**ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ**

# Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти бакалавр

на тему: «Розроблення програмних комплексів для парсингу інформації з вебсайтів»

Виконав: здобувач вищої освіти
за освітньо-професійною
програмою Інформаційні
управляючі системи спеціальності
126 Інформаційні системи та
технології
освітнього ступеня бакалавр
групи 126ІСТ_бд_2022[1](стн)
Демченко О.Ю.
Керівник: Копішинська О.П.
Рецензент: Муравльов В.В.

**Полтава – 2024 року**

# ВСТУП

Відтоді, як з'явилися перші вебсайти, й обсяги інформації в інтернеті почали лавиноподібно зростати, так само, як і кількість самих вебсайтів, вони стали предметом постійного моніторингу й статистичного, технічного, програмного аналізу. За даними відомих моніторингових компаній кількість вебсайтів у всесвітній мережі інтернет перевищила 2 мільярди на початок 2023 р., в той час, як 10 років тому ця кількість складала лише 6,5 млн [1]. Разом із тим, значна частина вебресурсів є недоступною з різних причин. Найчастіше це пов'язано із закриттям вебсайтів, які не обслуговуються. Але досить поширеним є явища недоступності, низької відвідуваності, видимості сайтів у мережі через технічні причини. З огляду на статистику зростання кількості вебсайтів, стає зрозуміло, що людського життя не вистачить для перегляду та аналізу всіх даних на просторах інтернету, котрі є доступними для користувача. Коли користувачеві необхідно знайти великий обсяг інформації методом ручного збору даних з вебсайтів, на пошук, аналіз та обробку даних витрачається багато часу. В такому випадку перевага для збору, обробці та аналізу даних надається засобам технічного парсингу.

Парсинг – це процес автоматичного збору даних та їх структурування [2]. Спеціальні програми або сервіси-парсери «обходять» сайт та збирають дані, які відповідають заданій умові. Головними перевагами парсингу є висока швидкість пошуку, збору та структурування великого масиву даних.

*Актуальність* теми кваліфікаційної роботи ґрунтується на важливості розробки та застосування для практичних потреб оригінальних додатків для парсингу, створених за допомогою написання коду, використання спеціалізованих профільних сервісів чи застосунків.

*Метою* кваліфікаційної роботи є розроблення системи ефективного пошуку та розповсюдження інформації шляхом використання парсерів.

Для досягнення зазначеної мети необхідно вирішити наступні завдання:

– Здійснити аналіз концепції моніторингу й парсингу вебсайтів;

– провести дослідження технологій розробки додатків для парсингу обома методами;

– розробити рекомендації щодо практичної реалізації парсерів;

– здійснити техніко-економічне оцінювання прийнятих рішень.

*Об'єктом дослідження* кваліфікаційної роботи є процеси парсингу інформації на вебсайтах.

*Предметом дослідження* є технології розробки парсерів із використанням спеціалізованих профільних сервісів чи застосунків.

*Методи досліджень*: аналітико-синтетичний, інформаційно-пошуковий, дедуктивний, порівняльний, абстрактний, реалізації парсингу за допомогою ботів із застосуванням ZennoPoster-моделювання.

Інформаційну базу кваліфікаційної роботи складають: наукові статті, тематичні інтернет-публікації, інструкції розробників програмного забезпечення, статистичні дані вітчизняних та міжнародних аналітичних компаній, які є у вільному доступі, нормативні документи в галузі ІТ, програмне середовище ZennoPoster.

Практична значущість роботи: здійснений структурований аналіз основних методів та напрямів використання технологій парсингу вебсайтів, зокрема й за допомогою ботів, є цікавим для подальшого практичного втілення; рекомендації щодо розроблення додатків для парсингу можуть бути використані для подальших дослідженнь за даною тематикою.

Структура кваліфікаційної роботи логічно пов'язана з задачами досліджень, пояснювальна записка містить перелік умовних познак, вступ, три розділи основної частини, висновки, список використаних джерел. Загальний обсяг текстової частини кваліфікаційної роботи складає 54 сторінки формату А4, містить 28 рисунків і 2 таблиці. В роботі використано посилання на 31 науково–технічне інформаційне джерело.

# РОЗДІЛ 1
# АНАЛІЗ КОНЦЕПЦІЇ ТА ТЕХНОЛОГІЙ ПАРСИНГУ ВЕБСАЙТІВ

## 1.1 Історія походження та сутність концепції вебпарсингу

Парсингом називають автоматизований збір та систематизацію інформації за допомогою скриптів. Інша назва цього процесу це вебскрейпінг. Парсерами є самі скрипти, які збирають та обробляють інформацію. Методи парсингу з вебсайтів можна розділити на кілька груп: написання коду вручну, використання спеціалізованих профільних сервісів чи застосунків, таких як ZennoPoster, хмарні вебсервіси. Загалом, до переваг перших двох методів належать: швидка робота, масштабованість, легка підтримка, значні можливості на обробку та збереження інформації, швидка та надійна інтеграція зі сторонніми сервісами такими як Telegram API (Application Programming Interface), MS Excel, Google Docs тощо.

Концепція вебпарсингу була започаткована ще у 1993 р. до появи першої пошукової системи World Wide Web. У червні 1993 р. був створений перший вебробот, котрий називався World Wide Web Wanderer [3]. Тоді основною задачею парсеру було вимірювання обсягів всесвітньої павутини.

Уже в грудні 1993 р. з'явилася перша пошукова система WWW, яка працювала завдяки старанням іншого вебробота під назвою JumpStation [4]. JumpStation переглядав заголовки документів для індексації вебсторінок, знайдених за допомогою нескладного лінійного пошуку.

Найперші Web API та API Crawler розробили в 2000 р. [5]. Web API – це інтерфейс прикладного програмування для вебсервера або веббраузера. Тоді ж розробники відомих комерційних платформ Salesforce та eBay запустили API, за допомогою якого програмісти отримали змогу напряму отримувати доступ до відкритих даних мережі інтернет. Після цього багато вебсайтів розробляють та дають доступ до Web API, що значно прискорює та облегшує парсинг.

Можливості застосування сучасного вебпарсингу невичерпні, завдяки тому, що в мережі майже не існує сто відсоткових способів захисту даних на

вебсторінці. Все це обумовлюється тим, що під час завантаження вебсайту, браузер користувача отримує повний контент вебсторінки та сітку розмітки контенту, яку вдало знаходять та обробляють сучасні парсери. Фактично парсери можуть не просто знаходити інформацію на вже завантажених сторінках, а й перехоплювати дані на шляху отримування її від вебсервера до браузера користувача.

Вигода у використанні парсерів обумовлюється тим що, парсером легко отримувати, обробляти великий обсяг даних з різних джерел в найкоротший термін. Наприклад, вручну інформацію з 50 сторінок можна зібрати за кілька годин. Парсер, у порівнянні з людиною, може виконувати значно більше дій за короткий період. Більше того, завдяки гнучкості сучасних парсерів, отриману інформацію можна зберігати у найбільш поширених форматах, таких як: .txt, .xlsx, .xls, .csv, .db, .doc, .docx, .png, jpg та інших, в залежності від контенту, що необхідний користувачу або системі (рис. 1.1).

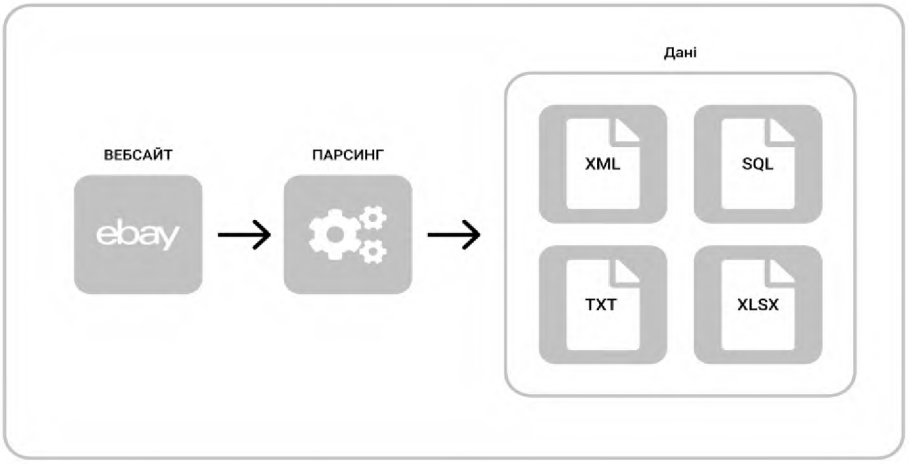


Рисунок 1.1 – Схема послідовності парсингу даних та їх збереження

Важливо зауважити, що парсинг та скрейпінг в сучасній термінології – слова-синоніми. Основним завданням технології скрейпінгу був пошук ресурсів у пошукових системах, на яких може зберігатися необхідна для користувача інформація, а парсинг своєю чергою робив усі інші маніпуляції з даними, такі як збір, обробка, збереження.

## 1.2 Метод обробки даних для здійснення парсингу на основі регулярних виразів

Зазвичай, у процесі парсингу даних виділяють три основних етапи: отримання доступу та завантаження даних, обробка їх для отримання потрібної інформації, збереження отриманої інформації в зручному для подальшого використання форматі [6]. Всі ці етапи реалізуються програмно усередині парсера.

На першому етапі відбувається сканування цільових вебсторінок. Парсер посилає багато HTTP(s) запитів на потрібні сторінки, зберігаючи отримані відповіді. При цьому список URL сторінок або задається заздалегідь, або формується в процесі сканування за заданим алгоритмом.

Найбільш важливим та технічно складним є другий етап. Він полягає в реалізації алгоритмів аналізу та відбору потрібного контенту з масиву даних, отриманих на першому етапі.

Існує кілька підходів до вирішення задачі відбору потрібних даних із завантажених вебсторінок. Вони відрізняються ступенем складності та застосовуються залежно від специфіки конкретної задачі.

Найпоширенішими підходами при розробці таких алгоритмів можна назвати: використання регулярних виразів, аналіз деревоподібної структури HTML шаблонів, завантаження сторінок за допомогою засобів автоматизованого керування браузерами, застосування технологій машинного навчання.

Щоб зрозуміти, як це працює, потрібно більш детально розібрати кожен з підходів по черзі.

Використання регулярних виразів є сильним інструментом для пошуку рядків, перевірки їх на відповідність будь-якому шаблону та іншій подібній роботі. Англомовна назва цього інструменту – Regular Expressions або просто RegExp [7]. Умовно кажучи, регулярні вирази – спеціальна мова для опису шаблонів рядків. Використання цього інструменту може відрізнятися в різних мовах програмування.

Першочергово варто зазначити, що будь-який рядок сам собою є регулярним виразом. Наприклад виразу, «система» буде відповідати рядок «система» і тільки він, тому що регулярні вирази залежать від реєстру символів. Тому рядок «Система» з великої літери вже не відповідає виразу вище і є іншим виразом. Також потрібно мати на увазі, що RegExp, як і будь-яка мова, мають спеціальні символи, які потрібно екранувати: . ^ $ * + ? { } [ ] \ | ( ).

Припустимо що необхідно знайти в тексті всі абревіатури. Якщо просто вказати в запиті ДСНС, то під нього не попадуть «ДПА», «ПДАУ». Та й проблему з реєстром першої літери треба якось вирішити.

У цій ситуації користувачу допоможуть набори: замість вказівки конкретного символу, користувач може записати цілий список, і якщо в досліджуваному рядку на вказаному місці стоятиме будь-який із перелічених символів, рядок вважатиметься відповідним. Набори записуються в квадратних дужках: патерну [ABCD] відповідатиме будь-який із символів "A", "B", "C" або "D".

Усередині набору більша частина спецсимволів не потребує екранування, проте використання перед ними не буде вважатися помилкою. Як і раніше, необхідно екранувати символи "\" і "^", і, бажано, "]".

Незвичайна на перший погляд поведінка регулярних виразів із символом «]» насправді визначається відомими правилами, але набагато легше просто екранувати цей символ, ніж їх запам'ятовувати. Крім цього, потрібно екранувати символ «–», який використовується для завдання діапазонів.

Якщо одразу після символу [ записати символ ^, то набір набуде зворотного змісту, тобто, відповідний буде вважатися будь-який символ крім зазначених. Так, патерну [^xyz] відповідає будь-який символ, крім, власне, "x", "y" або "z".

Отже, застосовуючи цей інструмент до нашого прикладу, якщо ми напишемо [П][ДАА]х[ААУ], то кожен з рядків «ПДАУ», «ДПАА», і навіть «УПЕТ» відповідатимуть шаблону. Також за допомогою регулярних виразів є можливість перевірити положення рядка щодо решти тексту.

Вираз \b означає межу слова, \B – не межу слова, ^ – початок тексту, а $ – кінець. Так, по патерну \bПДАУ\b у рядку "ПДАУ моя альма–матер" знайдуться перші 4 символи.

Приклад того, як працює парсинг регулярними виразами, зображено на рис. 1.2.


Рисунок 1.2 – Приклад регулярного виразу

У користувача може виникнути потреба позначити набір, до якого входять літери, наприклад, від "а" до "я". Замість того, щоб писати [абвгдеєжзи] можна скористатися механізмом діапазонів і написати [а-и]. Так, патерну x[0-8A-F][0-8A-F] відповідає рядок «xA6», але не відповідає «xb9» (див. рис 1.2).

Для деяких наборів, які використовуються досить часто, є спеціальні шаблони. Так, для опису будь-якого символу переносу (пробіл, табуляція, перенесення рядка) використовується \s, для цифр – \d, для символів латиниці, цифр та підкреслення "_" – \w.

Якщо необхідно описати взагалі будь-який символ, то для цього використовується крапка. Якщо вказані класи написати з великої літери (\S, \D, \W), то вони змінять свій сенс на протилежний – будь-який непереносний символ, будь-який символ, який не є цифрою, та будь-який символ крім латиниці, цифр або підкреслення відповідно.

Третій етап полягає у приведенні в зручний вигляд вже одержаних корисних даних. На цьому етапі дані очищаються від непотрібних елементів, кластеризують, якщо необхідно додатково видозмінюються та форматуються.

