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

бакалавр
(назва освітнього ступеня)

на тему: Аналіз програмно-алгоритмічних засобів для оптимізації логістичних потоків

Виконав: студент IV курсу, групи СНс-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

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

Керівник (підпис) Шимчук Г.В.
(прізвище та ініціали)

Нормоконтроль (підпис) Литвиненко Я.В.
(прізвище та ініціали)

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

Рецензент (підпис) Тиш Є.В.
(прізвище та ініціали)

Тернопіль
2023

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)
«__» _переддень захисту_ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(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 логістичних потоків

Керівник роботи Шимчук Григорій Валерійович, старший викладач кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-133
2. Термін подання студентом завершеної роботи Дата захисту 2023р.
3. Вихідні дані до роботи Літературні та інтернет джерела про методи та програмно-алгоритмічні засоби для оптимізації логістичних потоків

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1.Аналіз літературних джерел. 1.1. Web Scraper. 1.2. Import.io. 1.3. Netpeak Checker. 1.4. Висновки до розділу 1. 2. Проєктування та реалізація програмного рішення. 2.1. Вибір технології розробки. 2.2. Розробка скрипта для парсингу. 2.3. Тестування продукту. 2.4. Висновки до розділу 2. 3. Безпека життєдіяльності, основи охорони праці. 3.1. Дія електричного струму на організм людини, види електротравм. 3.2. Вимоги безпеки до ПК та устаткування. 3.3. Висновок до третього розділу Висновки. Перелік використаних джерел. Додаток А. Додаток Б.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Повний перелік слайдів у презентації, включаючи перший та останній слайди (з номерами, через крапку).
1. Титульний слайд. 2. Актуальність теми. 3. Мета. 4. Завдання. 5.Аналіз літературних джерел. 6.Алгоритм формування цільової клієнтської бази для замовника. 7. Алгоритм роботи скрипта. 8. Середовище та технології розробки. 9. Запуск скрипта. 10. Робота скрипта. 11. Збір даних. 12. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	23.01.2023	*Виконано*
2.	Підбір джерел про автоматизацію логістичних потоків	24.01.2023-26.01.2023	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	27.01.2023-31.01.2023	*Виконано*
4.	Виконання дослідження щодо автоматизації	01.02.2023-07.02.2023	*Виконано*
	логістичних потоків		
	Розроблення скрипта для парсингу Google Maps		
5.	Оформлення розділу « Аналіз літературних джерел»	08.02.2023-09.02.2023	*Виконано*
6.	Оформлення розділу «Проєктування та реалізація	10.02.2023-12.02.2023	*Виконано*
	програмного рішення		
7.	Виконання завдання до підрозділу «Безпека	05.06.2023-08.06.2023	*Виконано*
	життєдіяльності та основи охорони праці»		
8.	Оформлення кваліфікаційної роботи	09.06.2023-11.06.2023	*Виконано*
9.	Нормоконтроль	12.06.2023-13.06.2023	*Виконано*
10.	Перевірка на плагіат	14.06.2023	*Виконано*
11.	Попередній захист кваліфікаційної роботи	15.06.2023	*Виконано*
12.	Захист кваліфікаційної роботи	19.06.2023	

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

Керівник роботи _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Аналіз програмно-алгоритмічних засобів для оптимізації логістичних потоків // Кваліфікаційна робота освітнього рівня «Бакалавр» // Якуб'як Юліяна Павлівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-41 // Тернопіль, 2023 // С. 55, рис. – 23 , табл. – 1, кресл. – 0, додат. – 2, бібліогр. – 19.

Ключові слова: парсинг, парсер, розробка, Node.js, скрипт, логістика, Git Bash, клієнти, ключові слова.

Кваліфікаційна робота присвячена дослідженню парсингових систем для пошуку клієнтської бази закордоном. В першому розділі кваліфікаційної роботи описано парсингові системи. Висвітлено їх переваги та недоліки. Розглянуто основний функціонал та особливості. Проведено порівняння розглянутих засобів.

В другому розділі кваліфікаційної роботи обрано інструментарій та технології, які будуть використані в розробці. Досліджено алгоритми формування цільової клієнтської бази та роботи скрипта. Подано лістинги, в яких наведено коди деяких функцій скрипта. Проведено тестування отриманого кінцевого продукту.

ANNOTATION

Software and Algorithmic Tools Analysis for Optimisation of Logistics Flows // Qualification work of the educational level "Bachelor" // Yakubiak Yuliiana // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-41 // Ternopil, 2023 // P. 55, fig. - 23, tabl. - 1, draw. - 0, annexes. – 2, references - 17.

Keywords: parsing, parsing, development, Node.js, script, logistics, Git Bash, customers, keywords.

The qualification work is devoted to exploring of parsing systems for finding a client base abroad. The first section of the qualification work describes parsing systems. Their advantages and disadvantages are highlighted. The main functionality and features are considered. A comparison of the considered means is carried out.

In the second section of the qualification work, the tools and technologies that will be used in the development are selected. The algorithms of forming the target client base and script operation are investigated. Filled listings which show codes for some of the script's functions. The resulting final product was tested.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface.

CSS – це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду.

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.

JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript.

Point-and-click – один з методів керування графічним інтерфейсом користувачем, що полягає у наведенні курсора на активну область та натиснення кнопки на цю область.

Selector – це бібліотека/інструмент, який дозволяє вибирати значення з об'єкта JSON (текстового формату обміну даними між комп'ютерами) за допомогою певного синтаксису запиту.

SEO – процес коригування HTML-коду, текстового наповнення, структури сайту, контроль зовнішніх чинників для відповідності вимогам алгоритму пошукових систем, з метою підняття позиції сайту в результатах пошуку в цих системах за певними запитами користувачів.

Sitemap – це XML-файл з інформацією для пошукових систем про сторінки веб-сайту, які підлягають індексації.

URL – уніфікований локатор ресурсів або адреса ресурсу – стандартизована адреса певного ресурсу в інтернеті.

ЗМІСТ

ВСТУП

Актуальність теми: Інформаційні ресурси є надзвичайно важливими для підприємств і грають ключову роль у логістичних операціях. Інформація визначає потреби об'єктів логістичних систем і різних ланок ланцюгів поставок. Основна мета обміну інформацією полягає в узгодженні вимог різних суб'єктів щодо розміру замовлень, наявності запасів і швидкості переміщення ресурсів.

Інформаційний потік є невід'ємною частиною загального логістичного потоку і повинен достовірно відображати реальну практичну діяльність у галузях фізичного розподілу, виробництва та матеріально-технічного постачання.

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. Тому сучасний логіст повинен бути знайомий і вміти використовувати новітні інформаційні технології у своїй повсякденній роботі. Зростаючий розвиток і широке застосування обчислювальної техніки ставлять вимоги до сучасних логістів, які повинні аналізувати складні логістичні процеси на підприємстві за допомогою впровадження та використання інформаційних систем.

Об’єкт дослідження: система для збору даних про клієнтів закордоном.

Мета роботи полягає у розробці та впровадженні системи для пошуку та збору клієнтської бази.

Завдання роботи:

1. Визначити основні поняття системи парсингу даних.
2. Дослідити структуру та компоненти систем для парсингу.
3. Проаналізувати принципи роботи існуючих систем.
4. Дослідити специфіку та особливості.
5. Розробити скрипт для парсингу даних, який буде мати за мету вирішення задачі парсингу гугл карт за допомогою ключових слів та країн.

1. АНАЛІЗ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1 Web Scraper

Web Scraper – інструмент вилучення веб-даних із простим інтерфейсом «point-and-click» для сучасних веб-сайтів.

Безкоштовний і простий у використанні інструмент вилучення веб-даних для всіх. Завдяки простому інтерфейсу «point-and-click» можливість витягти тисячі записів із веб-сайту займає лише кілька хвилин налаштування Sitemap.

Web Scraper використовує модульну структуру, яка складається з селекторів, які інструктують скребок про те, як обійти цільовий сайт і які дані витягти. Завдяки такій структурі видобуток даних із сучасних і динамічних веб-сайтів, таких як Amazon, Tripadvisor, eBay, а також з менш відомих сайтів, є легким.

Вилучення даних запускається у вашому браузері не вимагає встановлення нічого на вашому комп'ютері. Вам не потрібен досвід кодингу Python, PHP або JavaScript, щоб почати скрепінг. Крім того, можна повністю автоматизувати вилучення даних у Web Scraper Cloud [1].

Для застосування даного інструменту необхідно встановити спеціальне розширення в Google Chrome. Після встановлення, виклик розширення відбувається з консолі браузера на сторінці, з якої буде проводитись витягування даних. Далі необхідно створити Sitemap. Перше, що потрібно зробити при створенні Sitemap - вказати початковий url. Це URL-адреса, з якої почнеться витягування даних. Ви також можете вказати кілька початкових URL-адрес, якщо скрепінг має починатися з кількох місць. Наприклад, якщо ви хочете очистити кілька результатів пошуку, ви можете створити окрему початкову URL-адресу для кожного результату пошуку. Додаткові поля введення URL-адреси можна додати, натиснувши + поруч із введеною URL-адресою. Вікно створення Sitemap зображено на рисунку 1.1. Після створення карти сайту початкову вкладку URL-адреси можна знайти, вибравши Редагувати метадані в

sitemap_name спадному меню Sitemap. Спадне меню маніпуляцій з Sitemap зображено на рисунку 1.2.

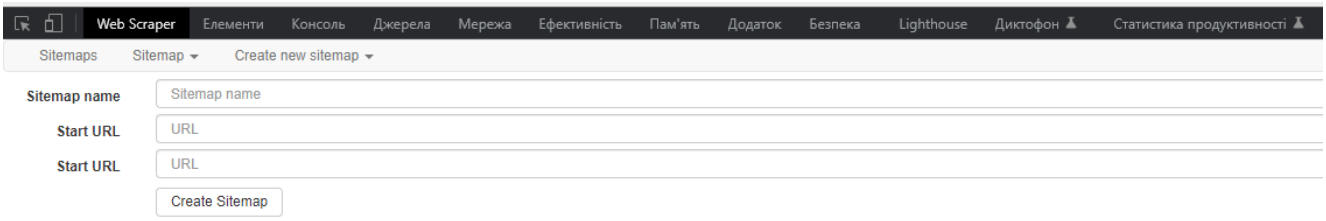

Рисунок 1.1 – Вікно створення Sitemap

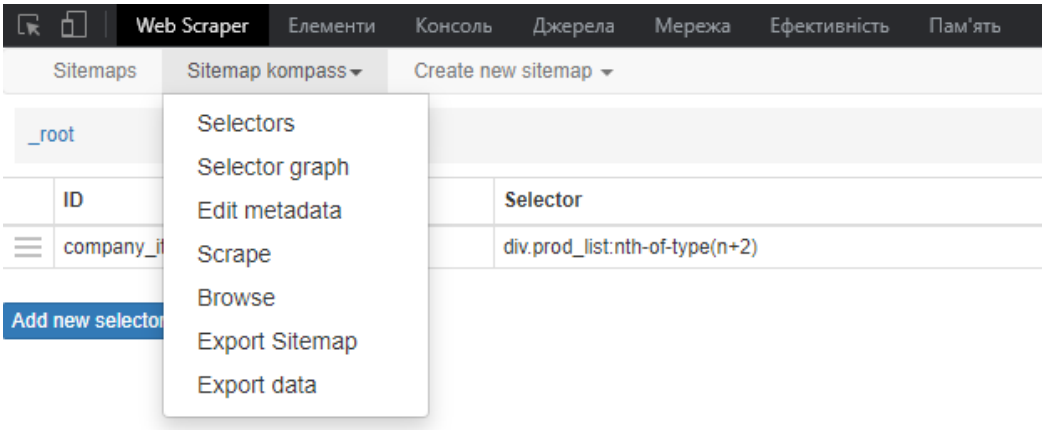

Рисунок 1.2 – Спадне меню маніпуляцій з Sitemap

Після створення карти сайту ви можете почати додавати, змінювати та переходити між селекторами на панелі "Selectors".

Селектори додаються в структуру деревовидного типу. Web Scraper виконає селектори в порядку, як вони організовані в деревоподібній структурі. Щоб стягнути дані з сайту, ви можете створити селектор посилань, який витягне всі посилання на статтю на першій сторінці. Потім, як дочірній селектор, ви можете додати селектор тексту, який витягуватиме статті зі сторінок статті, на які селектор посилань знайшов посилання.

