentResult)
          log.debug('current line written')
        })
        .catch(error => {
          log.error(error)
        });


      await parseLinks(placesLinksIterator(parsedPlacesLinks))
    }

    function handleResponseData(data) {
      const currentLine = {}


      const infoStrEnd = data.indexOf('itemprop="name">');
      const infoStrStart = data.lastIndexOf('<meta content=',
infoStrEnd);
      let infoStr = data.slice(infoStrStart + 15, infoStrEnd - 2);

      if (infoStr.indexOf(" · ") > 0) {
        infoStr = infoStr.split(' · ');
        currentLine.name = infoStr[0].replace(',', '');
        currentLine.address = infoStr[1].replace(',', '');
      } else {
        currentLine.name = infoStr.replace(',', '');
        currentLine.address = "null";
      }



      const activityStrEnd =
data.indexOf('itemprop="description">');
      const activityStrStart = data.lastIndexOf('<meta content=',
activityStrEnd)

      if (activityStrEnd !== -1) {
        let activity = data.slice(activityStrStart + 15,
activityStrEnd - 2);
        if (activity.indexOf(' · ') > 0) {
```

```
          activity = activity.slice(8)
        }

        currentLine.activity = activity.replace(',', '');
      } else {
        currentLine.activity = 'null';
      }


      const phoneStrStart = data.indexOf('tel:')
      const phoneStrEnd = data.indexOf('",', phoneStrStart);
      if (phoneStrStart !== -1) {
        currentLine.phone = data.slice(phoneStrStart + 4,
phoneStrEnd - 1);
      } else {
        currentLine.phone = 'null';
      }


      const httpRegexG = /https?:\/\/(?:www\.)?[-a-zA-Z0-
9@:%._\+~#=]{1,256}\.[a-zA-Z0-9()]{1,6}\b(?:[-a-zA-Z0-
9()@:%_\+.~#?&\/=]*)/g;

      let linksArray = data.match(httpRegexG);

      let result = linksArray.filter(link => {
        if (link.indexOf("google") === -1 &&
link.indexOf("gstatic") === -1 && link.indexOf("ggpht") === -1 &&
link.indexOf("schema") === -1 && link.indexOf("broofa") === -1) {
          return true
        } else {
          return false
        }
      });
      result = new Set(result);

      if (result.size === 0) {
        currentLine.site = 'null';
      } else {
        currentLine.site = Array.from(result)[0];
      }

      return currentLine
    }
```