Четвертий етап – збереження даних в необхідний для користувача формат. У легкому випадку дані будуть збережені у текстовому документі або таблиці Excel. Але в більшості своїй масив серіалізується відповідно до певної моделі і зберігається до бази даних.

Кожен із цих етапів необов'язково має бути яскраво виражений. У складі парсера один модуль може виконувати кілька функцій, наприклад, форматування і збереження даних у потрібному форматі.

## 1.3 Сучасні технології здійснення парсингу на основі готових рішень

На даний час існує вже чимало підходів до реалізації парсингу. Основним підходом завжди вважався метод розробки парсера кодом на основі застосування для цього зручної мови програмування та додаткових бібліотек. Користувач у такий спосіб завжди має можливість створити парсер, котрий буде застосовано для виконання одноманітної задачі, тобто планового знаходження та обробки інформації з вебсторінки.

В якості прикладу можна навести задачу, коли користувач бажає розмістити на своєму вебсайті поточний курс валют: для цього розробляється парсер, який послідовно та заплановано буде брати дані про курс з офіційного сайту Національного банку України (НБУ), після чого передавати ці дані до бази даних (БД) користувача сайту і показувати курс на сайті. Проблеми цього рішення в тому, що такий парсер є одноманітним і завжди послідовно виконує одну й ту саму задачу. Разом із тим, для створення парсера потрібні знання в області програмуванні, яких може не бути у користувача.

Іншим підходом є використання готового рішення для парсингу, оскільки на сьогодні ринок програмних застосунків для парсингу достатньо насичений. Чимало компаній вбачають у парсингу перспективну технологію, адже з кожним днем вебсайтів стає все більше, як і інформації, яку вони розміщують. Разом із цим росте й попит на технології парсингу та автоматизацію вебпроцесів.

Станом на 2023 р. на ринку готового програмного забезпечення для парсингу інформації та автоматизації вебпроцесів вже існують безумовні лідери такі як: Scraping-Bot, Scrapeworks, Diggernaut, ScrapingBee, Mozenda, Content Grabber, FMiner, ZennoPoster та інші.

Не варто забувати й про бібліотеки з відкритим вихідним кодом для ручної розробки парсерів, серед яких також є безумовні лідери. Для мови програмування Python найбільш популярними є: BeautifulSoup, Selenium, Lxml. Для мови програмування JavaScript варто виділити: Cheerio, Osmosis, Apify SDK. Бібліотеки набирають популярності завдяки значній економії часу та досягнення якості при написанні коду.

Серед перелічених готових рішень для подальшого вивчення було вибрано рішення, яке здалося максимально корисними із міркувань ефективності на витрачений час – це відоме рішення ZennoPoster.

Щодо ручної розробки, то підхід до парсингу, який розглядається – це розробка програми на мові програмування Python із залученням інструменту Selenium та вебдрайвера ChromeDriver.

Важливо зазначити те, що інструмент Selenium був створений для автоматизації вебпроцесів [8], основними його функціям є взаємодія з браузером, а також такі дії:

– виконання емуляції кліку по елементу;

– заповнення форм різної складності;

– прокручування сторінки.

Іншими словами Selenium під управлінням мови програмування Python вправно імітує поведінку користувача в браузері. Він надає розширення для емуляції взаємодії користувача з браузерами, сервер розповсюдження для масштабування розподілу браузера та інфраструктуру для реалізації специфікації W3C WebDriver, яка дозволяє писати взаємозамінний код для всіх основних веббраузерів, таких як Google Chome, Mosilla Firefox, Opera, Chromium та інші. Цей продукт став можливим, завдяки волонтерам, які витратили тисячі годин власного часу та зробили вихідний код у вільному доступі для будь-кого.

Сам по собі Selenium – це просто інструмент, що використовується у автоматизації дій веббраузера. У більшості випадків його використовують в тестуванні вебдодатків, але цим не обмежується. Зокрема, його також використовують в рутинних задачах на кшталт адміністрування вебсайту, або для регулярного парсингу даних з різних джерел.

Інструмент Selenium використовується для посилання HTTP запитів драйверу за допомогою протоколу JsonWire Protocol, у яких зазначено дію, яку має зробити браузер у рамках поточної сесії, архітектура роботи інструменту Selenium зображена на рис. 1.3. принцип дії описується наступним чином: клієнт Selenium у поєднанні з однією з мов програмування, які підтримуються бібліотеками інструменту, звертається до драйвера браузера за допомогою JsomWireProtocol із запитом на кшталт «отримати заголовок H1 на сторінці», далі вебдрайвер звертається до свого браузера та отримує від нього повністю завантажену вебсторінку. Після цих маніпуляцій за допомогою регулярних виразів Selenium отримує вміст тегу H1 та може його обробляти.

Однією з переваг Selenium є те, що він по праву може називатись кросплатформенним інструментом, адже його розробку підтримують більшість відомих браузерів та операційних систем [8].

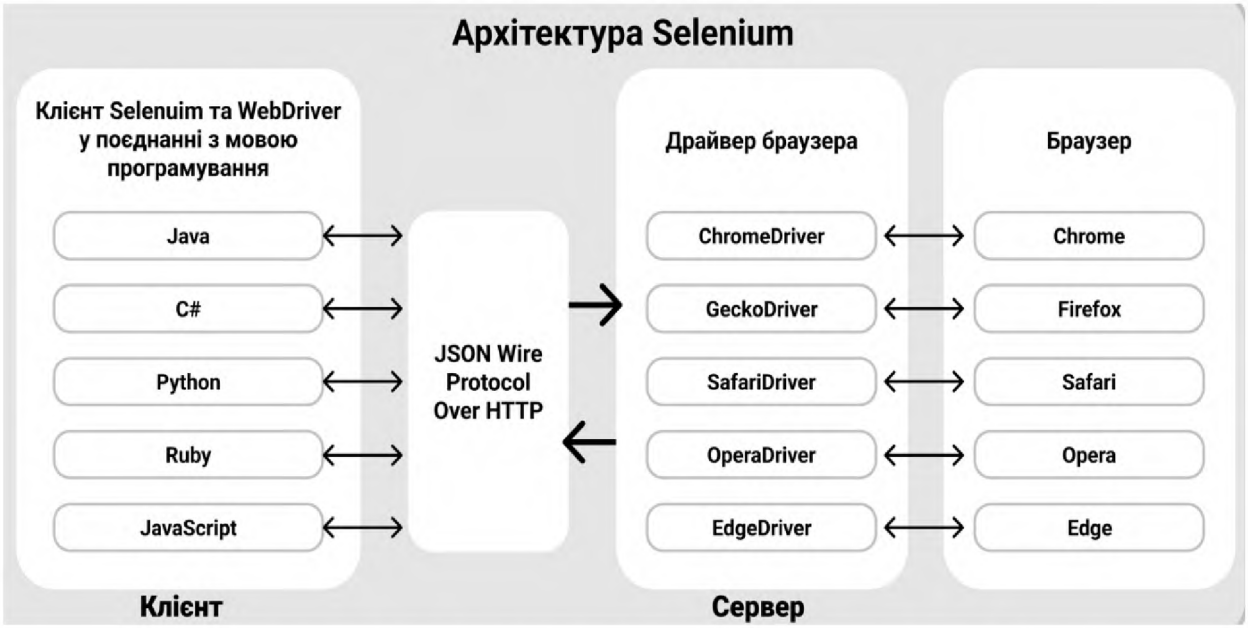


Рисунок 1.3 – Архітектура процесів інструменту Selenium

У рамках проєкту Selenium почали розробку та випуск інструменту «Selenium IDE» – це розширення для веббраузера Mosilla Firefox, що є бібліотекою Selenium з графічним інтерфейсом GUI, побудованим з використанням XUL [9]. Розширення дозволяє запис, збереження та відтворення сценаріїв тестування та парсингу вебсторінок. Сценарії зберігаються у форматі HTML як таблиці.

Серед недоліків Selenium можна вважати той факт, що простий користувач ПК не має змоги швидко навчитись використовувати цей інструмент, адже для цього потрібні детальні знання та навички в програмуванні однією з доступних мов. Сам інструмент не має графічного інтерфейсу.

Наступним підходом для парсингу, який заслуговує на увагу, є застосування готового рішення ZennoPoster від виробника програмного забезпечення ZennoLab. Основні задачі, які виконує ZennoPoster, - автоматизації дій у браузері [10]. За лічені хвилини можна автоматизувати будь-яку роботу в браузері, яку раніше виконували вручну. На відміну від методу Selenium + Python, ZennoPoster має зручний графічний інтерфейс, який до того ж є інтуїтивно зрозумілим для користувача (рис. 1.4).

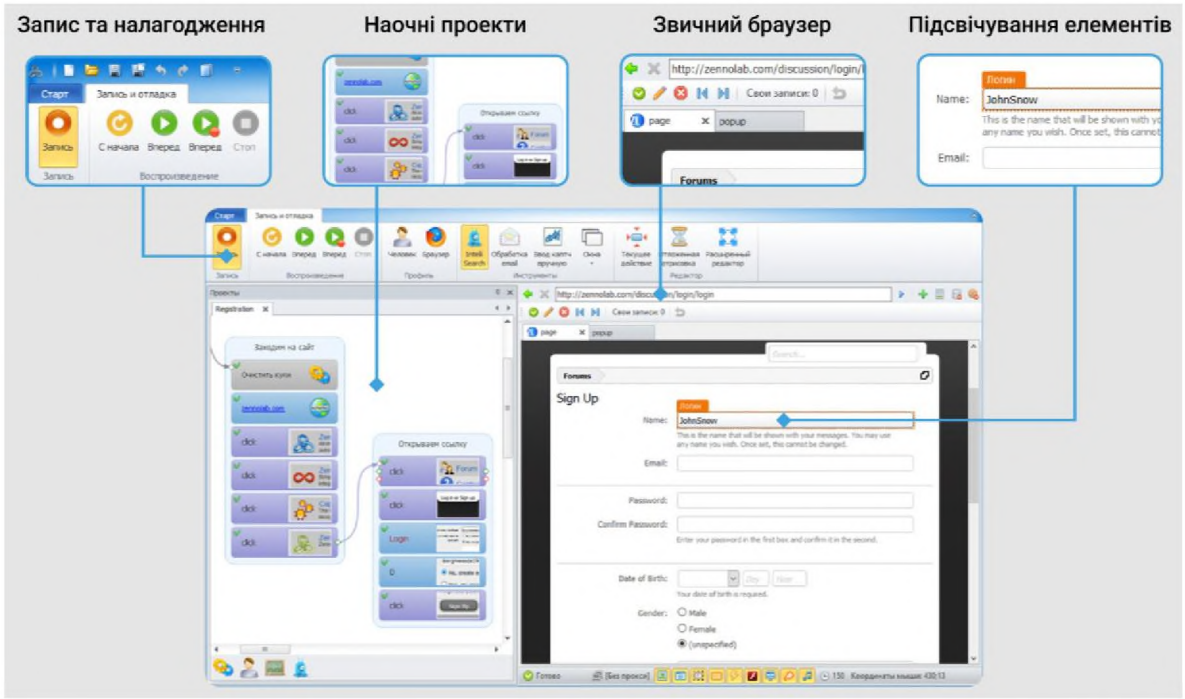


Рисунок 1.4 – Графічний інтерфейс програмного середовища ZennoPoster

До основних переваг ZennoPoster відносять [11]:

– багатопоточність: ZennoPoster дозволяє працювати в браузері у багатопотоковому режимі. У кожному потоці незалежні cookie, кеш та проксі. Це прискорює роботу у десятки разів;

– зручний інтерфейс, завдяки візуальному drag–and–drop інтерфейсу з програмою працювати легко;

– автозапис дій: робота програми ZennoPoster базується на виконанні шаблону (набору дій). В програмі створена унікальна система автоматичного запису шаблону за діями користувача у браузері. Від користувача знадобиться мінімум зусиль для автоматизації своєї роботи;

– гнучка логіка, шаблони підтримують логічні розгалуження. Користувач зможе створювати цикли, перевірки, перемикання та керувати ходом виконання шаблону за допомогою логічних дій. Це дозволяє створювати гнучкі рішення для завдань користувача;

– потужний ProxyChecker, користувач має змогу збирати безкоштовно тисячі живих проксі та вибирати найкращі з них, використовуючи велику кількість різноманітних фільтрів та правил. Це дозволить користувачеві зберігати анонімність на всіх етапах роботи;

– система емуляції людини та браузера, програма створює профіль людини для роботи в Інтернеті. Ім'я, прізвище, дата народження, e-mail, стать, національність – всі ці дані автоматично створюються програмою. Користувач не буде турбуватися про більшість видів захисту від ботів – програма обходитиме їх автоматично;

– робота зі списками, таблицями, Google Таблицями, базами даних та текстовими файлами, у програмі вбудовані засоби для роботи зі списками, таблицями (xlsx, ods, csv, можна задати свій формат), Google Таблицями, простими файлами та базами даних;

– підтримка свого коду, спеціально для користувачів, які знають мову програмування C#, створено класи управління браузером із написанням свого коду, тобто, замість шаблону дій користувач має змогу написати свою програму,

яка повністю управляє браузером. Також користувач зможе використовувати всі переваги, які йому дає ця мова і одночасно повноцінно керувати браузером;

– зовсім не давно з'явилася можливість інтеграції з Visual Studio.

При використанні готових рішень для парсингу, важливо враховувати те, що парсинг як процес потребує обчислювальних потужностей від сервера чи комп'ютера користувача, у якого відбувається парсинг. Від потужностей та можливостей напряму залежить те, з якою швидкістю програма матиме змогу на обчислення даних. Виробники програмного забезпечення завжди вказують рекомендовані системні вимоги. У технічній документації програмного забезпечення ZennoPoster, розробники зазначили мінімальні та рекомендовані системні вимоги.