Web Scraper має кілька селекторів, які можна використовувати для вилучення даних різного типу та для різної взаємодії з веб-сайтом. Селектори можна розділити на три групи: селектори вилучення даних для вилучення даних,

засоби вибору посилань для навігації сайтом та селектори елементів для виділення елементів, які розділяють кілька записів.

Селектори вилучення даних просто повертають дані з вибраного елемента. Наприклад, засіб виділення тексту видобуває текст із виділеного елемента. Ці селектори можна використовувати як селектори вилучення даних:

- виділення тексту;
- виділення посилань;
- селектор спливаючих посилань;
- виділення зображення;
- селектор таблиці;
- селектор атрибутів елементів;
- селектор HTML;
- згрупований селектор.

Селектори посилань витягують URL-адреси з посилань, які пізніше можна відкрити для вилучення даних. Наприклад, якщо в дереві карти сайту є селектор посилань, який має 3 дочірніх селектора тексту, тоді веб-скребок витягує всі URL-адреси за допомогою селектора посилань, а потім відкриває кожне посилання та використовує ці дочірні селектори вилучення даних для вилучення даних. Звичайно, селектор посилань може мати селектори посилань як дочірні селектори, тоді ці дочірні селектори посилань будуть використовуватися для подальшої навігації по сторінці. Наразі доступними селекторами посилань є виділення посилань та селектор спливаючих посилань.

Селектори елементів призначені для виділення елементів, які містять кілька елементів даних. Наприклад, селектор елементів може бути використаним для вибору списку елементів на сайті електронної комерції. Селектор поверне кожен елемент, що виділено як батьківський елемент дочірнім селекторам. Дочірні селектори елементів витягуватимуть дані лише в межах елемента, який їм надав селектор елемента.

Кожен селектор має параметри конфігурації. Тут можна побачити найпоширеніші з них. Параметри конфігурації, специфічні для селектора, описані в документації селекторів.

- selector - селектор CSS, який вибирає елемент, над яким селектор буде працювати;
- multiple - слід перевіряти, коли кілька записів (рядків даних) буде вилучено за допомогою цього селектора. Дані, видобуті з двох або більше селекторів із кількома прапорцями, не буде об'єднано в один запис;
- delay - затримка перед використанням селектора;
- parent selectors - налаштуйте батьківські селектори для цього селектора, щоб створити дерево селектора.

Web Scraper має інструмент вибору point and click, який робить виділення елементів доступним для всіх користувачів. Його можна відкрити, натиснувши кнопку «Select in selector creation interface». Він виділить елементи, які він вибере при натисканні жовтим кольором, а вже вибрані елементи будуть виділені червоним кольором. Виділення елементів можна скасувати, клацнувши на них ще раз, поки інструмент "Selector" все ще активний.

Після того, як ви створили селектори для Sitemap, ви можете приступати до скрепінгу. Відкрийте панель "Scrape" і почніть скрепінг. За бажанням можна змінити інтервал запиту і затримку завантаження сторінки. Відкриється нове спливаюче вікно, в якому Scraper буде завантажувати сторінки і витягувати з них дані. Після завершення скрепінгу спливаюче вікно закриється і ви отримаєте сповіщення спливаючим повідомленням. Ви можете переглянути витягнуті дані, відкривши панель "Перегляд", та експортувати їх, відкривши панель "Експортувати дані як CSV" [2].

1.2. Import.io

Import.io дозволяє витягувати дані безпосередньо з Інтернету. Це явище широко відоме як веб-скрейпінг, але Import.io може набагато більше. Використаний інтерфейс «point-and-click» перетворює веб-сайти на дані

кількома простими клацаннями, дозволяючи вам отримати необхідні дані, незалежно від того, чи вимагає це взаємодії зі сторінкою, JavaScript або лежить за логіном.

Import.io дозволяє створити екстрактор і надати йому приклад URL-адреси, що містить дані, які потрібно витягти. Після того, як Import.io завантажує веб-сторінку, вона надає вам дані, які вона знаходить, і дає вам можливість ідентифікувати дані, які ви хочете зібрати, за допомогою «point-and-click». Під час вибору даних Import.io аналізує базову структуру веб-сторінки та визначає, де розташовані елементи даних.

Після входу у свій обліковий запис ви потрапите на інформаційну панель Import.io. Почніть з натискання вкладки «EXTRACTORS» ліворуч, а потім натисніть «NEW EXTRACTOR» у верхній частині лівої панелі та вставте URL-адресу бізнес-сторінки у вікні «Create Extractor», натисніть «EXTRACT» щоб завантажити сторінку. Вікно створення екстрактора зображено на рисунку 1.3.

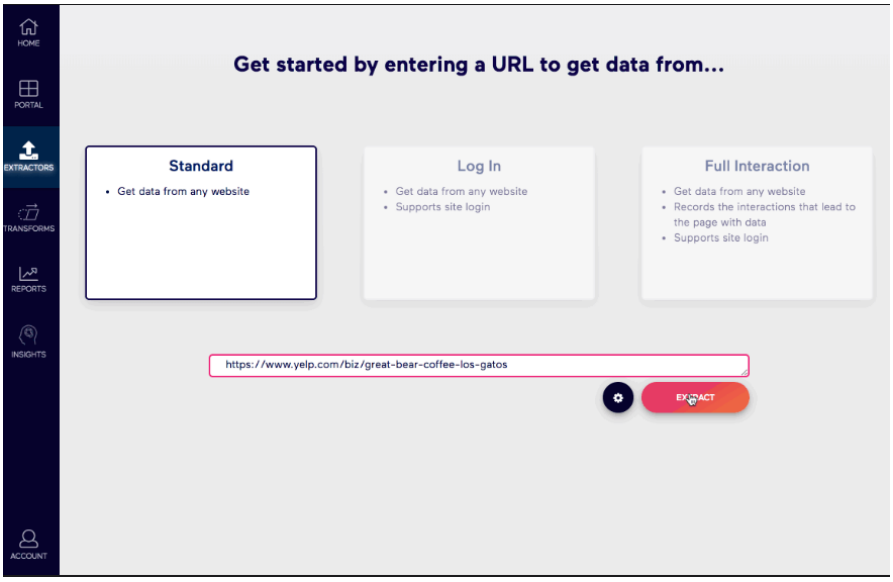

Рисунок 1.3 – Вікно створення нового екстрактора

Після завантаження сторінки Import.io спочатку спробуємо визначити списки або таблиці на сторінці. Щоб додати точку даних, можна натиснути на кнопку «+» , щоб додати стовпець, а потім клацнути на сторінці, щоб вибрати цю точку даних. Щоб назвати або перейменувати цей стовпець, двічі клацніть на

імені або скористайтеся текстовим полем у верхній правій частині сторінки. Вікно редагування точок даних зображено на рисунку 1.4.

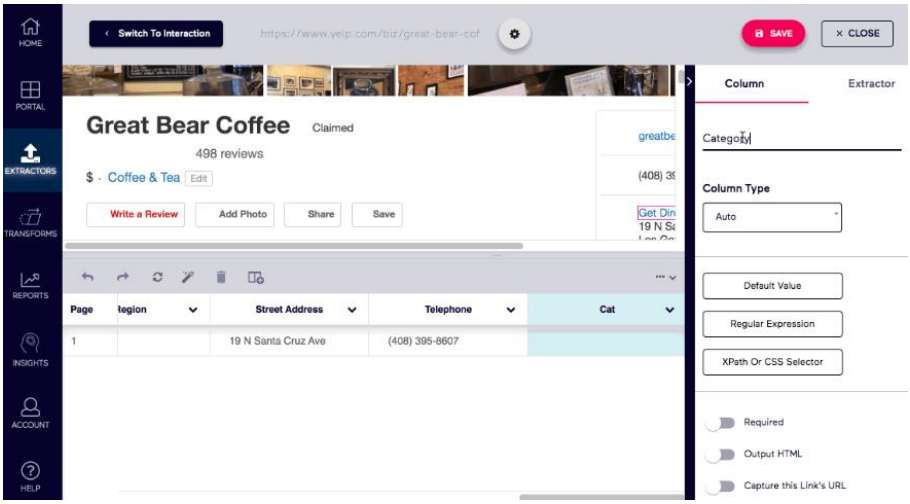

Рисунок 1.4 – Вікно редагування точок даних

Оскільки це видобувач деталей, можна обмежити повернення вибраних даних одним рядком на сторінку для видобутої сторінки, а не списком даних. Для цього відкрийте параметр «Advanced», потім виберіть «Rows» та перевірте, чи встановлено значення «Single Row». Вікно редагування видачі даних зображено на рисунку 1.5.

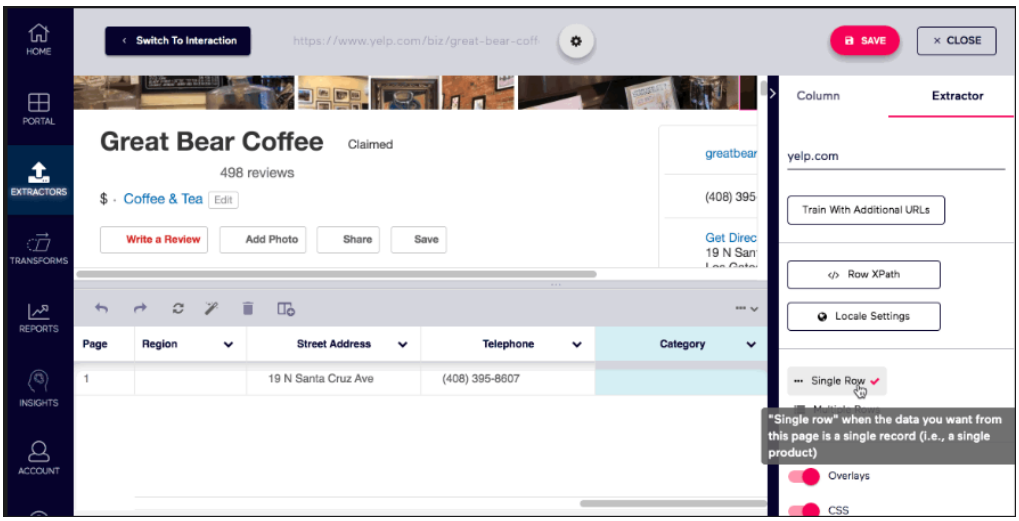

Рисунок 1.5 – Вікно редагування видачі даних

Вибравши точки даних, натисніть кнопку Зберегти, щоб зберегти екстрактор, потім натиснути «Save and run.» Після збереження екстрактора він

перенаправить вас на інформаційну панель, де ви зможете попередньо переглянути або завантажити результати запуску, коли він буде завершений.

За допомогою екстрактора деталей, ми можемо повернутися назад і додати приховані значення або очистити інші значення в Import.io, перш ніж отримати вихідні дані.

Можна натиснути кнопку «Edit», щоб повернутися до подання редагування екстрактора. Після додавання інших стовпців, можна додати значення, які мають бути спочатку, вимкнувши стиль сторінки. Для цього перейдіть до додаткових параметрів, виберіть «Page», а потім відключіть CSS. Стиль можна знову ввімкнути, знову ввімкнувши CSS.

Під час навчання екстрактора він збереже вигляд сторінки, коли ви вперше її створите. Вікно навчання екстрактора зображено на рисунку 1.6. Якщо сторінка зміниться, ви можете натиснути кнопку «Refresh» (1), щоб додати оновлену версію сторінки, і відредагувати за допомогою кнопки «Train With Additional URLs»(2).

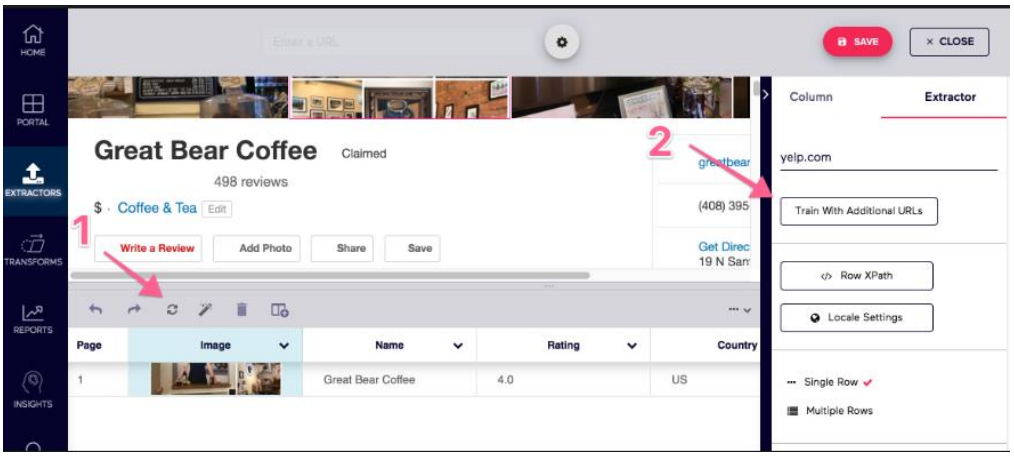

Рисунок 1.6 – Вікно навчання екстрактора

Тренування з додатковими URL-адресами відкриє екран, який дозволить вам додати одну або кілька сторінок для редагування, щоб перевірити навчання вашого екстрактора та покращити його, якщо потрібно. Додавши другу сторінку, ви можете переключатися між двома сторінками, щоб повторно вибрати точки даних, якщо потрібно. Екстрактор тепер використовуватиме обидві сторінки, щоб зрозуміти, яку точку даних ви вибираєте.

Окрім додавання стовпців даних, ви також можете редагувати наявні стовпці даних. Якщо натиснути стрілку випадаючого списку поруч із назвою стовпця, відкриється меню параметрів стовпця.

У настройках стовпця можна скористатися параметром «Set regular expression» для зіставлення та фільтрування рядка. Виділіть весь потрібний текст, потім перейдіть до «Set regular expression» і встановіть для параметра «Match» значення Google Pays+(.+), а для параметра «Replace» значення $1. Це відповідає рядку, який містить Google Pay, а потім виведе значення "Так" або "Ні" у стовпець даних. Потім ми можемо використати «Duplicate column» та змінити «Match» на Apple Pays+(.+), щоб створити окремий стовпець для Apple Pay.

Перш ніж зберегти екстрактор, ви можете перетягнути стовпці даних, щоб змінити порядок даних, і повернутися до перегляду даних, щоб перевірити результат. Завершивши, натисніть кнопку «Save and run» або «Save only» [3].

1.3. Netpeak Checker

Netpeak Checker є комплексним інструментом, призначеним для аналізу важливих SEO-метрик веб-сайтів. Ця програма допомагає користувачам отримати інформацію про власний сайт та конкурентів, що дозволяє приймати обґрунтовані рішення для поліпшення їхньої онлайн-присутності та залучення більшої кількості клієнтів.

Netpeak Checker є інструментом, створеним для допомоги користувачам у дослідженні важливих показників веб-сайтів і отриманні цінних даних для SEO-оптимізації. Він надає різноманітну інформацію, яка включає:

a) Відомості про стан On-Page сайту, такі як відповідь сервера та час відповіді, вміст тегів, таких як Title, Description, Robots, Canonical, Hreflang, доступність сторінок сайту у файлі robots.txt, кількість внутрішніх та зовнішніх посилань, наявність та кількість редіректів, а також URL кінцевого ланцюжка редіректу, вміст заголовків H1 - H6, розмір HTML сторінок.

b) Кількість символів та слів на сторінках сайту;

c) Парсинг контактних даних, зокрема кількість та значення електронної пошти, кількість та значення номерів телефонів, кількість та посилання на месенджери, такі як Telegram, WhatsApp, Viber, WeChat, кількість та посилання на соціальні мережі, такі як Facebook, Instagram, Linkedin, Twitter, YouTube, Pinterest.

d) Дані DNS, включаючи IP-адресу, країну та континент;

e) Інформація про домен з даними Whois, такі як дата реєстрації та закінчення домену, а також кореневий домен;

f) Дані відомих SEO-сервісів, таких як Serpstat, Ahrefs, Majestic, Moz, SimilarWeb, SEMrush;

g) Дані про трафік сайту, зокрема загальна кількість відвідувачів і розподіл трафіку за різними каналами (прямий, пошуковий, реферальний, соціальний, пошта тощо).

h) Дані індексації та результатів пошукових систем Google, Bing, Yahoo.

Всі вищезгадані метрики зручно зібрані в сайдбарі праворуч в інтерфейсі програми. Для отримання даних досить просто відзначити потрібні пункти. Вікно параметрів Netpeak Checker зображено на рисунку 1.7.

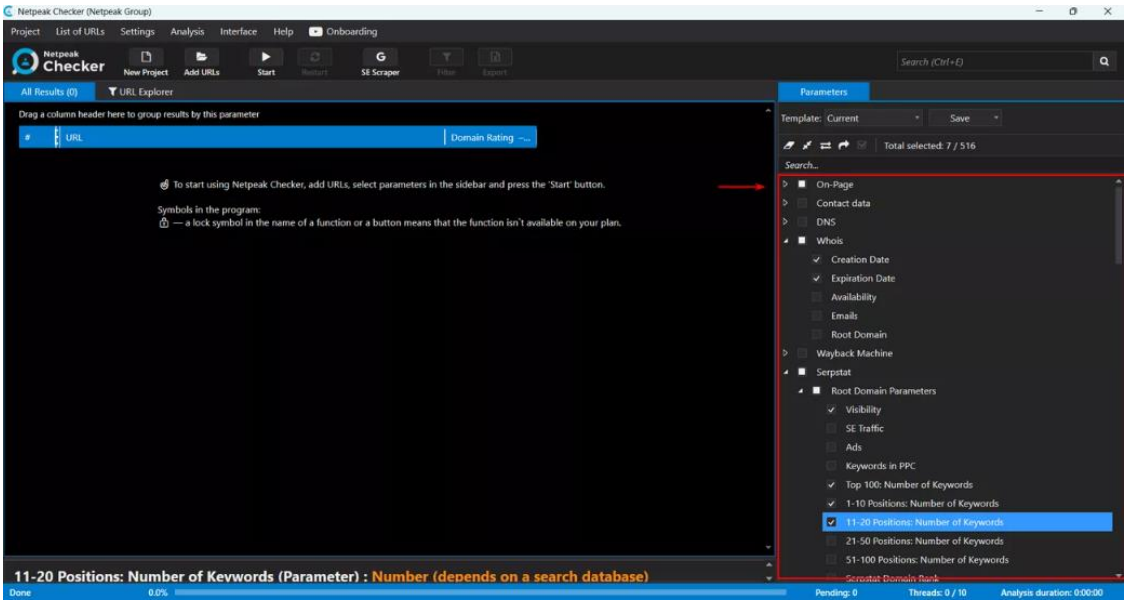

Рисунок 1.7 – Вікно параметрів застосунку

Інтерфейс Netpeak Checker зручний та інтуїтивно зрозумілий. Інтерфейс застосунку зображено на рисунку 1.8.

1. У верхній частині вікна знаходиться командна панель з основними операційними командами: створення, збереження та завантаження проектів, експорт та вставка URL сторінок сайту, експорт та імпорт даних, налаштування програми тощо;
2. Всі дані зручно поділені на вкладки з можливістю фільтрації;
3. Результати зручно представлені у табличному вигляді;
4. Праворуч у сайдбарі у зручному списку перераховані всі доступні для аналізу метрики;
5. Над сайдбаром знаходиться рядок пошуку.

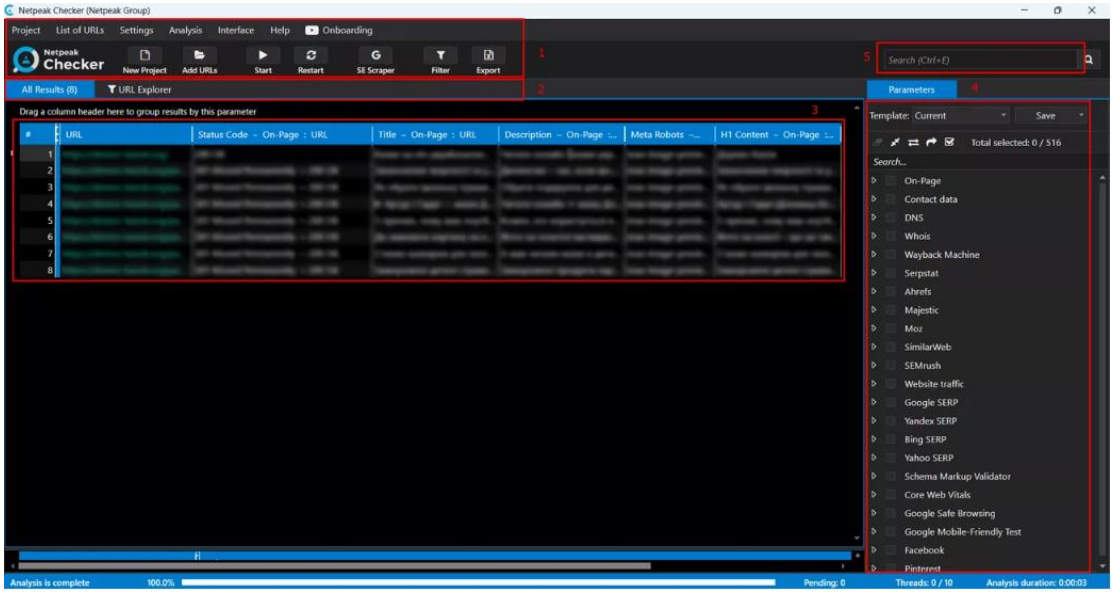

Рисунок 1.9 – Інтерфейс застосунку

Робота з даними відбувається за проектним принципом. Для кожного набору даних створюється окремий проект, який можна зберігати, експортувати та імпортувати. Також можна відкривати Netpeak Checker у кількох вікнах, що дозволяє працювати з кількома проектами одночасно. Усі операції з проектами доступні у вкладці «Project» командної панелі (зображеної на рисунку 1.10):

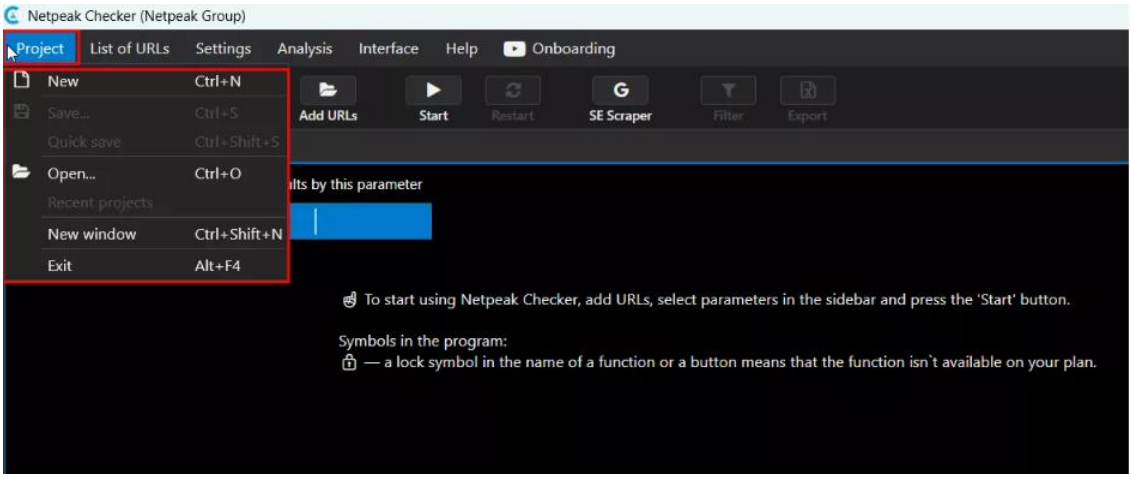

Рисунок 1.10 – Вкладка «Project»

Всі дані можна фільтрувати за допомогою операторів «Включити» та «Вимкнути», вибравши необхідні метрики та вказавши умови фільтрації. Для швидкого фільтрування даних можна використовувати рядок пошуку. Приклад фільтрування даних наведено на рисунку 1.11. Просто введіть туди метрику/елемент, що цікавить вас, і всі дані по ній автоматично відфільтруються (пошук відбувається по всій таблиці із метриками.):

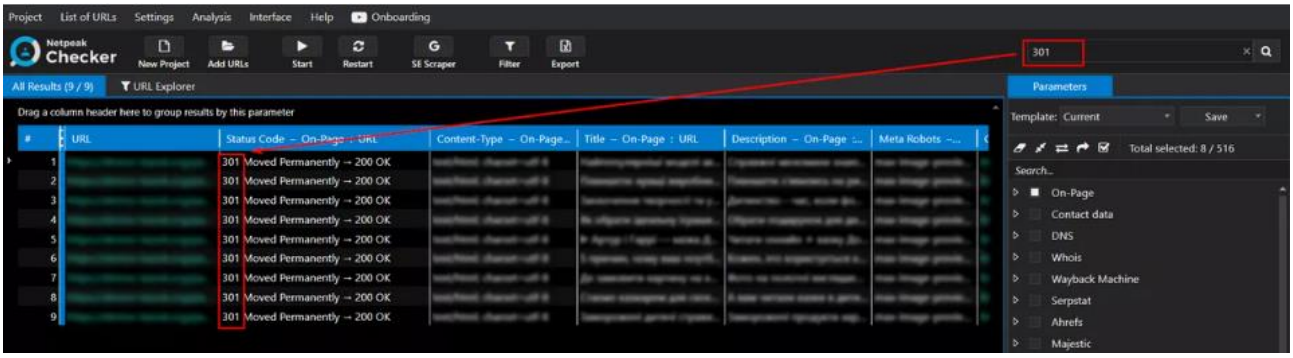

Рисунок 1.11 – Фільтрування даних

Слід окремо відзначити функціонал експорту даних до Google Drive, що дозволяє зберігати всі результати аналізу у форматі Google Sheets. Для цього достатньо авторизуватися у своєму обліковому записі Google і відзначити необхідний метод експорту в налаштуваннях програми. Налаштування експорту даних до Google Drive зображено на рисунку 1.12.