Мінімальні системні вимоги наступні [12]:

– процесор: Intel чи AMD, 2 ядра та не менше 1.5 Ггц на кожне з ядер;

– RAM: не менше 2048 Мб;

– операційна система: Microsoft Windows 7 та старші, важливо зазначити що збірка «Server Core» не підтримується виробником ПЗ;

– програмні платформи: .Net Framework 3.5 та Visual C++ redistributable 2008, 2010, 2013 x86 або x64 в залежності від бітності операційної системи;

Рекомендовані системні вимоги наступні:

– процесор: 8 – 16 ядер, Intel чи AMD;

– RAM: 16 Гб;

– програмні платформи: .Net Framework не старішій версії 4.6.2, Direct X починаючи з версії 9+.

Отже, беручи до уваги те, що програмне забезпечення ZennoPoster є надійним та потужним інструментом для парсингу, його застосування слід вважати пріоритетним для використання в задачах збору, обробки, аналізу та зберігання даних.

Таким чином, проведений аналіз технологій для парсингу визначив задачу досліджень, яка полягає в розробці пропозицій щодо реалізації технології парсингу з застосуванням програмного забезпечення ZennoPoster.

# РОЗДІЛ 2
# АНАЛІЗ ІНСТРУМЕНТАРІЮ ТА ОБГРУНТУВАННЯ ВИБОРУ РІШЕНЬ ДЛЯ ПАРСИНГУ

## 2.1 Функції та можливості програмного комплексу ZennoPoster

Програмний комплекс ZennoPoster – це програмний пакет, що складається з кількох програм, вікна яких зображені на рис. 2.1, і які дозволяють автоматизувати дії користувача у веббраузері [11]. Це програмне забезпечення є корисним для різних категорій професійних користувачів – як для SEO спеціаліста, який займається просуванням в мережі, так і для досвідченого вебпрограміста, який кодує складних роботів.

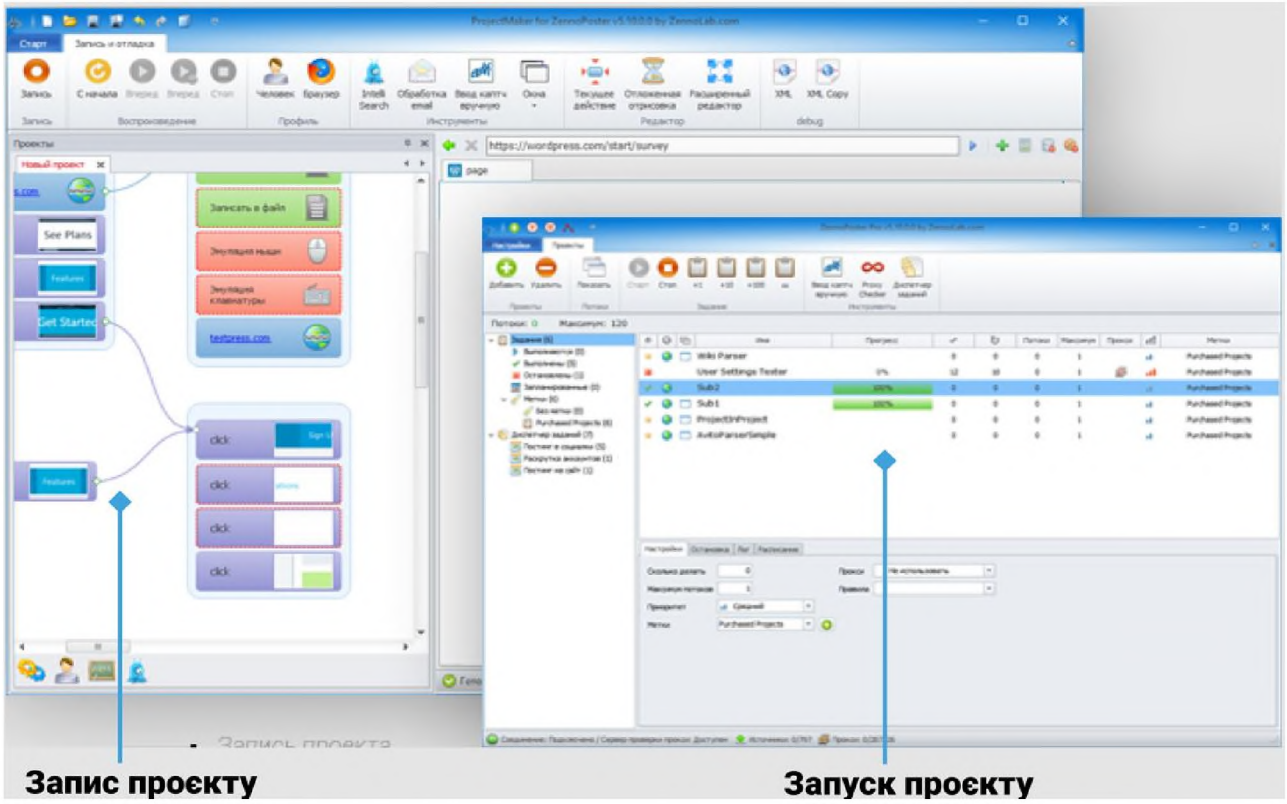


Рисунок 2.1 – Порівняння інтерфейсів ZennoPoster та ProjectMaker

Пакет ZennoPoster включає програми, опис яких наведено нижче.

1. ZennoPoster – основна програма, призначена для виконання автоматизованих шаблонів дій у браузері. Ця програма дозволяє керувати

потоками при виконанні шаблонів, використовувати різні проксі для користувацьких роботів, а також має вбудований планувальник для виконання шаблонів за розкладом.

2. ProjectMaker (PM) – програма для створення автоматизованих шаблонів дій (див. рис 2.1). Саме в ньому користувач буде записувати та автоматизувати свої дії на вебсторінках. Розробити бота досить легко – програма має набір стандартних дій (екшенів) для веббраузера, блоки яких конструюють шаблон, а також режим запису дій, який записує в шаблон всі Ваші дії в браузері. Додаток також дозволяє працювати зі списками, таблицями, Google таблицями, базами даних, текстовими та іншими файлами на персональному комп'ютері та підвантажувати їх у веб, значно спрощуючи роботу SEO-фахівців.

Розподілення програм є логічним. Адже після розробки та тестування бота, користувачу не потрібно використовувати графічний інтерфейс програми для розробки. Таким чином, PM можна закрити аби направити потужність системи на необхідні обчислення.

Безпосередньо знайомство користувача з програмним комплексом ZennoPoster розпочинається з застосунку ProjectMaker, адже саме в цьому застосунку виконується розробка ботів (рис. 2.2).

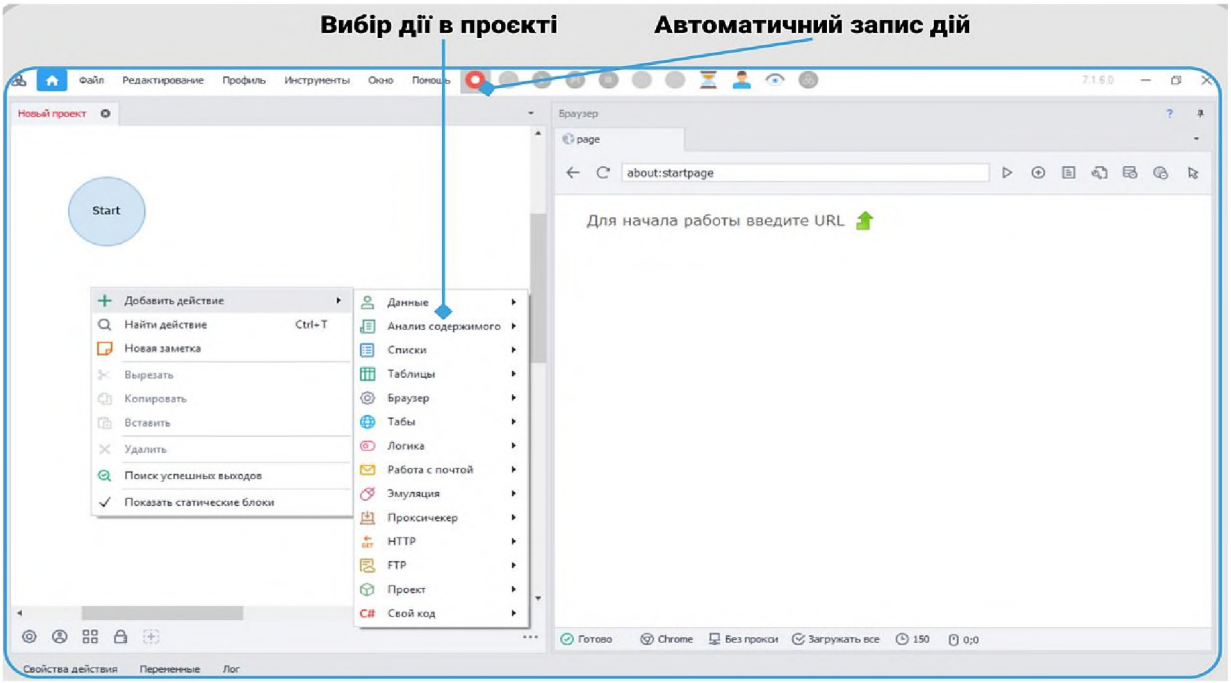


Рисунок 2.2 – Перші дії в застосунку ProjectMaker

При першому запуску застосунку, автоматично вмикається функція запису дій (див. рис. 2.2). Це зроблено для того, щоб автоматизувати відтворення певних дій надалі і не витрачати час на налаштування в ручному режимі. Автоматичний запис дій записує безпосередньо дії які користувач виконував у браузері (рис. 2.3), такі як:

– відкриття сторінки у браузері;

– друкування тексту (для подальшої емуляції);

– кліки мишею;

– перехід на інші сторінки веб-сайту.

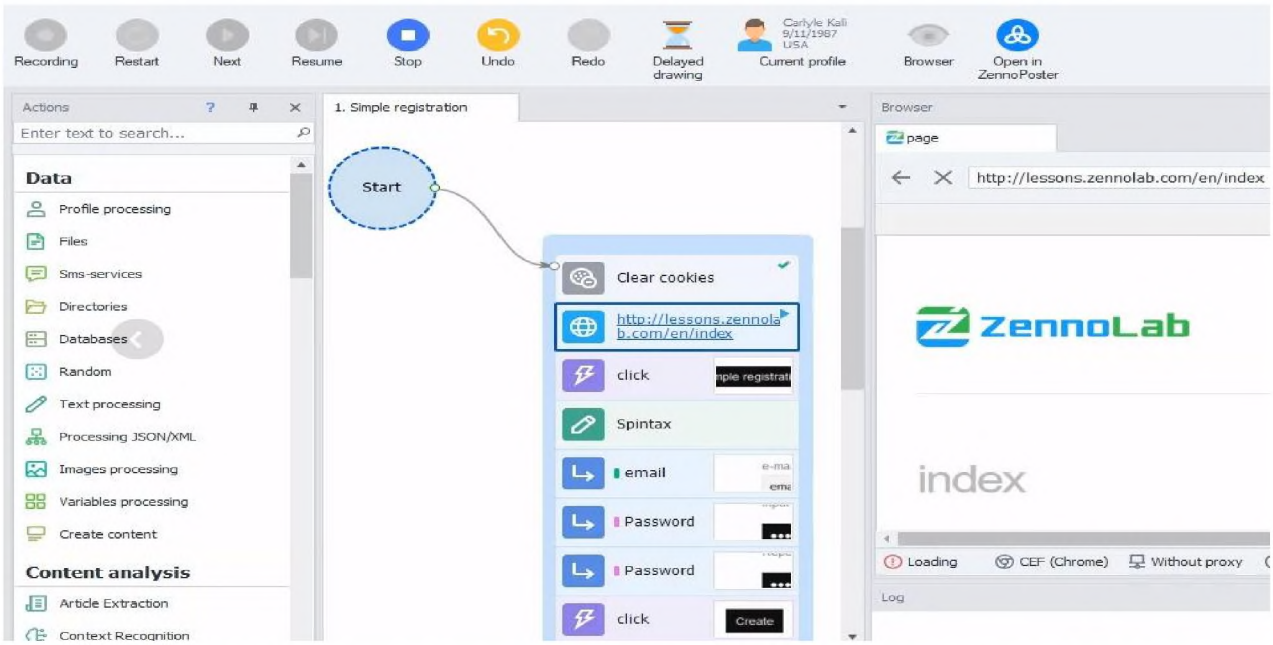


Рисунок 2.3 – Автоматично створені дії після використання браузера

Запис дій користувача є корисним функціоналом, але він не повною мірою покриває потреби в розробці ботів та більше підходить для незначної автоматизації дії в браузері. Рекомендується на початку вимкнути цю функцію та робити покрокову розробку аби надалі мати змогу більш точно коригувати дії бота та не гаяти час на неточності.

Основне контекстне меню в PM відкривається якщо на робочому просторі застосунку зробити клік правою кнопкою миші зображено на рис. 2.4. До контекстного меню входять такі функції як: додавання дій до проєкту, пошук дій.

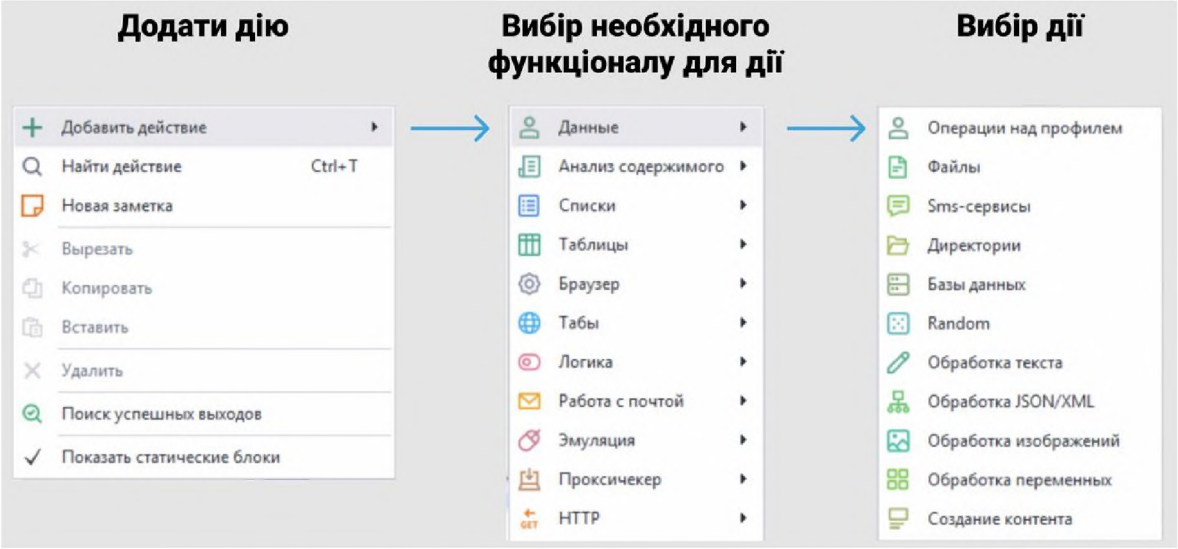


Рисунок 2.4 – Меню додавання дій для бота

ZP вміщує в себе перелік основних функцій. У процесі розробки в проєкті виконуються основні дії, зміст яких наведено нижче.