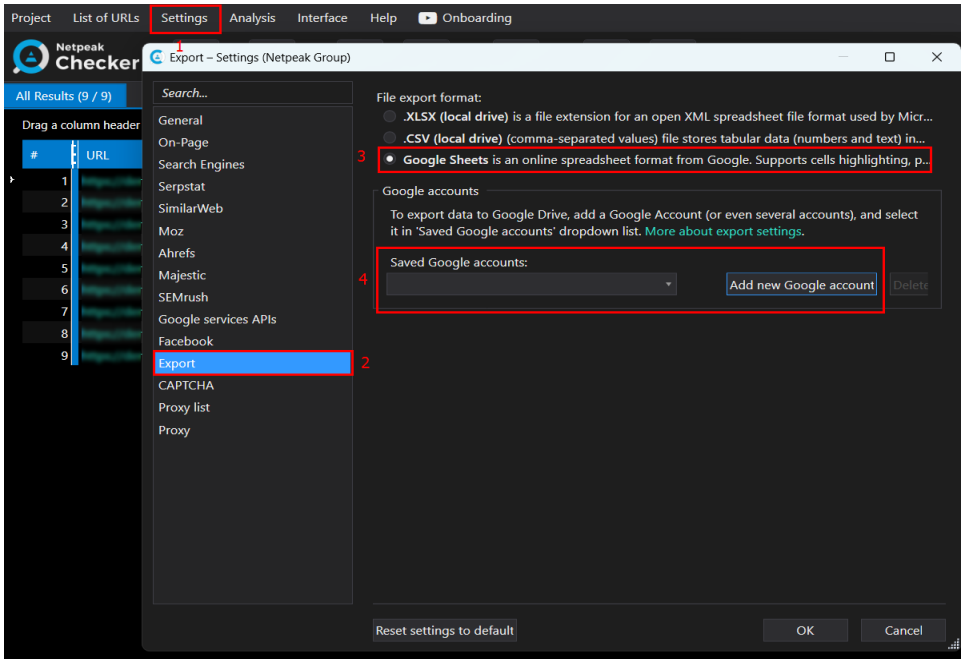

Рисунок 1.12 – Налаштування експорту даних до Google Drive

Ще одним важливим функціоналом програми є можливість парсингу пошукової видачі, що дозволяє за переліком ключових фраз спарсити результати топ-1, топ-3, топ-10 та топ-50 видачі пошукових систем Google, Bing та Yahoo. Для цього достатньо вставити (або імпортувати) потрібні ключові слова та налаштувати параметри пошукової системи. Усі результати пошукової видачі ви отримаєте у зручному табличному форматі, який можна експортувати для подальшого аналізу. Налаштування парсингу пошукової видачі зображено на рисунку 1.13.

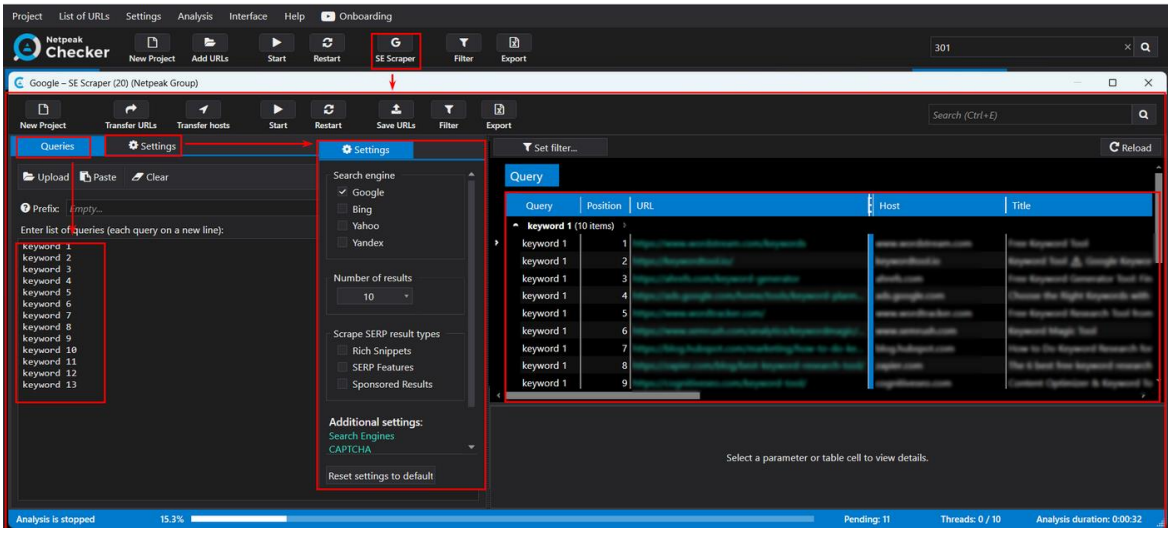

Рисунок 1.13 – Налаштування парсингу пошукової видачі

Для безперебійного парсингу даних із сайтів та результатів видачі пошукових систем Netpeak Checker надає можливість використовувати проксі-сервери та API популярних сервісів для автоматичного розгадування капчі. Для цього достатньо заповнити необхідні поля у налаштуваннях програми. Поля, необхідні для налаштування подано на рисунку 1.14.

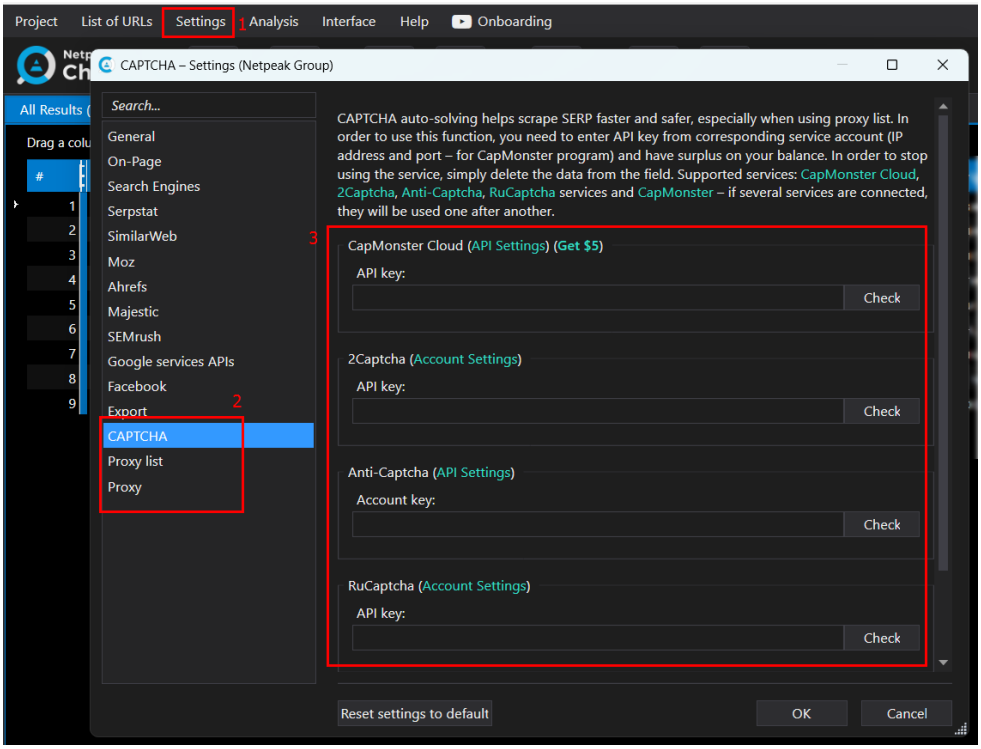

Рисунок 1.14 – Налаштування проксі

Netpeak Checker дозволяє передавати дані в Netpeak Spider , доповнюючи його звіти новими відомостями, що робить сайт більш комплексним та повним. Для цього достатньо експортувати дані Netpeak Checker у CSV форматі, а потім вибрати пункт «Upload parameters to enrich data…» у Netpeak Spider. Всі дані додадуться до метриків, що вже є в таблиці. Експорт даних відображено на рисунку 1.15.

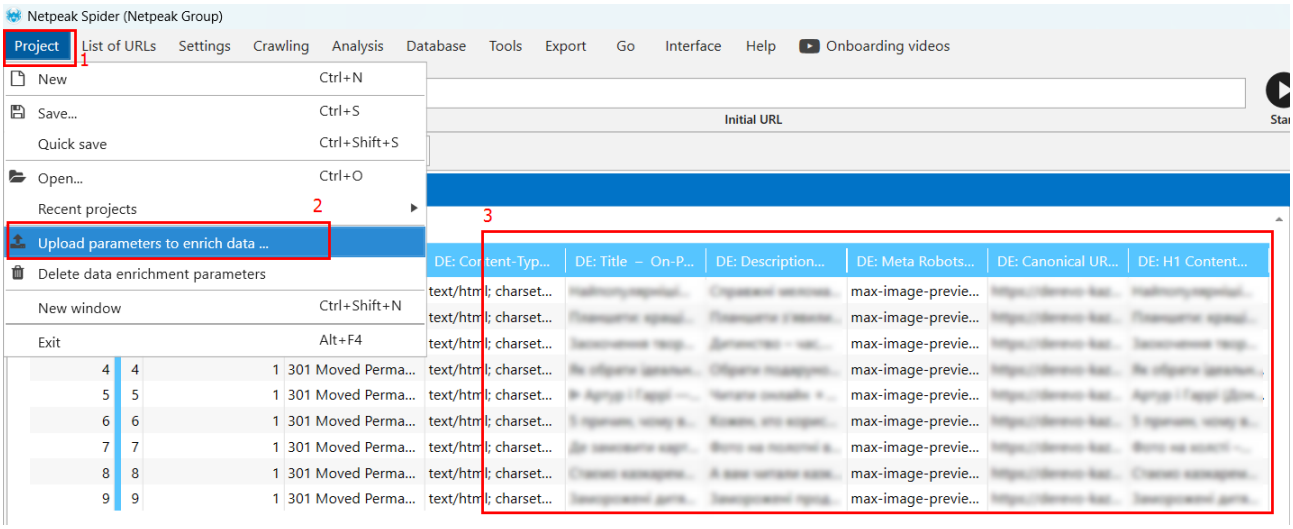

Рисунок 1.15 – Експорт даних з Netpeak Checker

Netpeak Checker - це багатофункціональний інструмент 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 [4].

1.4. Висновки до розділу 1

В першому розділі кваліфікаційної роботи було проаналізовано функціонал програм-парсерів та розширень-парсерів (результати порівнянь можна побачити в таблиці 1.1). Виходячи з розглянутих прикладів було вирішено розробити скрипт для парсингу даних з гугл карт, для кращого та ефективнішого пошуку клієнтів як закордоном, так і всередині країни.

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.

Таблиця 1.1 – Порівняння ресурсів

Характеристики	Netpeak Checker	Web Scraper	Import.io
Опис	Аналіз SEO та аудит веб-сторінок	Автоматизований збір даних з веб-сайтів	Онлайн-платформа для збору даних без програмування
Функції	Аналіз ключових слів, технічний аудит, порівняння з конкурентами	Витягування даних, налаштування параметрів парсингу, збереження даних	Витягування даних, автоматизація, регулярне оновлення
Переваги	Широкий функціонал для SEO-аудиту, зручність порівняння з конкурентами	Гнучкість та налаштування скриптів, автоматизація процесу збору даних	Простий у використанні, можливість створювати роботів без програмування
Недоліки	Обмежена безкоштовна версія, платна підписка для повного доступу	Вимагає розуміння програмування, можливі обмеження з захистом даних	Обмежена можливість налаштування складних сценаріїв, платна підписка
Ціна	Платна	Безкоштовний (з деяким обмеженням) та платні плани	Платний план
Тип інструменту	SEO аналіз та аудит	Веб-скрапер (Web scraping)	Платформа збору даних
Налаштування	Широкий спектр можливостей для SEO аналізу та аудиту, можливість порівняння з конкурентами	Гнучкість та налаштування скриптів для витягування даних, параметри обходу та парсингу	Можливість створення роботів без програмування для витягування даних
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
Підтримка та оновлення	Підтримка та оновлення для платних планів	Обмежена підтримка для безкоштовного плану, підтримка та оновлення для платних планів	Підтримка та оновлення для платного плану

2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ

2.1. Вибір технології розробки

Для реалізації задачі було обрано середовище Atom, яке надає можливість розробляти застосунки, які реалізовані на платформі Node.js.

Atom – це редактор коду, який надає засоби для кросплатформового редагування коду. Він має вбудований пакетний менеджер і інтерфейс навігації файловою системою. Atom дозволяє спільно працювати з кодом, надає інтелектуальну систему автодоповнення вводу і підтримку режимів сумісності з Vim і Emacs. Також він підтримує API для розробки розширень.

У Atom можна відкривати кілька файлів в різних вкладках і показувати їх одночасно з використанням вертикального або горизонтального розбиття панелей. Інтерфейс можна налаштовувати за допомогою тем оформлення, підтримуються вкладки, закладки, контекстний пошук коду, схлопування блоків коду, курсори і області виділення, позначення змін, автодоповнення та перевірка коду для різних мов програмування.

Для редактора Atom доступний широкий набір пакетів-розширень, які можуть додавати нові функціональність. Ці пакети можна встановлювати за допомогою вбудованого пакетного менеджера apm. Розробка розширень для Atom схожа на розробку програм для Node.js і включає в себе використання модулів Node.js і популярних JavaScript-бібліотек.

У Atom також є каталог сторонніх пакетів, в якому можна знайти багато додаткових пакетів і тем оформлення, що дозволяє налаштувати редактор під свої потреби. [5].

Node.js – це платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Вона дозволяє 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 управління модулями в Node.js використовується пакетний менеджер npm (Node Package Manager). Node.js може бути використаний для створення клієнтських і серверних програм. Для обробки багатьох паралельних запитів використовується асинхронна модель, що базується на обробці подій та використанні зворотних викликів (callback).

Node.js подібний до інших фреймворків, таких як Perl AnyEvent, Ruby Event Machine і Python Twisted, але він приховує цикл обробки подій (event loop) від розробника, нагадуючи обробку подій у веб-застосунку, що працює в браузері.[6].

Для запуску скриптів 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. [7].

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 оболонка командного рядка, яка розшифровується як Bourne Again Shell. Вона є покращеною версією оригінальної оболонки Bourne і має додаткові функції та можливості, такі як параметри виклику кількох символів, редагування командного рядка, журнал команд та інші. Bash доступний в Unix-подібних операційних системах, таких як macOS і Linux.

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. [8].

2.2. Розробка скрипта для парсингу

Для кращого розуміння алгоритму формування цільової клієнтської бази для замовника, було розроблено блок-схему, зображену на рисунку 2.1. Процес формування відбувається наступним чином:

a) Аналітик отримує інформацію про заявку замовника на формування клієнтської бази в певній область чи з певним продуктом;

b) Розуміючи галузь, до якої відноситься даний продукт, формуються ключові слова та перевіряються на актуальність в Google Maps;

c) Далі формується орієнтовний список цільових країн (наприклад, країни ЄС);

d) запускається скрипт, на основі визначених ключових слів та цільових країн;

e) Після завершення роботи скрипта формується таблиця на основі отриманих результатів;

f) Якщо даних, які було зібрано, достатньо – сформована база відправляється до замовника;

g) Якщо ж їх занадто мало або вигрузка має малу конверсію – формуються нові ключові слова або визначаються нові цільові країни (інколи обидва варіанти). Після чого знову проганяється скрипт, формується таблиця та перевіряється дані на повноту на задовільну конверсію

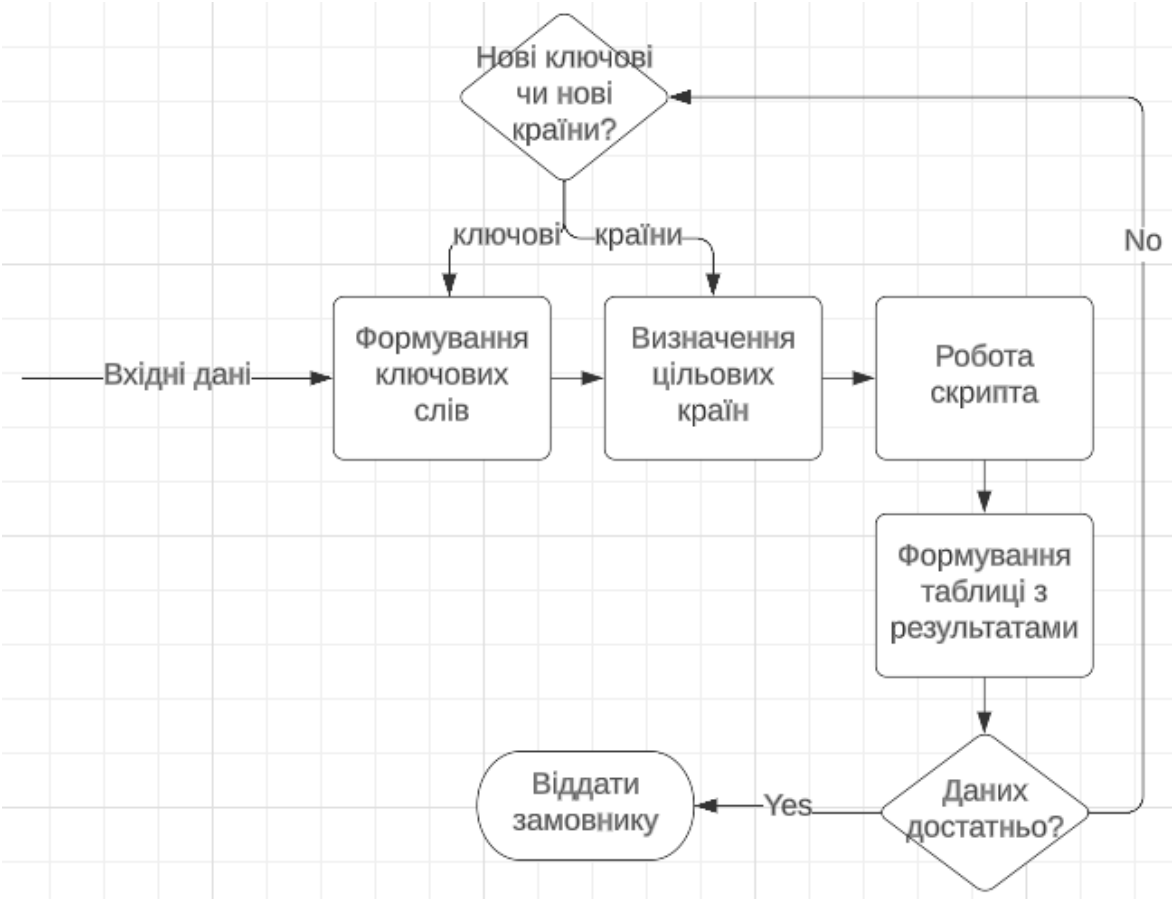

Рисунок 2.1 – Алгоритм формування цільової клієнтської бази для замовника

Щоб зрозуміти алгоритм роботи самого скрипта було розроблено блок-схему, подану на рисунку 2.2.

Основний алгоритм роботи скрипта наступний:

a) Початковим кроком є запуск програми Git Bash;

b) Далі вводиться команда для запуску скрипта, ключові слова в лапках, цільова країна, а також кількість карток, які будуть витягнути за один запит;

c) Скрипт формує масив запитів з міст цільової країни та ключових слів;

d) Запити з масиву проганяються в Google Maps;

e) Перевіряється чи запит містить звичайні картки чи короткі картки;

f) 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;

g) Якщо запит містить короткі картки, їх записується в кінцевий файл.

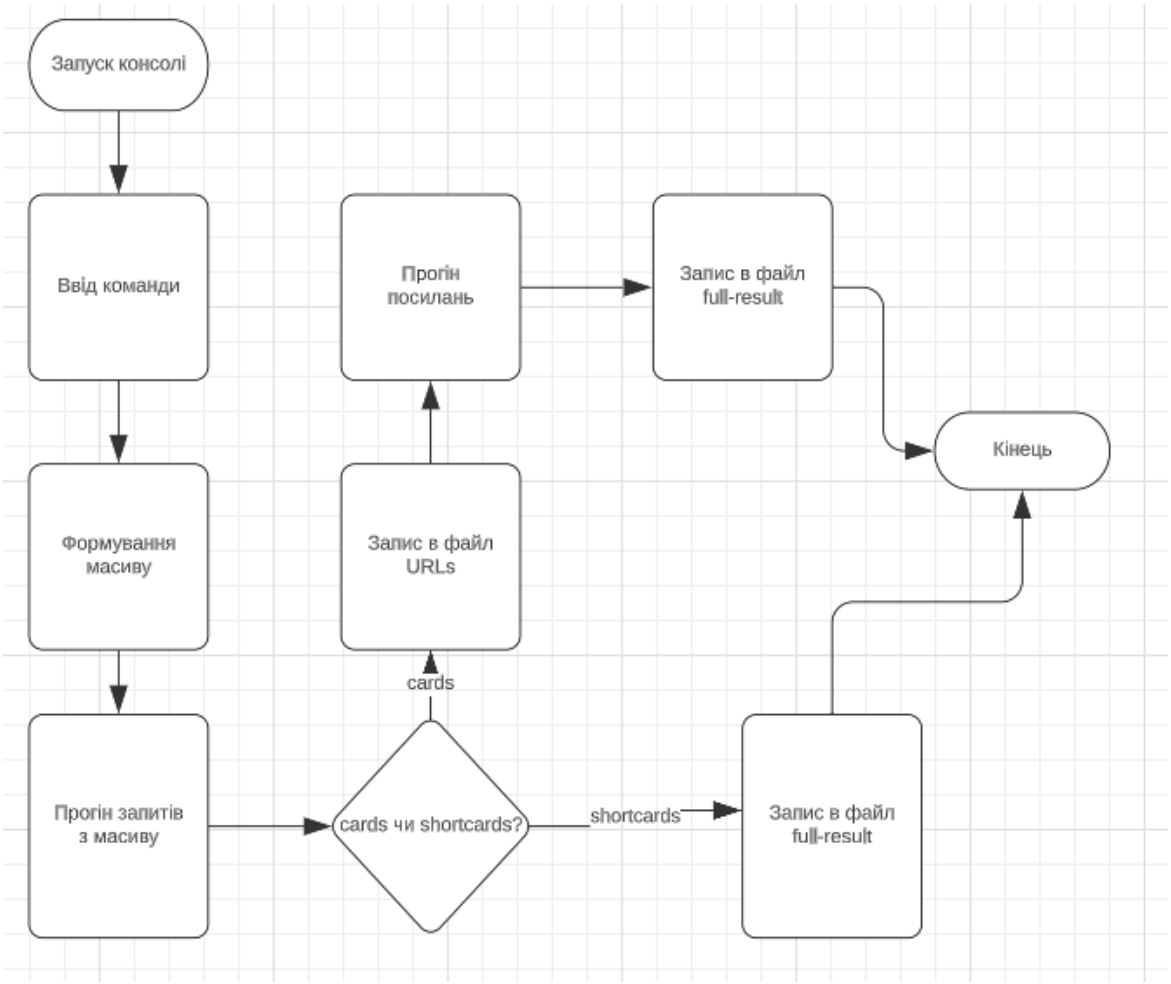

Рисунок 2.2 – Алгоритм роботи скрипта

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. Код створення масиву запитів для перебору подано в лістингу 2.1.