1. Взаємодія з даними, вміщує в себе функції обробки даних та роботу з профілем користувача який будуть бачити вебсайти, інакшими словами за допомогою функції «операції над профілем» можна тонко налаштувати:

– зберігання proxy;
– зберігання налаштуваннь додатків браузера (плагіни);
– встановлювати часовий пояс;
– збереження геолокації;
– збереження файлів cookie.

Налаштувавши та зберігши такі дії можна створити картину користувача для сайту аби надалі сайт розумів що ним користується людина, а не бот, та не заблокував сеанс користувача розцінивши його дії як шкідливі.

2. Аналіз вмісту допоможе користувачу визначити тему тексту веб-сторінки або просто текст, який користувач покаже. Замість того, щоб розсилати посилання на всі ресурси поспіль, користувач може спочатку визначити тему тексту сторінки, на яку хочете зробити посилання.

Так само користувач може знайти посилання на своєму сайті та перевірити тему сторінок, на які вони посилаються.

Припустимо, що у користувача є база посилань для постів. За допомогою функції Context Recognizer можна розділити свою базу даних на кілька баз даних за контекстом [13]. У той час, коли користувачу знадобиться реклама сайту, він візьме не повну базу, а ту, яка відповідає тематиці сайту. Користувач зможе опублікувати статтю про автострахування в автомобільному блозі, а не в тому, який публікує анонси нових фільмів.

Користувач може проаналізувати сайт (наприклад, блог) та знайти сторінки, які найкраще відповідають темі вашого оголошення. Залишаючи актуальні коментарі та пости, користувач отримує тематичні посилання;

3. Взаємодія зі списками переважно використовуються для отримання рядків даних із текстового документа або записів даних у файл. Припустимо, є файл зі списком URL-адрес і користувачеві потрібно отримати доступ до них по одному, або, наприклад, розбір якогось значення із сайту потребує їх відсортувати. Однак, це не весь функціонал взаємодії зі списками (рис. 2.5), PM підтримує ще багато інших дій:

– додавання та отримання елементів списку;
– видалення рядків та дублікатів;
– прив’язка до файлу;
– отримання кількості рядків;
– перемішування;
– сортування значень.

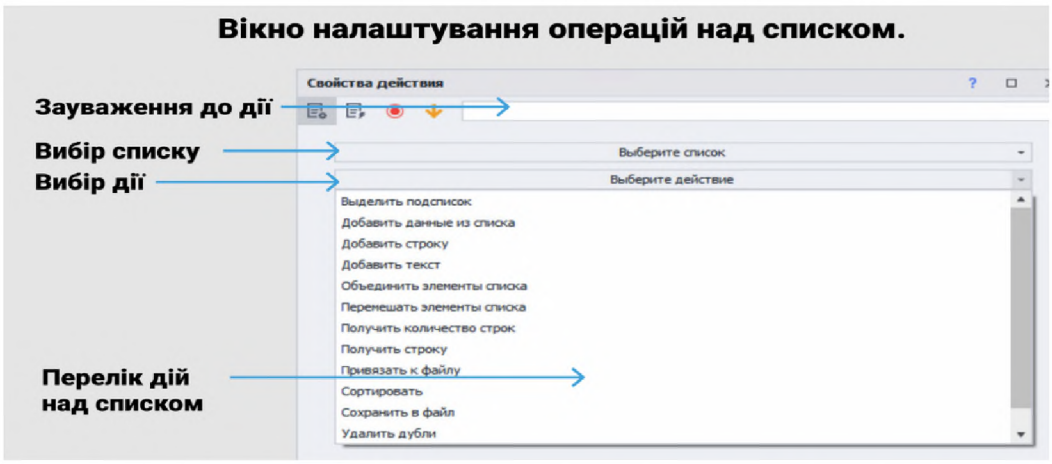


Рисунок 2.5 – Налаштування операцій над списком

Додати операцію над списком можна через контекстне меню, далі необхідно вибрати пункт додати дію, вибрати підрозділ контекстного меню списки та натиснути на кнопку «Операції над списком». Після наведених дій відкриється вікно налаштування операцій над списком (див. рис. 2.5).

4. Взаємодія з таблицями є важливим функціоналом РМ, адже завдяки таблицям у користувача є можливість на управління отриманими даними. Таблиці використовуються для отримання більш складно організованих даних, ніж списки, (наприклад, список товарів для інтернет-магазинів, де в кінцевому підсумку представлені різні дані: назва, ціна, опис і т. д.). Створити та налаштувати операції із таблицями можна тим самим чином як і із списком, РМ дає змогу тонко налаштувати роботу із таблицями у вікні операцій із таблицями (рис. 2.6).

5. Взаємодія з браузером.

6. Таби (управляння даними).

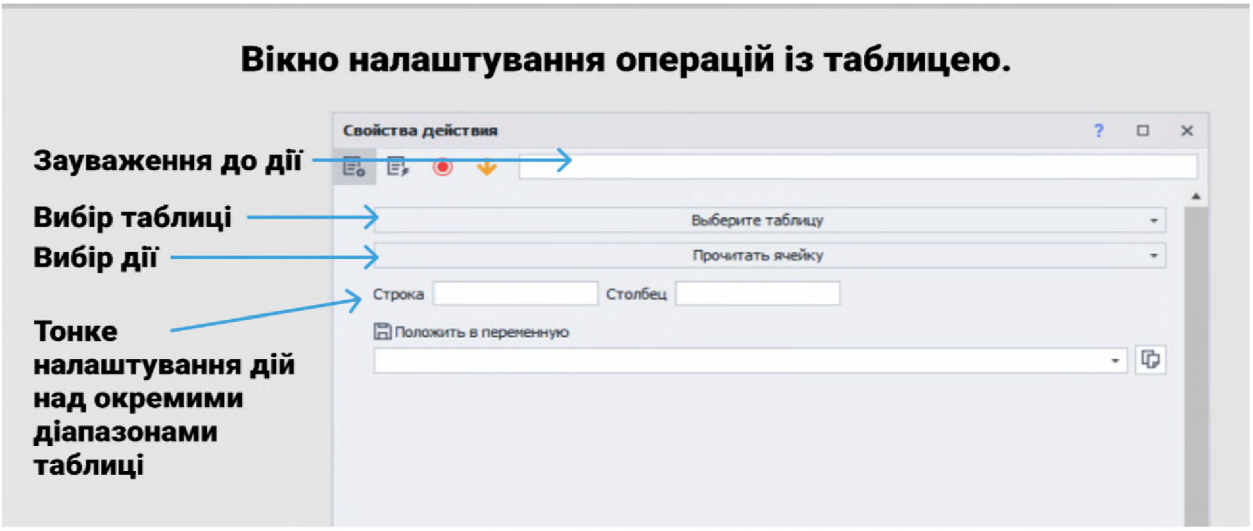


Рисунок 2.6 – Вікно операцій із таблицею

7. Логіка (налаштування змінних, та логічні перемикачі дій) Робота зі змінними в РМ варта окремої уваги, адже вони основа будь-якого проєкту у програмуванні, в порівнянні з традиційним програмуванням де кожну дію зі змінними потрібно налаштовувати окремо в застосунку ProjectMaker основні операції зі змінними автоматизовані.

Отже, користувачу для передачі даних із комірки таблиці в змінну не потрібно писати алгоритм достатньо всього декількох кліків і операція буде успішно завершена, а данні перенесені. Вікно управляння змінними можна викликати із контекстного меню. Змінні в РМ мають пряму інтеграцію зі списками та таблицями, таким чином в налаштування операцій зі списком чи таблицею можна налаштувати передачу даних на пряму зі змінною та навпаки.

Змінна – це контейнер у пам'яті, який може приймати задане або значення, що обчислюється. Змінні в ZennoPoster можна створювати, змінювати назву змінної та видаляти, а також надавати їм різні значення. Найзручніше працювати зі змінними через Вікно змінних (рис. 2.7).

Слід розділяти змінні C# , які строго типізовані та змінні проєкту, які не мають строгої типізації. Однак значення цих двох типів змінних завжди можна конвертувати без втрати та спотворення даних.

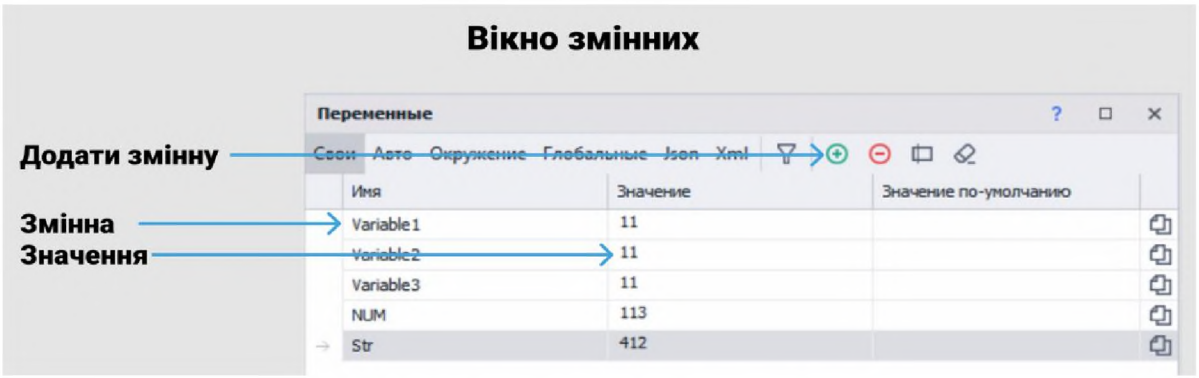


Рисунок 2.7 – Вигляд вікна змінних

Функції в в ZennoPoster:

- взаємодія з поштою;
- емуляція (налаштування для емуляції дії з клавіатури чи миші);
- проксічекер (функціонал для анонімізації дії в інтернеті);
- взаємодія з запитами HTTP (GET запит, POST запит, HTTP запит);
- взаємодія з FTP (протокол передачі файлів у мережі);
- взаємодія з власним кодом мовами програмування (C#, JavaScript).

Таким чином, визначено інструментарій подальшої роботи над парсингом.

## 2.2 Порівняння технологій створення ботів для парсингу

Сьогодні потрібні рішення, які значно ефективніше справляються з буденними задачами, і саме в цей момент на п'єдестал виходять автоматизовані системи, або іншими словами боти. Розробка ботів – це доволі важливий та необхідний крок для людства, адже шукати, обробляти, аналізувати та розповсюджувати інформацію набагато легше, швидше та ефективніше, автоматизувавши цей процес.

Якщо розібратися детальніше, то ботами можна назвати автоматизовану конвеєрну стрічку на виробництві, яка оснащена датчиками, які своєю чергою аналізують продукцію на конвеєрній стрічці шукають неточності чи помилки аби вчасно виправити помилку. Ботом також можна назвати автоматизований чат, такі системи ще називаються чат-ботами. Його основна задача полягає в тому, щоб дати відповідь на запитання користувача чи інші функції, для яких його розробили.

На відміну від процесів виробництва боти для парсингу не потребують фізичних датчиків чи якогось точного місця розташування або якоїсь специфічної розробки. Боти для парсингу можна створити багатьма методами, чи підходами з застосуванням різних стеків технологій.

Основним із підходів розробки ботів для парсингу завжди був та залишається метод розробки кодом, тобто програміст власноруч «пише» бота, зручною для цього мовою програмування, та на свій розсуд визначає необхідні функції чи технології які буде застосовувати.

Перед попереднім методом у програмного засобу ZennoPoster є величезна проблема, адже це ПЗ перевантажене надлишковими функціями які заважають недосвідченому розробнику створювати бота з максимальною швидкодією та мінімальною кількістю кроків.

Велика кількість влаштованих розробниками ПЗ функцій створює значну проблему з вибором обладнання для серверу та обчислювальними можливостями системи. Адже бот, який написаний методом розробки кода

власноруч, може працювати навіть на мінікомп'ютерах типу «Raspberry Pi» [14], через те що в цьому боті є лише важливі та необхідні функції які за часту не потребують великих обчислювальних можливостей на відміну від ПЗ ZennoPoster який у свої роботи ще на старті потребує серверне обладнання з середніми обчислювальними можливостями.

ПЗ ZennoPoster на відміну від розробки кодом, буде вимагати все більше обчислювальних можливостей в міру збільшення кількості дій парсингу чи автоматизації функції, чим більше кроків робить бот, тим більше дій він обраховує [15], тим більше потужностей потребує (рис. 2.8).