Лістинг 2.1 – Створення масиву запитів для перебору:

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

Реалізуємо перевірку даних на валідність: Основна логіка коду полягає у наступному:

Отримуються аргументи командного рядка за допомогою const arguments = process.argv.slice(2). process.argv, що містить масив аргументів командного рядка, а slice(2) використовується для отримання аргументів починаючи з третього.

Перевіряється, чи аргументи не є порожніми або мають неправильний формат. Умова arguments.length == 0 || arguments.length > 3 перевіряє чи довжина масиву аргументів дорівнює 0 або перевищує 3. Якщо умова виконується, виводиться повідомлення про помилку log.e('arguments empty or has a wrong format') і викликається функція exit(), яка припиняє виконання програми.

Перевіряється, чи властивість arguments[1].toUpperCase() існує у places.countriesList. Це перевіряє чи короткий код країни, переданий як другий аргумент командного рядка, існує у списку країн. Якщо умова не виконується, виводиться повідомлення про помилку log.e('you have entered nonexistent country shortname'), і викликається функція exit().

Перевіряється, чи третій аргумент командного рядка існує і є числом. Використовується !Number.isInteger(+arguments[2]) для перевірки чи можна перетворити третій аргумент у число. Якщо умова не виконується, виводиться повідомлення про помилку log.e('third argument is not a number'), і викликається функція exit().

Коротко кажучи, дана функція перевіряє правильність та формат аргументів командного рядка і забезпечує коректність введених даних перед продовженням виконання програми. Код функції наведено в лістингу 2.2.

Лістинг 2.2 – 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();
}
```

Розглянемо функцію startParse, яка виконує процес початку парсингу. Основна логіка коду полягає у наступному:

Створюється новий екземпляр браузера за допомогою puppeteer.launch({ headless: false }). У цьому випадку браузер запускається з графічним інтерфейсом, оскільки headless встановлено false. Створюється нова сторінка за допомогою browser.newPage(). Встановлюється розмір вікна сторінки за допомогою page.setViewport({ width: 1280, height: 600 }).

Виконується перехід на вказаний URL за допомогою page.goto("https://www.google.com.ua/maps/?hl=en").

Очікується поява селектора 'input#searchboxinput' на сторінці протягом 5 секунд за допомогою page.waitForSelector('input#searchboxinput', {timeout:5000}).

Запускається цикл for...of для кожного currentQuery в queriesArray.

Виконується handleSingleQuery(page, currentQuery, placesArray) для обробки одного запиту. Якщо виникає помилка, виводиться повідомлення про помилку log.error(throuble on ${currentQuery},\n, error).

Виводиться повідомлення log(currentQuery).

Виводиться повідомлення log.debug('done all scrolling').

Браузер закривається за допомогою browser.close().

Викликається parseLinks(placesLinksIterator(parsedPlacesLinks)) для обробки посилань.

Виводиться повідомлення log.start(lines total: ${parsedPlacesLinks.length}\nlines done [%s], 1).

Отже, дана функція виконує кроки підготовки, запуску браузера, переходу на веб-сторінку та обробки запитів для парсингу. Після закінчення парсингу виводяться відповідні повідомлення. Код функції наведено в лістингу 2.3.

Лістинг 2.3 – Запуск роботи скрипта

```
async function startParse(placesArray) {
  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,
placesArray);
    } 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);

}
```

Функція керування введенням з клавіатури handleSingleQuery, яка виконує обробку одного запиту. Основна логіка коду полягає у наступному:

На початку функції сторінка отримує фокус на полі вводу за допомогою await page.focus('input#searchboxinput').

Наступні рядки коду виконують дії, щоб очистити поле вводу: натискаються клавіші Control+A для вибору всього тексту, потім Backspace для видалення тексту.

Потім вставляється новий текст за допомогою await page.type('input#searchboxinput', item) та натискається клавіша Enter для відправки запиту.

Використовується await page.waitForSelector для очікування наявності селектора .m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd на сторінці протягом 60 секунд.

Якщо селектор знайдений, виконується блок коду всередині:

Спочатку викликається функція autoScroll(page), яка прокручує сторінку.

Потім перевіряється наявність елемента з селектором ".lcr4fd.S9kvJb". Якщо елемент знайдений, виконується наступний блок коду.

Використовується page.evaluate для отримання даних з елементів сторінки.

Дані про карти збираються у масив cardsData.

Після цього викликається parseResultWriter.writeRecords(cardsData), яка записує ці дані у файл.

Якщо елемент з селектором ".lcr4fd.S9kvJb" не знайдений, виконується інший блок коду.

Знову за допомогою page.evaluate отримується масив URIs (URL-адреси) елементів сторінки.

URIs додаються до масиву parsedPlacesLinks.

Записуються дані в CSV-файл за допомогою csvWriter.writeRecords(URIs).

Незалежно від результату попередніх кроків, виконується log.info("done single"). Якщо відбувається помилка під час виконання блоку коду всередині await page.waitForSelector, виконується блок коду всередині catch.

Виводиться повідомлення log('FAIL').

Виводиться докладна інформація про помилку log.error(fail on ${item} query,\n, e).

У загальному, цей код виконує пошук на сторінці і обробляє отримані результати, залежно від наявності певних елементів. Код функції керування введенням з клавіатури подано в додатку А.

Автоматичне прокручування сторінки в даному випадку було реалізовано наступним чином:

Функція спочатку перевіряє наявність селектора .PbZDve за допомогою await page.$('.PbZDve'). Якщо селектор знайдений, виводиться повідомлення log('end selector'), і функція повертає значення true, що означає, що прокрутка закінчилась. Потім функція отримує список елементів з селектором .hfpxzc за допомогою await page.$$('.hfpxzc'). Якщо кількість елементів перевищує значення змінної scrollCardsAmount, виводиться повідомлення log('too much places cards'), і функція повертає true.

Функція чекає протягом 500 мілісекунд за допомогою new Promise(r => setTimeout(r, 500)). Цей крок виконує затримку перед наступними діями. Далі виконується page.evaluate для виконання JavaScript-коду у контексті сторінки.

Спочатку прокручується сторінка вгору на 300 пікселів за допомогою scrollElem.scrollTop = scrollElem.scrollTop - 300. Після прокрутки сторінки функція чекає, поки мережева активність не стане неактивною за допомогою page.waitForNetworkIdle(). Це допомагає забезпечити завершення завантаження додаткових даних після прокрутки.

Потім знову виконується page.evaluate для прокрутки сторінки вниз на значення 100000 пікселів. Після цього знову чекається, поки мережева активність не стане неактивною.

Нарешті, викликається рекурсивно autoScroll(page), що запускає наступний цикл прокрутки сторінки.

У загальному, ця функція використовується для автоматичної прокрутки сторінки, допоки не будуть виконані певні умови (досягнуто кінця списку або перевищено кількість елементів). Код даної функції наведено в лістингу 2.4.

Лістинг 2.4 – Автопрокручування сторінки

```
async function autoScroll(page) {
  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;
  }

  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)
      }
```

Для обробки зібраних посилань було розроблену функцію parseLinks(), яка відповідає за обробку посилань. Основна логіка коду полягає у наступному:

Перевіряється, чи url дорівнює false. Якщо це так, виводиться повідомлення log.info('exit'), і функція exit() викликається. Це служить для припинення виконання програми.

Виконується await axios(url), що виконує запит за допомогою бібліотеки Axios на вказаний URL. Якщо запит успішний, виконується блок коду всередині .then. Виводиться статус відповіді log(response.status).

Викликається handleResponseData(response.data), що, очевидно, обробляє дані відповіді.

Отримані результати записуються за допомогою parseResultWriter.writeRecords(currentResult).

Виводиться повідомлення log.debug('current line written'). Якщо запит не вдалий, виконується блок коду всередині .catch.

Виводиться повідомлення про помилку log.error(error).

Після обробки посилання викликається parseLinks(placesLinksIterator(parsedPlacesLinks)), що рекурсивно викликає parseLinks з новим посиланням для обробки.

У загальному, цей код виконує запит до вказаного URL, обробляє отримані дані і повторює процес для наступного посилання. Код функції обробки посилань наведено в лістингу 2.5.

Лістинг 2.5 – Обробка посилань