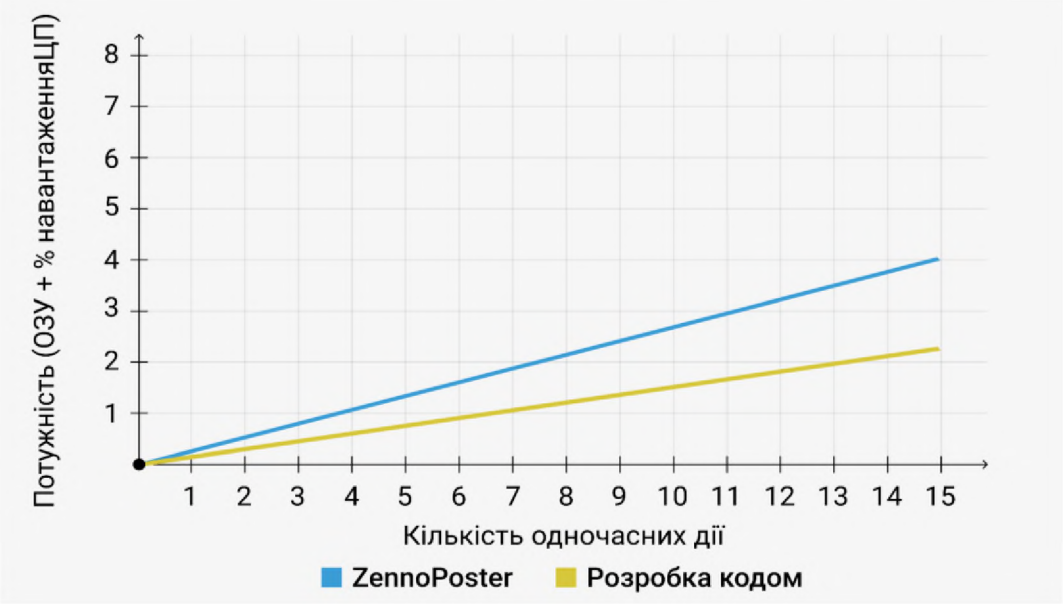


Рисунок 2.8 – Порівняння залежності підходів розробки ботів від розрахованих можливостей серверу

Зокрема, ще одна важлива проблема двох підходів це наявність графічного інтерфейсу. Те, що для користувачів видається великою перевагою, може бути проблемою для можливостей сервера, адже частина потужностей йде на те, щоб вивести прорахунок графічного інтерфейсу.

Порівнняння ZP та власне написання бота кодом є не зовсім об'єктивним, адже попри однотипне застосування це кардинально різні підходи, кожен з них вимагає знання та рішення які по різному виконують ті чи інші дії.

Тому до порівняння буде більш чесно додати веб інструмент Integromat. Integromat – це потужна інтеграційна платформа, яка дозволяє візуалізувати, проєктувати та автоматизувати свою роботу за лічені хвилини [16]. Це аналог ZP, але в вигляді веб сайту, він має багато можливостей, які вміщує в себе ZP.

На відміну від ZP, Integromat не потребує від користувача завантажувати програмне забезпечення та турбуватися за вираховні можливості, всі обрахунки відбуваються на серверах сервісу.

У порівнянні з ZP у сервісу Integromat існують недоліки, основним недоліком вважається те що для доступу до сервісу необхідне підключення до всесвітньої мережі internet в той момент як ZennoPoster може працювати без підключення з завчасно завантаженими веб сторінками.

Наступний недолік полягає в тому що Integromat хоч і має безплатний тарифний план, але він обмежений в кількості дій на рік які може виконати бот [17]. Зокрема його вистачає в тому випадку якщо користувач використовує Integromat в якості сервісу для проєктних задач.

## 2.3 Обчислювальні можливості серверів для парсингових ботів

Порівнявши технології та рішення для створення ботів, з'являється розуміння, що боти залежні від обчислювальних можливостей серверів чи комплексів. У в той час, коли деякі сервіси надають можливості хмарного сховища та хмарних обчислень, інші потребують локального розгортання ПЗ на сервері чи персональному комп'ютері користувача. Загалом при створенні ботів шляхом застосування спеціального ПЗ рекомендується створити надлишкову потужність серверу.

Пріоритетним обладнанням в створенні ботів для парсингу є серверне, оскільки сервери створюються за для безперебійної роботи 24 години 7 днів на тиждень [18], важливо зазначити, що мова йде перш за все про постійну роботу ботів, не одного, а декількох у багато потоковому режимі.

Робота ботів, перш за все, потребує надлишковості оперативної пам'яті та обчислювальних можливостей центрального процесора, тому фахівці рекомендують використовувати процесори з кількістю ядер не менше 4 з тактовою частотою від 3000 МГц на одне ядро. Якщо користувач має можливість використовувати професіональне серверне обладнання рекомендується звернути увагу на сервери з багатопроцесорною архітектурою, це створить надлишковість обчислювальних можливостей для розгортання ботів.

Для створення ботів часто звертаються до сервісів оренди. Це роблять тоді коли користувач не хоче власноруч керувати сервером та налаштовувати його.

VPS (Virtual Private Server) та VDS (Virtual Dedicated Server) є схожими концепціями в галузі віртуалізації серверів, де один фізичний сервер розділяється на окремі віртуальні сервери. Використання VDS для створення ботів є більш економним явищем, адже оренда VDS в деяких випадках може бути вигіднішою, аніж використання персонального сервера, у зв'язку з використанням великої кількості електроенергії.

Послуги з оренди віртуального виділеного сервера VPS або VDS можна замовити у хост-провайдера [19]. Провайдер має дата-центр, тобто приміщення, де знаходиться багато серверів. Ресурси кожного сервера ділять на частини і запускають кілька незалежних один від одного віртуальних серверів VPS/VDS. За функціоналом вони мало відрізняються від фізичних серверів, схема роботи VDS зображена на рис. 2.9.

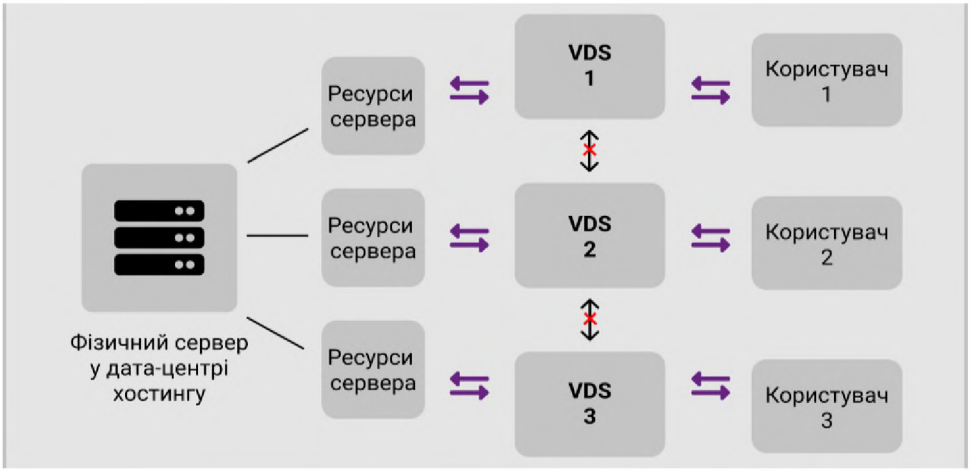


Рисунок 2.9 – Схема роботи VDS

Також VDS вибирають ще й тому що користувач може більш тонко використовувати ресурси та розраховувати на них, тобто завдяки послузі з оренди завжди можна розраховувати на те що в разі потреби ви отримаєте рівно стількипотужності, скільки необхідно для виконання тієї чи іншої задачі.

На сьогодні більшість розробників сайтів намагаються створити обмеження або перепони для автоматичного збору даних (парсингу) [20], часто це готові рішення або власні розробки, котрі або повністю унеможливлюють процес автоматичного збору, або ж сповільнюють його. Більшість з них є недостатньо ефективними або навіть застарілими. Серед таких обмежень є:

– блокування сеансу за user-agent. При зверненні на сайт парсер відсилає запит user-agent, у якому повідомляє сайту інформацію себе. Деякі сайти відразу блокують доступ парсерів, які в user-agent представляються як програми. Заміна користувача-агента цілком вирішує це обмеження;

– заборона у robots.txt. Наприклад, у robots.txt може бути прописана заборона індексування якихось розділів для Google-бота. Якщо користувач налаштує user-agent як Googlebot, то спарсити інформацію з цього розділу неможливо;

– блокування за IP адресою. Якщо бот тривалий час займається парсингом сайту, то його можуть заблокувати на певний або невизначений час.

Існує два варіанти вирішення:

1) використовувати VPN;

2) у налаштуваннях парсера зменшити швидкість, щоб не робити зайве навантаження на сайт і зменшити ймовірність блокування;

– деякі сайти захищаються від парсингу за допомогою розумного аналізатора активності. Якщо дії користувача схожі на роботизовані, то аналізатор може заблокувати чи обмежити доступ до сайту;

– використання на сайті технології «ReCAPTCHA» (рис. 2.10).

ReCAPTCHA – безкоштовний сервіс, який захищає ваш сайт від спаму та зловживань. Він використовує передові методи аналізу ризиків, щоб відрізнити

людей від роботів [21]. Загалом, це – найпоширеніші процеси по блокуванню роботи парсерів.

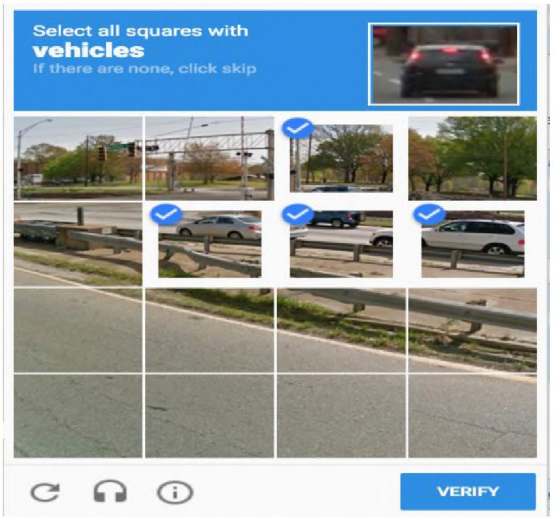


Рисунок 2.10 – Приклад технології ReCAPTCHA під час перевірки користувача

Завдяки тому, що названі процеси по блокуванню парсерів вже достатньо вивчені, більшість парсерів створюються із наперед визначеними діями для їх обходу. Зокрема в програмному забезпеченні ZennoPoster влаштовано багато функцій для обходу обмежень, в тому числі описана ReCAPTCHA.

Отже, при порівнянні технологій та рішень для створення ботів було виявлено, що розробка ботів за допомогою готових рішень є більш простим та, в порівнянні з ручним написанням коду, значно швидшим методом. Однак, у зв'язку з перенасиченістю технологіями, готові рішення мають значний недолік у зв'язку з тим, що кожна з функцій потребує врахування можливостей і тягне за собою здорожчання надлишкових потужностей сервера.

При розробці бота для парсингу важливим етапом є знайомство з обмеженнями для парсингу, адже якщо такі обмеження не враховувати при розробці бота, зібрана ним інформація може бути не повною чи взагалі відсутньою у зв'язку з блокуванням бота на сайті системою розпізнання бота.

При створенні бота рекомендується звернутися до постачальників VPS з метою оренди серверних потужностей, що дасть можливість розраховувати на необхідну для бота розрахункову потужність та значно зменшить витрати, які б з'явилися при застосуванні особистого сервера.

# РОЗДІЛ 3
# РОЗРОБКА БОТІВ ДЛЯ ВЕБПАРСИНГУ У ПРОГРАМНОМУ КОМПЛЕКСІ ZENNOPOSTER ТА ІНТЕГРАЦІЯ З TELEGRAM BOT API

## 3.1 Послідовність розроблення бота для парсингу інформації

Для реалізації завдання кваліфікаційної роботи розроблено бота для парсингу інформації на прикладі онлайн маркетплейсу olx.ua [22]. Цей сервіс обрано з міркувань його популярності: сайт є лідером за кількістю аудиторії в Україні, як платформа для продажу товарів та послуг. Саме з цього сайту часто беруть інформацію про ціноутворення вторинного ринку або ринку послуг. Важливо зазначити, що дослідження мають виключно науковий характер для розуміння процесу розробки та використання ботів для парсингу і не вносять змін до роботи сервісу.

Перед створенням бота необхідно первинно дослідити сайт та технології які він використовує, визначитись з полями та даними, які необхідно отримати або функції, які необхідно додати до проєкту. На початку відкриваємо вебсайт та окреслюємо модель інформації, щоб надалі чітко розуміти необхідні елементи. Роботу розпочато з головної сторінки (рис. 3.1).

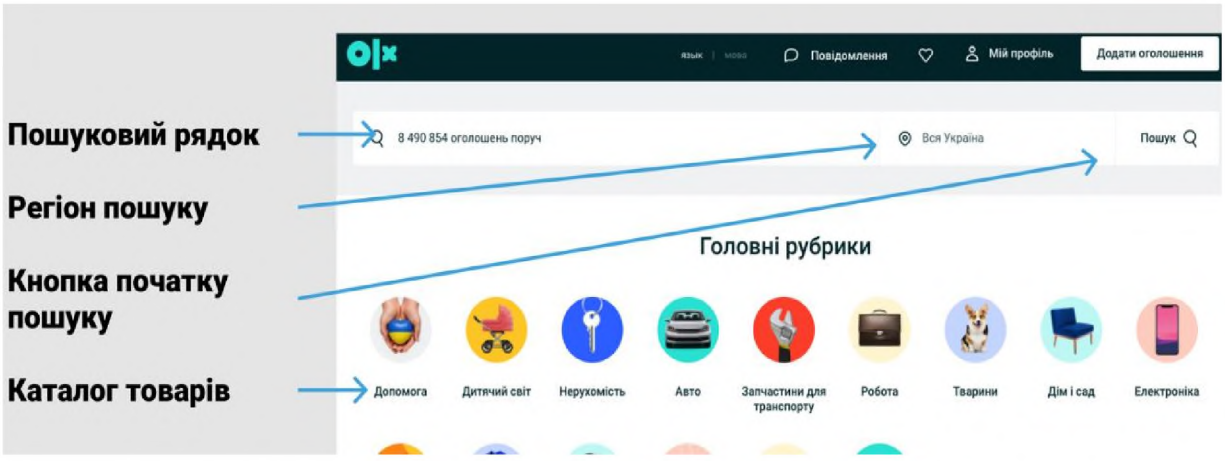


Рисунок 3.1 – Аналіз основної сторінки сайту olx.ua

У випадку з сайтом olx.ua видача оголошень відбувається в 3 етапи:

- пошук оголошень з головної сторінки;
- видача усіх оголошень за запитом;
- сторінка з вмістом оголошення.

Проаналізувавши дії, які виконує користувач, можна зробити перелік послідовних дій, які далі буде виконувати бот, імітуючи поведінку користувача. Переглянувши головну сторінку сайту (див. рис. 3.1), розділяємо її на послідовні кроки. Таким чином, можна розробити послідовність дій для бота, вони будуть наступні:

- зайти на сайт;
- створити необхідний запит до пошукового рядка;
- вибрати регіон пошуку оголошень;
- натиснути на кнопку «Пошук».

Після проведення цих маніпуляцій потрібно розділити дії на елементи. Це потрібно тому що PM не може сам зрозуміти послідовність дії.

Першою дією у проєкті вважається «Очищення файлів cookie», тобто бот анонімізує свої дії та удає ніби користувач зайшов на вебсайт вперше. Далі необхідно за допомогою інструменту «Перехід на сторінку» виконати емуляцію заходу на вебсайт.

У налаштуваннях дії потрібно вказати «Відкрити URL» та записати повне посилання головної вебсторінки, в нашому випадку це «https://www.olx.ua/uk/».

Знаючи точну послідовність кроків, зрозуміло, що після заходу на сайт наступним послідовним кроком для бота є робота з рядком пошуку та емуляція введення даних, тобто введення запиту.

Після входу на сайт звертаємо увагу на те, що в пошуковому рядку вже активований курсор для введення запиту, це полегшує наступні дії, адже розробнику не потрібно автоматизовувати натискання на пошуковий рядок для активації курсору. Наступна логічна дія в проєкті є емуляція введення запиту в пошуковий рядок, зробити це можна багатьма способами, але найдоцільніше у нашому випадку робити ці дії інструментом «Емуляція клавіатури».

У налаштуваннях інструменту «ЕК» необхідно вказати текст запиту, який надалі буде введено до пошукового рядка. В нашому випадку в налаштуванні емуляції вказуємо слово «комп'ютер».

Далі за необхідністю, на сайті можна вказати регіон для пошуку оголошень, в моєму випадку це місто «Полтава».

По аналогії з діями користувача, боту необхідно після роботи з пошуковим рядком перейти до взаємодії з полем «Вибору регіону пошуку». В середовищі РМ це можна зробити за допомогою інструменту «Емуляція миші», але в цій дії є певна проблема. В РМ емуляція натискання мишею на потрібний елемент сторінки реалізована шляхом прив'язування до координат певного елементу на сторінці, але сітка координат може мінятися від пропорції вікна браузера тобто сітка координат браузера, який відкритий на половину екрану, буде відрізнятися від сітки координат браузера, який відкритий в повноекранному режимі. Також на це ще впливає й роздільна здатність монітора. Повнота проблеми, як на рис. 3.2, показує, що вплине на перенесення боту на інший ПК чи систему.

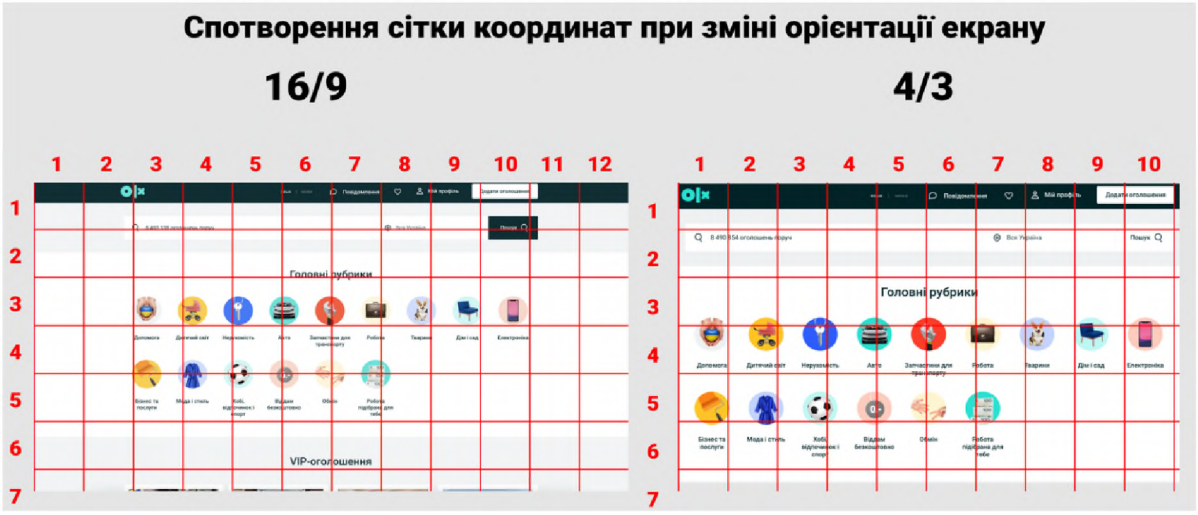


Рисунок 3.2 – Спотворення сітки координат при зміні орієнтації екрану

Обійти цю проблему можливо декількома способами. Одним із головних методів вважається пошук координат елементів у залежності від пропорцій вікна браузера. В РМ для цього є вбудований функціонал, але цей метод не є цілком надійним за рахунок того, що розмітка елементів це своєрідна квадратна сітка, і

якщо нам потрібно знайти кнопку, яка має скруглення, то отримаємо координати місця, в якому немає активної частини елемента «кнопка».

Для автоматизації дії у веббраузерах у застосунку PM є ще одна функція, яка по праву користується популярністю: мова йде про використання мови запитів до елементу XPath або XML Path Language. За допомогою XPath можливо реалізувати більш універсальний та стійкий до змін верстки сайту алгоритм пошуку даних у порівнянні з регулярними виразами [23]. Ця мова запитів дозволяє значно спростити логіку парсерів, а відтак, прискорити їхню розробку.

Визначившись із потрібною технологією, необхідно скопіювати код XPath рядка «Вибору регіону» на пряму із вбудованого функціонала «Пошук елемента» (рис. 3.3) та в PM використати функцію «Виконати подію» з функцією «Клік по XPath елементу». Таким чином бот натисне на поле вводу, після чого за допомогою інструменту «Емуляція клавіатури» користувач має змогу ввести необхідний текст у поле для введення.

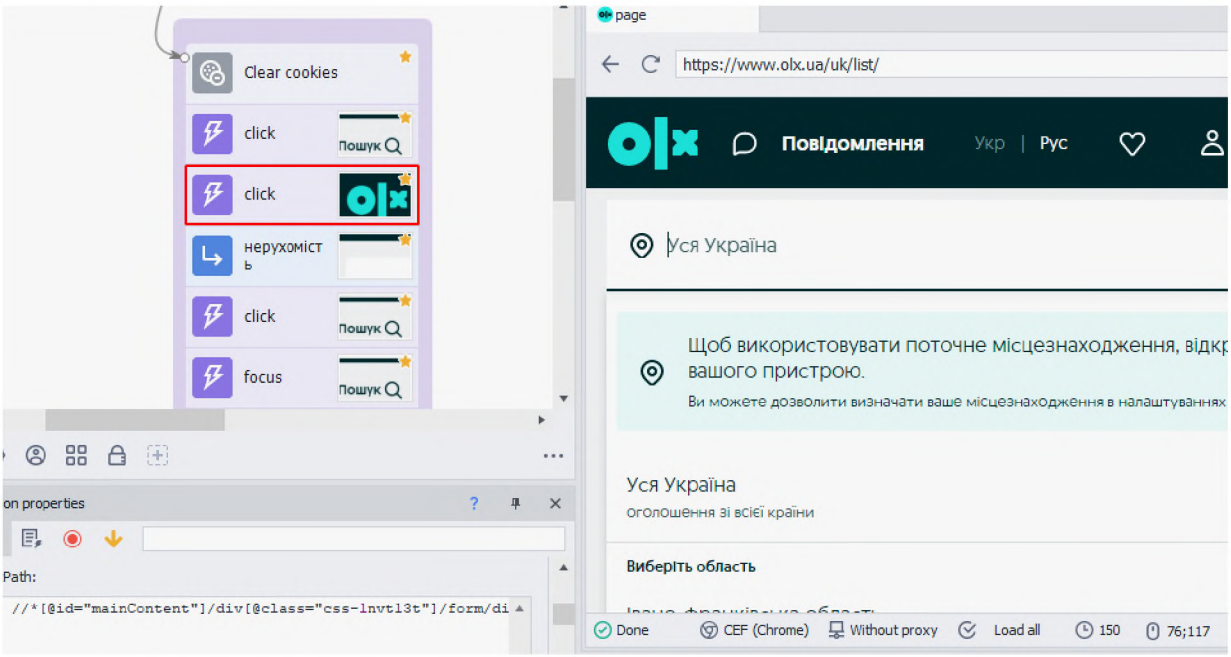


Рисунок 3.3 – Копіювання коду XPath за стандартним функціоналом браузера

Останній крок на цій сторінці, це натискання кнопки «Пошук», це реалізується тим же чином як і попередній крок з залученням функції «Клік по XPath елементу».

Після кліку по кнопці «Пошук», сайт створює сторінку з видачею оголошень. На цій сторінці показані всі оголошення, котрі підійшли під уведені критерії пошуку. Далі також необхідно провести аналіз елементів нової сторінки аби налаштувати необхідні дії для бота (рис. 3.4).

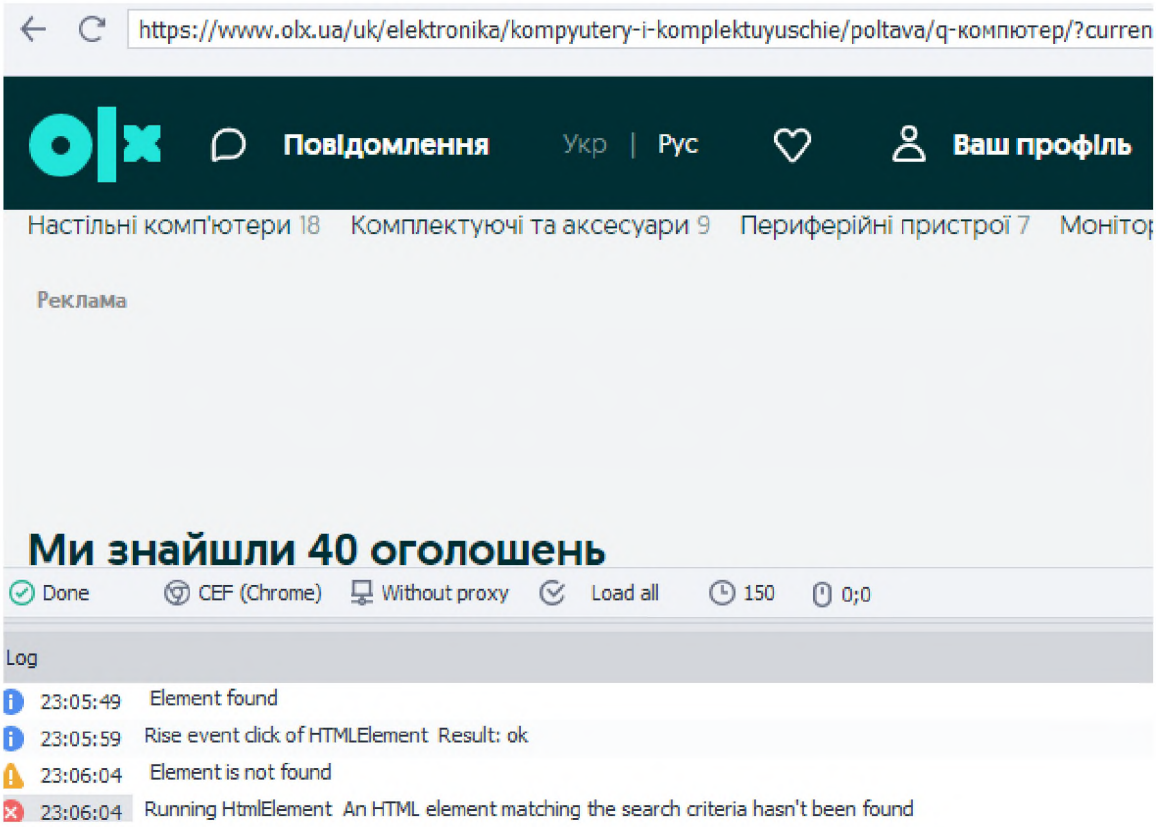


Рисунок 3.4 – Аналіз елементів наступної сторінки

Проаналізувавши сторінку з оголошеннями, бачимо кількість оголошень на сторінці та кількість сторінок з оголошеннями. Таким чином, розробляємо план дії для бота, який буде наступним:

– зібрати посилання на всі оголошення зі сторінки;

– перейти на наступну сторінку з оголошеннями та повторити збір посилань на оголошення;

– повторити ці дії до тих пір, поки бот не збере усі посилання з усіх сторінок.

Може з’явитись логічне запитання, навіщо збирати посилання на оголошення, якщо можна одразу заходити на оголошення та збирати інформацію. Пояснення наступне: якщо є перелік посилань на оголошення, то

бот витратить менше часу на збір інформації, адже із переліку дій виключиться дія, в якій бот кожного разу повертається на сторінку з оголошеннями.

Наступну дію зі збором посилань найлегше можна реалізувати шляхом застосування регулярних виразів. Якщо подивитись на HTML-код сторінки та застосувати інструмент «Пошук елементу», існує фрагмент коду, що відповідає за певне оголошення (рис. 3.5).