```
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))
}
```

Розглянемо обробку даних відповіді, отриманих під час парсингу. Основна логіка коду полягає у наступному:

Створюється порожній об'єкт currentLine, який буде містити результат обробки даних.Знаходиться індекс кінця рядка інформації (infoStrEnd) та початку рядка інформації (infoStrStart). За допомогою data.slice отримується рядок інформації infoStr з даних.

Перевіряється, чи містить рядок infoStr роздільник "·". Якщо так, то розділяється рядок на дві частини (назва та адреса) і присвоюється відповідним властивостям currentLine. Якщо роздільник відсутній, назва присвоюється властивості name, а адреса отримує значення "null".

Знаходяться індекс початку рядка з активністю (activityStrStart) та кінця рядка з активністю (activityStrEnd).

Перевіряється, чи існує рядок з активністю. Якщо так, то за допомогою data.slice отримується активність та присвоюється властивості activity currentLine. Якщо рядок відсутній, активність отримує значення "null".

Знаходяться індекс початку рядка з номером телефону (phoneStrStart) та кінця рядка з номером телефону (phoneStrEnd).

Перевіряється, чи існує рядок з номером телефону. Якщо так, то за допомогою data.slice отримується номер телефону та присвоюється властивості phone currentLine. Якщо рядок відсутній, номер телефону отримує значення "null".

Використовується регулярний вираз httpRegexG для виявлення всіх посилань у тексті.

Фільтруються посилання, щоб виключити ті, які містять певні ключові слова, такі як "google", "gstatic", "ggpht", "schema" і "broofa".

Отримані посилання зберігаються у вигляді масиву result та перетворюються у множину (Set) для видалення дублікатів.

Перевіряється розмір множини result. Якщо вона рівна 0, значенню властивості site currentLine присвоюється "null". Якщо розмір не дорівнює 0, то властивості site currentLine присвоюється перше посилання з множини.

Повертається об'єкт currentLine з обробленими даними.

Ця функція призначена для обробки рядка HTML-даних і створення об'єкта з витягнутою з нього інформацією про назву, адресу, активність, номер телефону та посилання. Код функції наведено в додатку Б.

2.3. Тестування продукту

Для запуску скрипта відкриваємо Git Bash (вікно програми зображено на рисунку 2.3) та вводимо команду запуску скрипта. В консолі вводимо команду node, вказуємо назву файлу (в нашому випадку це parser.js), в лапках вказуємо ключове слово, потім вказуємо короткий код країни та кількість карток на один запит.

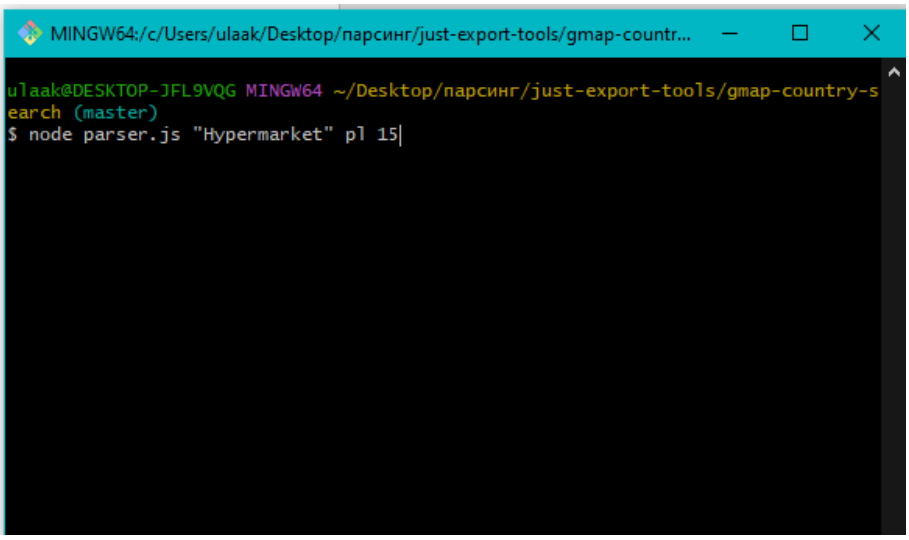

Рисунок 2.3 – Вікно програми Git Bash

Після натиснення клавіші Enter перед користувачем з’явиться сформований масив даних, які будуть вводитися в рядок пошуку при виконанні скрипта. В даному випадку масив даних наведений на рисунку 2.4.

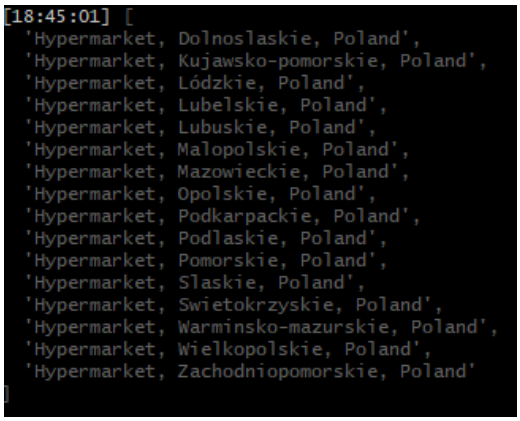

Рисунок 2.4 – Сформований масив даних

Далі перед користувачем відкриється веб-переглядач, в якому користувач може слідкувати за тим, що в даний момент часу робить скрипт. Вікно веб-переглядача під час роботи зображено на рисунку 2.5.

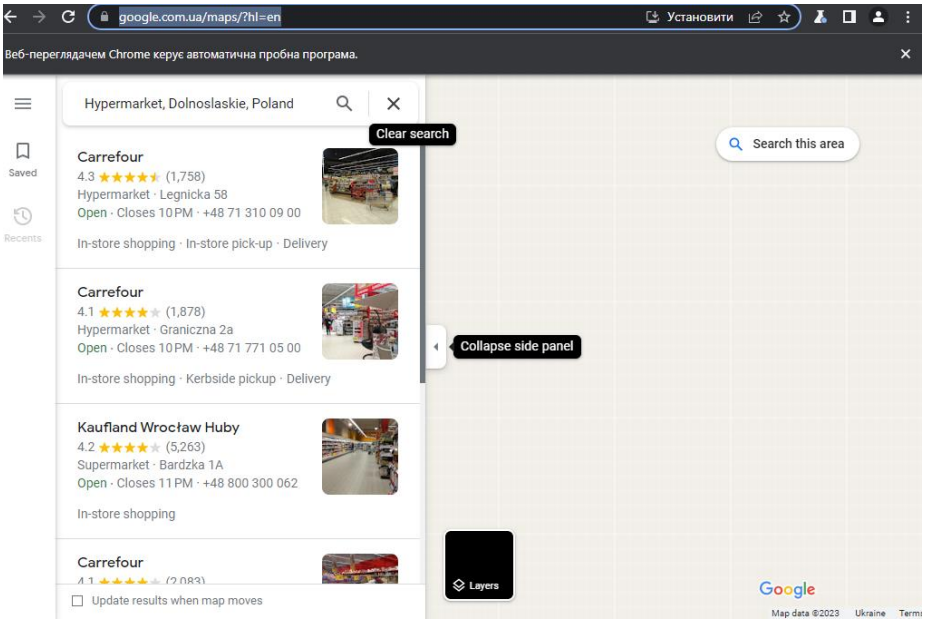

Рисунок 2.5 – Вікно веб-переглядача під час роботи

Після обробки одного запиту в консолі з’явиться короткий звіт щодо виконаних дій над запитом, як на рисунку 2.6.

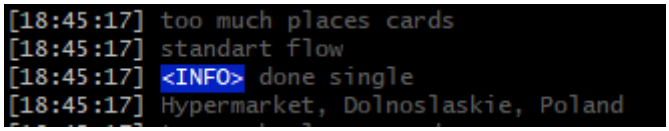

Рисунок 2.6 – Повідомлення в консолі

Як тільки скрипт пройде всі запити і збере всі посилання, почнеться етап, на якому буде вигружатись інформація, яка вказана в картці компанії, тобто назва компанії, активність, адреса, телефон та сайт. Кількість даних та поточний стан роботи також відображаються в консолі, приклад наведено на рисунку 2.7.

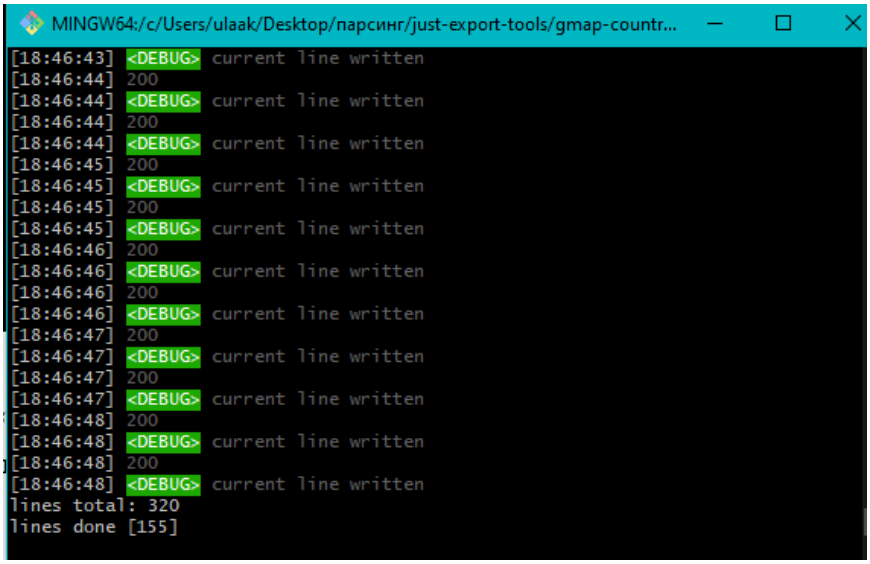

Рисунок 2.7 – Процес збору даних

В кінці буде виведено повідомлення про успішне завершення роботи скрипта.

Зібрані дані зберігаються в CSV-файлі. Результати парсингу можна побачити на рисунку 2.8.

	name	activity	address	phone	site
2	Auchan	Hypermarket	Francuska 6 55-040 Bielany WrocE,awskie, Poland	48224440222	null
3	Carrefour	Hypermarket	Czekoladowa 7/9 55-040 Bielany WrocE,awskie, Poland	48713271650	https://www.carrefour.pl/
4	Carrefour	Hypermarket	Noworudzka 2 57-300 KE,odzko, Poland	48748627700	https://www.carrefour.pl/
5	Kaufland RacibГirz	Supermarket	Podwale 22 47-400 RacibГirz, Poland	48800300062	http://www.kaufland.pl/%3Fcid%3DF19054B02C1200K01000W01030000
6	Carrefour	Hypermarket	al. Gen. JГizefa Hallera 52 53-324 WrocE,aw, Poland	48713690100	https://www.carrefour.pl/
7	Carrefour	Hypermarket	aleja PiE,sudskiego 84 59-220 Legnica, Poland	48767239200	https://www.carrefour.pl/
8	E.Leclerc	Hypermarket	ZakE,adowa 2-4 50-231 WrocE,aw, Poland	48717971700	https://wroclaw.leclerc.pl/
9	EUROSPAR	Supermarket	Arkady PowstaE„cГiw EљД…skich 2-4, 53-333 WrocE,aw, P	48717242160	http://www.spar.pl/
10	Jelenia Gora Auchan	Hypermarket	Aleja Jana PawE,a II 52 58-506 Jelenia GГira, Poland	48224440222	null
11	Auchan	Hypermarket	Szafirowa 55 44-100 Gliwice, Poland	48224440222	null
12	Lidl	Discount supermarket	MarszaE,ka JГizefa PiE,sudskiego 7 48-303 Nysa, Poland	48618803000	https://www.lidl.pl/
13	Carrefour	Hypermarket	Lipowa 1 44-100 Gliwice, Poland	48323027860	https://www.carrefour.pl/
14	WrocE,aw Market Hall	Produce market	Piaskowa 17 50-359 WrocE,aw, Poland	48713438457	http://hala-targowa.pl/
15	E. Leclerc Eљwidnica	Hypermarket	Strzegomska 2 58-100 Eљwidnica, Poland	48748577410	https://swidnica.leclerc.pl/gazetki/
16	Carrefour	Hypermarket	PodhalaE„ska 26 44-335 JastrzД™bie-ZdrГij, Poland	48324781600	https://www.carrefour.pl/
17	Carrefour	Hypermarket	Legnicka 58 54-204 WrocE,aw, Poland	48713100900	https://www.carrefour.pl/
18	Carrefour	Hypermarket	Graniczna 2a 54-610 WrocE,aw, Poland	48717710500	https://www.carrefour.pl/
19	Kaufland WrocE,aw Huby	Supermarket	Bardzka 1A 50-516 WrocE,aw, Poland	48800300062	http://www.kaufland.pl/%3Fcid%3DF19054B02C1200K01000W01030000
20	Carrefour	Hypermarket	Kozielska 20 47-224 KД™dzierzyn-KoEele, Poland	48774050200	https://www.carrefour.pl/
21	Auchan	Hypermarket	CH Korona BolesE,awa Krzywoustego 126, 51-421 WrocE,a	48224440222	null
22	Auchan	Hypermarket	Francuska 6 55-040 Bielany WrocE,awskie, Poland	48224440222	null
23	Carrefour	Hypermarket	Czekoladowa 7/9 55-040 Bielany WrocE,awskie, Poland	48713271650	https://www.carrefour.pl/
24	Carrefour	Hypermarket	Noworudzka 2 57-300 KE,odzko, Poland	48748627700	https://www.carrefour.pl/
25	Kaufland RacibГirz	Supermarket	Podwale 22 47-400 RacibГirz, Poland	48800300062	http://www.kaufland.pl/%3Fcid%3DF19054B02C1200K01000W01030000
26	Carrefour	Hypermarket	al. Gen. JГizefa Hallera 52 53-324 WrocE,aw, Poland	48713690100	https://www.carrefour.pl/
27	Carrefour	Hypermarket	aleja PiE,sudskiego 84 59-220 Legnica, Poland	48767239200	https://www.carrefour.pl/
28	E.Leclerc	Hypermarket	ZakE,adowa 2-4 50-231 WrocE,aw, Poland	48717971700	https://wroclaw.leclerc.pl/
29	EUROSPAR	Supermarket	Arkady PowstaE„cГiw EљД…skich 2-4, 53-333 WrocE,aw, P	48717242160	http://www.spar.pl/

Рисунок 2.8 – Дані, зібрані під час парсингу

2.4. Висновки до розділу 2

В другому розділі кваліфікаційної роботи було розглянуто технології, які будуть використовуватись при розробці скрипта. Розглянуто середовище розробки Atom, а також програму Git Bash.

На основі аналізу існуючих парсингових програму, проведеному в розділі 1, було розроблено основний алгоритм формування цільової клієнтської бази для замовника, а також, алгоритм роботи безпосередньо самого скрипта.

За допомогою Node.js та безпосередньо JavaScript, було реалізовано основний функціонал скрипта, що розробляється, а також набуто детальніше розуміння поставлених перед скриптом задач, які мають вирішуватись.

Проведено тестування скрипта на коректність роботи, а саме: формування масиву запитів, введення цих запитів в рядок пошуку, автоматичне прокручування сторінки, збір даних та запис цих даних в вихідний файл.

3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1. Дія електричного струму на організм людини, види електротравм.

Дія електричного струму на організм людини має багатогранний характер. Проходячи через тіло людини електричний струм здійснює термічний, електролітичний та біологічний вплив на різні системи організму. При цьому виникають порушення діяльності життєвоважливих органів людини: мозку, серця, легенів[16].

Термічна дія струму проявляється в опіках окремих частин тіла.

Електролітична дія струму полягає в розкладі крові та інших органічних рідин організму і викликає значні спотворення їх фізико-хімічного складу.

Біологічна дія струму проявляється як подразнення та збудження живих тканин організму, що супроводжуються неконтрольованими (судомними) скороченнями м'язів, в тому числі легенів та серця, і може привести до порушення біологічних процесів[17].

Всі види дії електричного струму на організм людини умовно поділяють на дві основні групи уражень: електричні травми та електричні удари.

Електричні травми - це місцеві пошкодження тканин і органів, причиною яких є дія електричного струму або електричної дуги. Розрізняють наступні електричні травми: a) електричний опік; b) електричні знаки; c) металізація шкіри; d) електроофтальмія; e) механічні ушкодження.

1. Електричний опік - найбільш розповсюджена електротравма. Опіки бувають двох видів: струмовий (або контактний) і дуговий. Струмовий опік зумовлений проходженням струму величиною більше 1А через тіло людини в результаті контакту із струмоведучою частиною і є наслідком перетворення електричної енергії у теплову.

Розрізняють 4 ступені опіку: 1 - почервоніння шкіри; 2 - утворення пузирів; 3 - омертвіння усієї товщини шкіри; 4 - обвуглення тканин. Ступінь ураження організму визначається не ступенем опіку, а площиною ураженої поверхні тіла.

Дуговий опік. При більш високих напругах між струмоведучою частиною і тілом людини виникає електрична дуга (температура дуги > 3500°С), яка і спричиняє дуговий опік. Вони, як правило, дуже важкі: 3 або 4 ступеня.

2. Електричні знаки - чітко окреслені пятна сірого або жовтого кольору на поверхні шкіри людини в місці контакту з електродами. Знаки бувають також у вигляді подряпин, ран, порізів або ушибів, бородавок та мозолів. Електричні знаки - безболісні, розміри п'ятен до 10 мм

3. Металізація шкіри - це проникнення у верхні шари шкіри дрібних частинок металу, який розплавився під дією електричної дуги. Це може відбутися при коротких замиканнях, відключенні рубильника під навантаженням, тощо. Металізація супроводжується опіком шкіри.

4. Електроофтальмія - ураження очей, яке викликається інтенсивним випромінюванням електричної дуги, спектр якої містить шкідливі для очей ультрафіолетові та інфрачервоні промені. Окрім того, можливе попадання в очі частинок розплавленого металу. Захист від електроофтальмії досягається за допомогою захисних окулярів.

5. Механічні ушкодження виникають в результаті різних неконтрольованих судомних скорочень м'язів під дією електричного струму. В результаті можуть виникати розриви шкіри, кровеносних судин і нервової тканини, а також вивихи суглобів і переломи кісток.