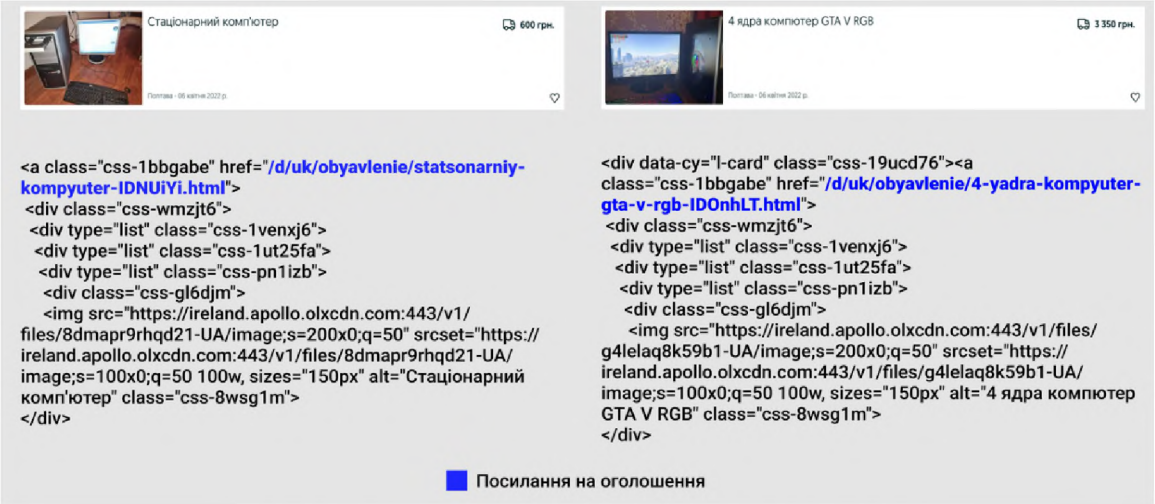


Рисунок 3.5 – Порівняння коду оголошень

Провівши аналіз, стає зрозуміло, що код всіх оголошень є однотипним, тобто можна легко знайти посилання на оголошення та автоматизувати цю дію. Після встановлення однотипності всіх оголошень, можна почати розробляти регулярний вираз для збирання посилання. Для наочності (див. рис. 3.5) частину коду, яку необхідно зібрати, виділено синім кольором.

Розробити та перевірити регулярний вираз в PM можна за допомогою інструменту «Конструктор регулярних виразів» (рис. 3.6). Саме під час створення регулярного виразу досліджено фрагмент коду оголошення та зроблено висновок, що в коді перед усіма посиланнями на оголошення стоїть код «/obyavlenie/», а в кінці кожного посилання обов'язково стоїть фрагмент «.HTML». Так буде визначено основні змінні для створення регулярного виразу.

Перевіривши регулярний вираз, було визначено, що посилання дублюються. Це відбувається через те, що сервер надсилає код сайту, у якому заздалегідь збережений код для мобільної версії сайту та комп'ютерної, а браузер

вже сам розуміє, яку версію коду необхідно використовувати, враховуючи розміри екрану та інші змінні (рис. 3.6). Таким чином, регулярний вираз знаходить посилання на оголошення із двох джерел: мобільної версії оголошення та комп'ютерної.

Вирішити проблему дублювання посилань досить легко: після збереження всіх посилань, які відповідають регулярному виразу, в список, за допомогою влаштованого функціоналу «Видалення дублікатів» видаляємо усі дубльовані посилання.

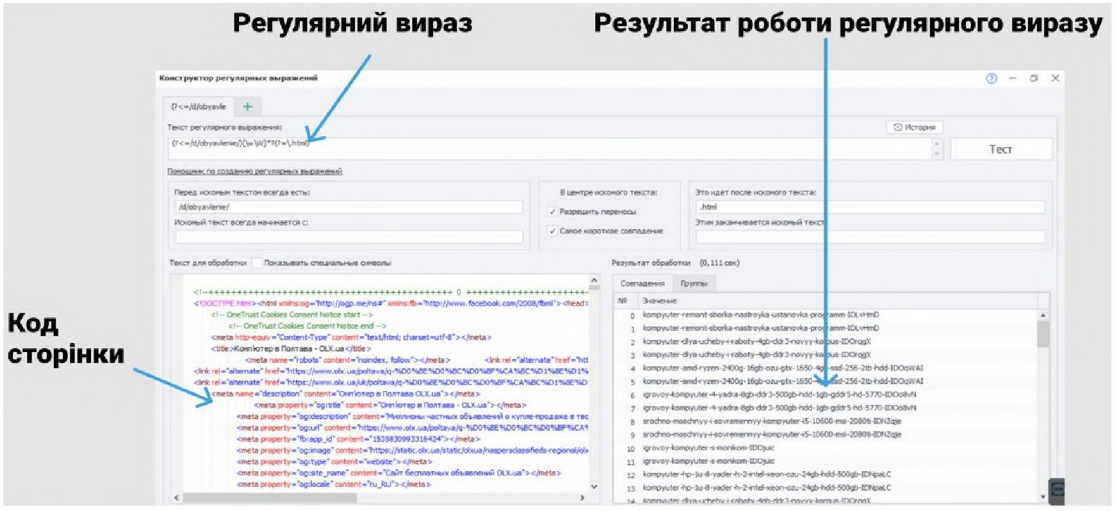


Рисунок 3.6 – Конструктор регулярних виразів

Після розробки регулярного виразу, наступним кроком є його використання. В РМ для парсингу інформації регулярними виразами є функціонал під назвою «Дані» (рис. 3.7).

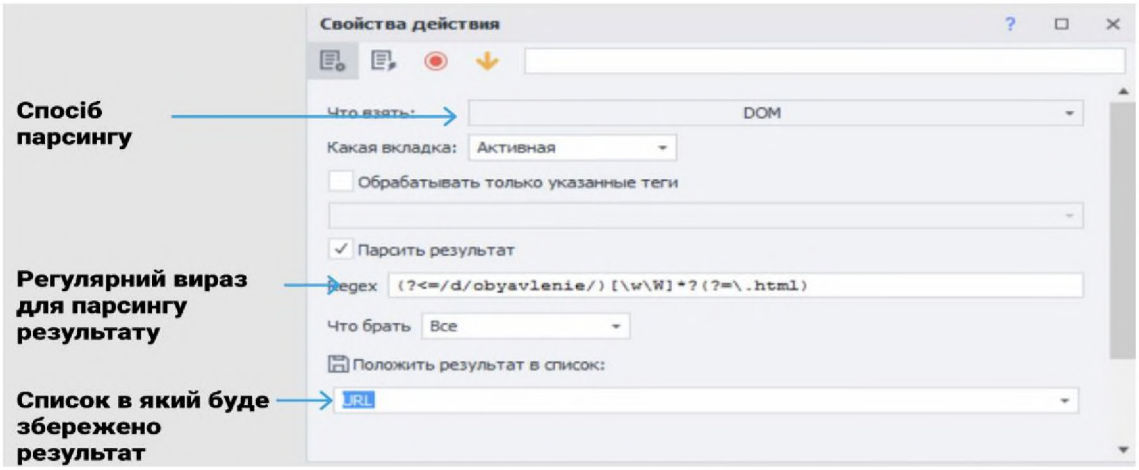


Рисунок 3.7 – Вікно функціоналу «Дані»

Цей функціонал (див. рис. 3.7) створено для того, щоб збирати дані за допомогою регулярного виразу. Функціонал, зокрема, призначений іще й для зберігання інформації у зручний спосіб, на разі плануємо зберігати посилання до списку, адже з ним надалі буде легко взаємодіяти.

Отже, створено список, до якого парсер автоматично зберігає посилання, та за допомогою функції «Видалення дублікатів» видаляє усі дублікати зі списку, щоб надалі не отримувати дублювання інформації та зменшити кількість дій для бота. Проведено тестовий запуск бота, щоб переконатися у його працеспроможності та правильності налаштувань: на виході отримуємо список, у якому були 40 унікальних посилань.

Далі необхідно перейти на наступну сторінку з оголошеннями, зробивши автоматизацію цих дій по аналогії з натисканнями на кнопку «Пошук».

Після збору посилань на всі оголошення, починаємо останній аналіз внутрішньої сторінки з оголошенням (рис. 3.8), саме на цій сторінці є вся необхідна інформація, яку збираємо та зберігаємо до таблиці.

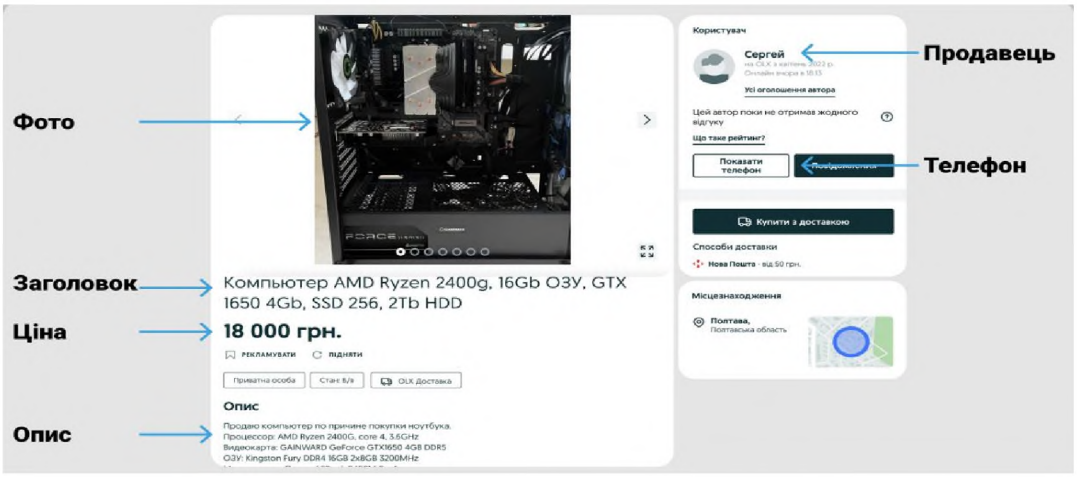


Рисунок 3.8 – Аналіз сторінки з оголошенням

Проаналізувавши сторінку з вмістом оголошення, створюємо в PM 7 змінних для того, щоб під час парсингу передавати результат парсингу у змінну. Змінні будуть наступні:

– photo (в цю змінну передамо посилання на фото з оголошення);
– h1, змінна з заголовком;

– price, змінна з ціною;
– desc, змінна з описом оголошення;
– name, змінна з ім’ям продавця;
– phone, змінна з контактними даними;
– link, змінна з посиланням на оголошення.

Розміщувати результат парсингу необхідно в змінну, щоб далі передати цю інформацію до таблиці. Змінна постійно буде займати один стовпчик, але щоразу новий рядок. Парсити інформацію будемо так само, як і посилання на оголошення: за допомогою регулярних виразів (код наведено нижче).

h1 - (?<=<h1\ data-cy="ad_title"\ class=".*">).*(?=</h1>)

price - (?<=<div\ data-testid="ad-price-container"\ class=".*"><h3\ class=".*">).*(?=</h3>)

desc - (?<=<h3\ class=".*">Опис</h3><div\ class=".*">)[\w\W]*?(?=<div\ class="css-cgp8kk"><div\ data-cy="ad-footer-bar-section"\ class="css-1ferwkx">)

phone - (?<=<li\ class=".*"><a\ href="tel:.*"\ data-testid="contact-phone"\ class="css-v1ndtc">\+)[\w\W]*?(?=</a></li></ul>)

name - (?<= <h2\ class=".*">)[\w\W]*?(?=</h2>)

photo - (?<=<div\ class="swiper-zoom-container"><img\ src=")[\w\W]*?(?="\ alt=".*"\ sizes=")

Після того, як бот готовий та пройшов калібрування усіх модулів, отримуємо таблицю Excel з результатами парсингу (рис. 3.9.)

| | B | D | F |
|---|---|---|---|
| | abc | abc | abc |
| eland.apollo.olxcdn.com:443/v1/files/mlauoepqv8d-UA/image;s=2000x1500 | Продам Skoda Octavia tour | Евгений | prodam-skoda-octavia-tour-IDOsN3Q |
| eland.apollo.olxcdn.com:443/v1/files/0vmqd0oycwms1-UA/image;s=934x741 | Toyota RAV4 2.2D cat 4-WD | Андрей | toyota-rav4-2-2d-cat-4-wd-IDOo80i |
| eland.apollo.olxcdn.com:443/v1/files/0vmqd0oycwms1-UA/image;s=934x741 | Toyota RAV4 2.2D cat 4-WD | Андрей | toyota-rav4-2-2d-cat-4-wd-IDOo80i |
| eland.apollo.olxcdn.com:443/v1/files/j8kyyqbboyvs3-UA/image;s=648x1152 | Материнская плата Kllisre X99 двухпроцессорная LGA 2011 v3 E-ATX | Андрей | materinskaya-plata-kllisre-x99-dvuhprotsess |
| eland.apollo.olxcdn.com:443/v1/files/j8kyyqbboyvs3-UA/image;s=648x1152 | Материнская плата Kllisre X99 двухпроцессорная LGA 2011 v3 E-ATX | Андрей | materinskaya-plata-kllisre-x99-dvuhprotsess |
| eland.apollo.olxcdn.com:443/v1/files/qlkevktg03q3-UA/image;s=750x1000 | продам торговое оборудовварие для пива на розлив. | Артем | prodam-torgovoe-oborudovvarie-dlya-piva-n |
| eland.apollo.olxcdn.com:443/v1/files/qlkevktg03q3-UA/image;s=750x1000 | продам торговое оборудовварие для пива на розлив. | Артем | prodam-torgovoe-oborudovvarie-dlya-piva-n |
| eland.apollo.olxcdn.com:443/v1/files/ktc9asgu7c6m3-UA/image;s=2322x4128 | Мощный игровой компьютер на Ryzen 5 3600 и ASUS ROG STRIX-GTX1070 | Вячеслав | moschnyy-igrovoy-kompyuter-na-ryzen-5-36 |
| eland.apollo.olxcdn.com:443/v1/files/2m03h8m49hkm3-UA/image;s=750x100 | Компьютер PC персональный компьютер | Александр | kompyuter-pc-personalnyy-kompyuter-IDOn |
| eland.apollo.olxcdn.com:443/v1/files/2m03h8m49hkm3-UA/image;s=750x100 | Компьютер PC персональный компьютер | Александр | kompyuter-pc-personalnyy-kompyuter-IDOn |

Рисунок 3.9 – Таблиця з результатами парсингу

Далі необхідно задати налаштування для збереження результатів парсингу до Microsoft Excel або Google sheets за допомогою функціоналу PM. За необхідності можна провести видалення дублікатів даних за допомогою штатного функціонала.