Електричний удар - це збудження живих тканин організму електричним струмом, яке супроводжується неконтрольованими судомними скороченнями м'язів. В залежності від наслідків дії струму на організм електричні удари умовно поділяються на 4 ступені: 1 - судомне скорочення м'язів без втрати свідомості; 2 - втрата свідомості, але збереження дихання і роботи серця; 3 - втрата свідомості і порушення діяльності серця або дихання (або того і другого); 4 - клінічна смерть, тобто відсутність дихання і кровообігу. Клінічна смерть -це перехідний період від життя до смерті. Ознаки клінічної смерті - зупинка серця, відсутність пульсу, дихання , зрачки очей розширені і не реагують на світло, відсутні больові відчуття. Тривалість клінічної смерті від 5 до 10 хвилин, в залежності від ступеня ураження та індивідуальних здатностей організму[18].

3.2. Вимоги безпеки до ПК та устаткування.

Правила охорони праці під час експлуатації електронно-обчислювальних машин (далі – Правила) поширюються на всіх суб'єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов'язану з електронно-обчислювальними машинами (далі – ПК) з відеодисплейними терміналами (далі – ВДТ), у тому числі на тих, які мають робочі місця, обладнані ПК з ВДТ і периферійними пристроями (далі – ПП).

Правила встановлюють вимоги безпеки до обладнання робочих місць операторів ПК з ВДТ та ПП (далі - оператори) та до роботи із застосуванням ПК з ВДТ і ПП.

Вимоги Правил є обов'язковими для роботодавців, операторів електронно-обчислювальних машин, операторів комп'ютерного набору, операторів комп'ютерної верстки та працівників інших професій, які у своїй роботі застосовують ПК з ВДТ і ПП.

Вимоги Правил не поширюються на:

– комп'ютерні класи вищих і середніх навчальних закладів, майстерні професійно-технічних навчальних закладів;

– робочі місця операторів ПК, які використовуються у сфері керування й експлуатації атомних електростанцій;

– робочі місця пілотів, чи водіїв-операторів транспортних засобів, обладнаних ПК, ПК у системах обробки даних на бортах засобів повідомлення й ПК у складі машин і устаткування, що переміщується у процесі роботи;

– так називані портативні системи обробки даних, якщо вони використовуються на робочому місці мінливо;

– обчислювальні машинки (калькулятори), каси, які регіструють, і прилади з невеликими пристроями індикації чи даних результатів вимірів;

– друкарські машинки класичної конструкції, обладнані відеотерміналом (так названі дисплейні друкарські машинки);

– комп'ютерні ігрові автомати і системи обробки даних, призначені для суспільного користування.

Вимоги Правил є обов'язковими для всіх працівників при організації і виконанні робіт, пов'язаних з експлуатацією, обслуговуванням, налагодженням і ремонтом ПК, а також при проектуванні і реконструкції підприємств, їхніх виробничих об'єктів, споруджень і робочих місць, обладнаних ПК.

Вимоги електробезпеки під час експлуатації ПК з ВДТ і ПП

ПК з ВДТ і ПП, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники), електропроводи та кабелі за виконанням і ступенем захисту мають відповідати класу зони за НПАОП 40.1-1.01-97 (z0011-98), мати апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію.

Під час ремонту ліній електромережі шляхом зварювання, паяння та з використанням відкритого вогню необхідно дотримуватися НАПБ А.01.001 2004 (z1410-04).

Лінія електромережі для живлення ПК з ВДТ і ПП виконується як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник.

Нульовий захисний провідник прокладається від стійки групового розподільного щита, розподільного пункту до розеток електроживлення.

Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі

та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97 (z0011-98).

У приміщенні, де одночасно експлуатуються понад п'ять ПК з ВДТ і ПП, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ПК з ВДТ і ПП повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Їхня конструкція має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше, ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним.

Не допускається підключати ПК з ВДТ і ПП до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення ПК з ВДТ і ПП потрібно виконувати за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі.

Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В.

Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог НПАОП 40.1-1.01-97 (z0011-98) та НАПБ А.01.001-2004 (z1410-04).

Електромережу штепсельних розеток для живлення ПК з ВДТ і ПП при розташуванні їх уздовж стін приміщення прокладають по підлозі поруч зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах, а також у пластикових коробах пластмасових рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розміщенні в приміщенні до п'яти ПК з ВДТ і ПП допускається прокладання трипровідникового захищеного проводу або кабелю в оболонці з негорючого чи важкогорючого матеріалу по периметру приміщення без металевих труб та гнучких металевих рукавів.

Не допускається в одній трубі прокладати ланцюги до 42 В та вище 42 В.

При організації робочих місць операторів електромережу штепсельних розеток для живлення ПК з ВДТ і ПП у центрі приміщення прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не допускається застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, які містять сірку[19].

3.3. Висновок до третього розділу

В даному розділі здійснено аналіз наступних питань: дія електричного струму на організм людини, види електротравм, вимоги безпеки до ПК та устаткування. Розглянуто чималу кількість нормативних документів та визначено основні види дії електричного струму на людський організм, а також їх наслідки. Розглянуто ступені електричного опіку та наслідки електричного удару.

Також було ознайомлено з вимогами до безпеки ПК та устаткування та передбачено необхідність дуже відповідального підходу до облаштування робочого місця на підприємстві. Визначено необхідність встановлення аварійних вимикачів електричного живлення в приміщеннях, в яких експлуатується понад 5 ПК. Продемонстровано вимоги безпеки до ПК та облаштування робочого місця для осіб, які будуть працювати з розробленим скриптом.

ВИСНОВКИ

В процесі дослідження програмно-алгоритмічних засобів в логістиці неможливо не звернути увагу на роль парсерів в даній сфері, адже саме завдяки парсерам можна формувати нові клієнтські бази та виходити на ширший ринок не лише на території Україні, але й за її межами. Проаналізувавши принципи роботи існуючих парсингових систем, було визначено основні поняття, структуру та компоненти, їх специфіку та особливості.

На основі отриманих даних було побудовано формування цільової клієнтської бази для замовника та, безпосередньо, алгоритм роботи майбутнього скрипта, а також, визначено, який інструментарій та які технології використовуватимуться. Завдяки використанню платформи Node.js, виконання скрипта швидке та надається можливість паралельно запускати декілька процесів, що значно скорочує час роботи над формуванням клієнтської бази для замовника.

Розроблений скрипт надає можливість проводити пошук та формування цільової клієнтської бази, де більшу частину роботи (практично всю, окрім зведення даних в спільну таблицю) виконує безпосередньо сам скрипт, що дає можливість паралельно виконувати інші завдання. Завдяки швидкому виконанню робота з даним скриптом не потребує великих часових затрат.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

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 12.06.2023].

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 12.06.2023].

3. Home | Import.io. Онлайн. Home | Import.io. [б. д.]. Режим доступу: https://docs.import.io/. [дата звернення 12.06.2023].

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 12.06.2023].

5. УЧАСНИКИ ПРОЕКТІВ ВІКІМЕДІА. Atom (текстовий редактор) — Вікіпедія. Онлайн. Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Atom_(текстовий_редактор). [дата звернення 12.06.2023].

6. Node.js — Вікіпедія. Онлайн. Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Node.js. [дата звернення 12.06.2023].

7. Git bash: Definition, commands, & getting started | Atlassian. Онлайн. Atlassian. Режим доступу: https://www.atlassian.com/git/tutorials/git-bash. [дата звернення 12.06.2023].

8. What Is Git Bash and How Do You Use It? MUO. Онлайн. Режим доступу: https://www.makeuseof.com/git-bash-what-how-use/. [дата звернення 12.06.2023].

9. Import.io. Онлайн. Import.io. Режим доступу: https://www.import.io/. [дата звернення 12.06.2023].

10. Next Generation of SEO Tools – Netpeak Software. Режим доступу: https://netpeaksoftware.com. [дата звернення 12.06.2023].

11. Парсинг даних з сайтів: що це та на чиєму боці закон. Режим доступу: https://highload.today/uk/parsing/. [дата звернення 12.06.2023].

12. 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 12.06.2023].

13. Легальні інструменти для збору даних в інтернеті: 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 12.06.2023].

14. Інформаційні системи в логістиці : Державний університет телекомунікацій. Онлайн. Головна. Режим доступу: https://dut.edu.ua/ua/lib/1/category/1137/view/1483. [дата звернення 12.06.2023].

15. Outdated browser | Lucid. Онлайн. Outdated browser | Lucid. Режим доступу: https://lucid.app/documents#/documents?folder_id=home. [дата звернення 12.06.2023].

16. Гогіташвілі, Г. Г. Основи охорони праці [Текст] : навчальний посібник / Г. Г. Гогіташвілі, В. М. Лапін ; 4-те вид. випр. і доп. – К. : Знання, 2008. – 302 с. – ISBN 978-966-346-400-8.

17. Атаманчук, П. С., В. В. Мендерецький, О. П. Панчук та О. Г. Чорна. Безпека життєдіяльності [Текст] : навчальний посібник. – К. : Центр учбової літератури, 2011. – 276с. – ISBN 9786110102452.

18. Гандзюк, М. П. Основи охорони праці [Текст] : підручник / М. П. Гандзюк, Є. П. Желібо, М. О. Халімовський ; за ред. М. П. Гандзюка ; МОН України. – 4-е видання. – К. : Каравела, 2008. – 384 с. – ISBN 966-8019-01-6.

19. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин. Онлайн. Офіційний веб-портал парламенту України. [б. д.]. Режим доступу: https://zakon.rada.gov.ua/laws/show/z0293-10#Text. [дата звернення 13.06.2023].

ДОДАТКИ

```
    async function handleSingleQuery(page, item) {
      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.waitForSelector('.m6QErb.DxyBCb.kA9KIf.dS8AEf.ecceSd',       {
timeout: 60000 }).then(async () => {
        try {
          await autoScroll(page);
        } 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.ariaLabel
              let activity = elem.querySelector(".Z8fK3b .W4Efsd
.W4Efsd:first-child span:first-child")
              if (activity !== null) {
                output.activity                                    =
activity.textContent.replace(/,/g, ".")
              } else {
                output.activity = "null"
              }
              let company_address = elem.querySelector(".Z8fK3b
.W4Efsd .W4Efsd:first-child span:nth-child(2) span:nth-child(2)")
              if (company_address !== null) {
                output.address                                     =
company_address.textContent.replace(/,/g, ".")
              } else {
                output.address = "null"
              }
              let phone = elem.querySelector(".Z8fK3b .W4Efsd
.W4Efsd:nth-child(2) span:nth-child(2) span:nth-child(2)")
              if (phone !== null) {
                output.phone   =   phone.textContent.replace(/,/g,
".")
              } 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)
    }
    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)
  });
}
```

```
    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
    }
```