Кінцевим етапом розробки бота в PM є збереження шаблону бота у формат .zp для того аби надалі запускати бота в програмному засобі ZennoPoster.

## 3.2 Інтеграція бота з Telegram Bot API та підтримка працездатності

Інтеграція бота з Telegram API це необхідний та логічний крок для полегшення взаємодії з ботом та можливість отримати можливість бути більш мобільним при роботі з ботом. Наприклад, за допомогою бота можна надсилати до свого Telegram технічні повідомлення різного характеру, або взаємодіяти з ботом за допомогою post/get запитів Telegram API.

Telegram Bot API – це сторонні програми, які працюють всередині Telegram. Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити. Користувачі керують своїми ботами за допомогою HTTPS-запитів до API ботів Telegram [24].

Бот може діяти як розумна газета, надсилаючи вам відповідний вміст, щойно його публікують. По суті, Telegram Bots – це спеціальні облікові записи, для налаштування яких не потрібен додатковий номер телефону. Користувачі можуть взаємодіяти з ботами двома способами:

– надсилаючи повідомлення та команди ботам, відкриваючи з ними чат або додаючи їх до груп;

– надсилаючи запити безпосередньо з поля введення, ввівши @username бота та запит. Це дозволяє надсилати вміст від вбудованих ботів безпосередньо в будь-який чат, групу чи канал.

Повідомлення, команди та запити, надіслані користувачами, передаються програмному забезпеченню, запущеному на серверах користувача. А сервер-

посередник обробляє усе шифрування та зв'язок із Telegram API [25]. Клієнт спілкуєтеся з цим сервером через простий HTTPS-інтерфейс, який пропонує спрощену версію API Telegram. Детальна схема такої архітектури представлена на рис. 3.10.

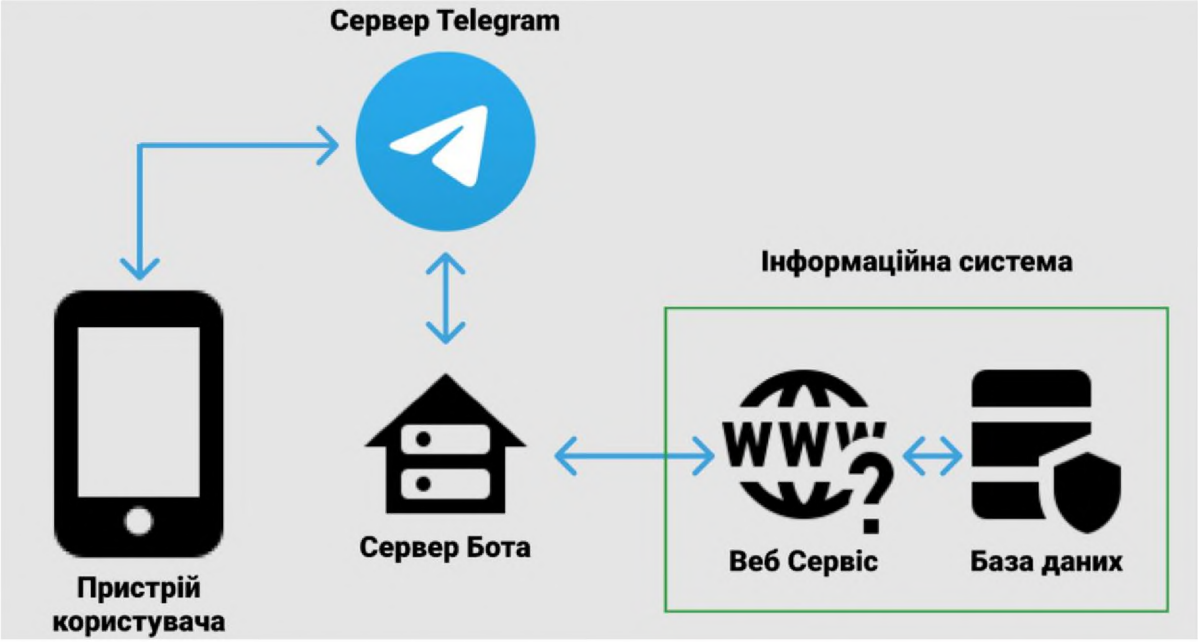


Рисунок 3.10 – Архітектура системи Telegram Bot API

Першочергово для взаємодії з Telegram Bot API потрібно в самому месенджері знайти контакт «https://t.me/BotFather» та в боті Telegram натиснути кнопку «Start», після чого перед вами з'явиться панель для створення та управління ботами з наступними функціями:

– створення нового бота;

– налаштування вже наявного бота.

Оскільки раніше ми вже створювали бота, натискаємо «Налаштування вже наявного бота» та перехожу до наступного меню яке дає можливість управляти ботом та таким функціями [26]:

– отримати API Token, для підключення до боту та взаємодії з ним через сервер;

– Bot Settings, меню для налаштування основних команд для бота, на кшталт команди «/start»;

– Transfer bot ownership, меню для передачі бота та додавання адміністраторів;

– Edit Bot, меню для налаштування назви, фото та підпису бота;

– Delete Bot, меню для видалення бота.

На початку взаємодії з ботом необхідно отримати API Token, який дає змогу підключитися з сервера до Telegrem Bot API та отримувати чи передавати HTTPs запити. Виглядає API Token наступним чином:

– 5046723928:AAE0B0WjxFpbV4Ew_uQLuAkbHInlbMlZPog, в токені закладено декілька змінних, а саме ключ для ідентифікації та пароль.

Для взаємодії з Telegram в PM є функціонал, який вправно надсилає та отримує HTTPs запити. Головне перед використанням запитів ознайомитись із документацією, адже звичайному користувачу без навичок в програмуванні Telegram API покажеться надто складною.

Основні параметри Telegram Bot API якими я буду користуватись це [26]:

– /getupdates, отримати оновлення, ця команда показує все що відбулося з ботом за 24 години, включаючи всі повідомлення;

– /sendMessage, для відправлення технічних повідомлень із PM в Telegram.

– text, повідомлення;

– /chat_id, унікальний ідентифікатор цільового чату або імені користувача цільової супергрупи.

Насправді параметрів для взаємодії з ботами Telegram безліч. Усі вони перераховані в документації, яку можна знайти за посиланням https://core.telegram.org/bots/api. У цьому дослідженні основною роллю Telegram Bot є відправлення сповіщень про результат парсингу.

Для цього створено бот та отримано його токен. Наступним кроком необхідно знайти, підписатись на свого бота та натиснути кнопку «Старт».

Після описаних маніпуляцій необхідно створити HTTPs запит до бота, щоб отримати свій ID, для спілкування з користувачем. Запит виглядає наступним чином: https://api.telegram.org/bot[Токен]/getUpdates.

Запит можна зробити просто у пошуковому рядку вашого браузера і якщо все зроблено вірно ви отримаєте повідомлення як зображено на рис. 3.11.

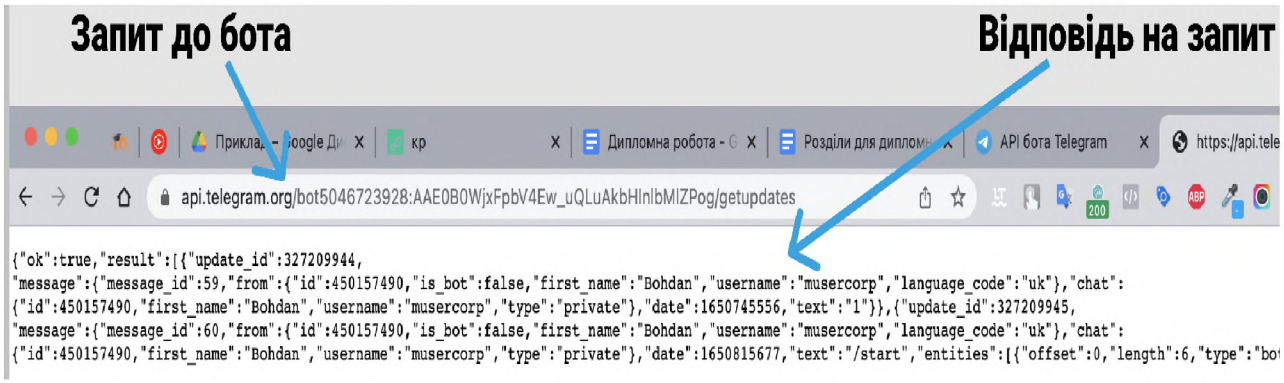


Рисунок 3.11 – Запит до бота та його відповідь

Створивши запит до бота з параметром /getUpdates, ми запитуємо у бота «що в тебе новенького?», на що бот відповідає повідомленням, в якому сказано, що «Користувач з ім'ям Богдан натиснув на кнопку /start», на початку повідомлення бота є параметр «ID». Цей параметр відповідає за аутентифікацію користувача, тобто для того, щоб написати користувачу, від імені бота, мені необхідно створити запит з параметром «/sendMessage», вказати ID користувача та текст повідомлення, текст запиту буде виглядати наступним чином:

https://api.telegram.org/bot[Токен]/sendMessage?chat_id=[ID]&text=Привіт

Розібравшись із налаштуваннями Telegram Bot API, переходимо до PM та розпочинаємо інтеграцію.

Інтеграція з ботом буде полягати в тому, щоб після кожного парсингу відправляти в Telegram повідомлення з аналізом наявності вмісту у кожної з 7 змінних які я створював раніше, бот проаналізує чи є в змінній дані та надішле повідомлення це необхідно для того аби розуміти правильність налаштувань регулярних виразів та працездатності парсеру. Адже кожен вебсайт рано чи пізно оновлюється що може вплинути на результат роботи регулярних виразів, адже вони є реєстрозалежні [7].

Таким чином можна відправляти майже будь-що. Робота інтеграції на прикладі одного запиту в робочому середовищі наведена на рис. 3.12.

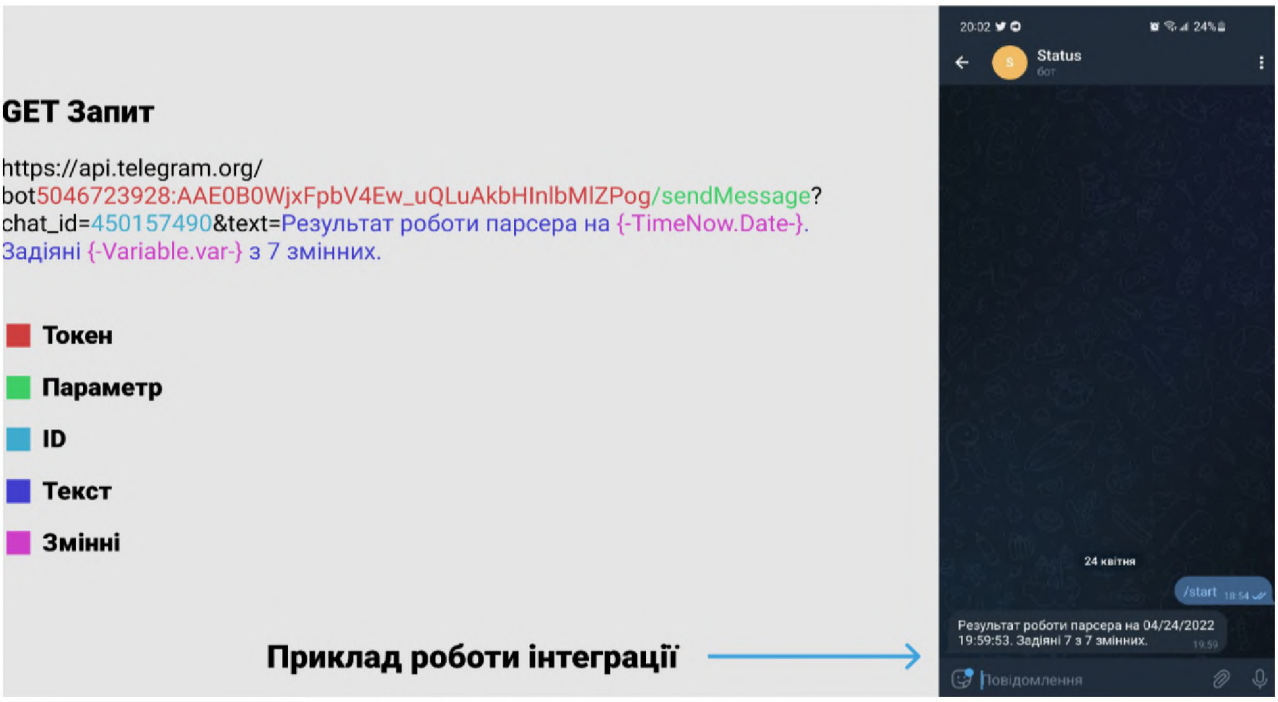


Рисунок 3.12 Приклад роботи інтеграції

Ще одним із видів застосування інтеграції парсера з Telegram є пряма інтеграція з ботом та управління ним. Тобто за допомогою прямої інтеграції я зможу надіслати в Telegram повідомлення із текстом для пошуку, а ZP своєю чергою прийме повідомлення та запустить парсинг з потрібним текстом пошукового рядка, після чого надішле Excel файл з даними парсингу до Telegram. Можливості, які можна застосувати за допомогою такої інтеграції, дійсно величезні.

При роботі з ботами важливо не лише обрати вдалу технологію створення, але й враховувати фактор підтримки його працездатності. У зв'язку з тим, що вебсайти для яких розробляють ботів, продовжують оновлюватись, виникає необхідність підтримувати працездатність бота, адже навіть незначна зміна у коді вебсайту може внести корективи в роботу регулярного виразу чи поведінки бота.

Задля виявлення проблем з виконанням дій бота в PM є функція «Bad End» яка виконає спеціальний алгоритм дій в разі якщо під час роботи бота в якійсь із дій відбудеться помилка (рис. 3.13). Для використання функції застосовується окремий сценарій.

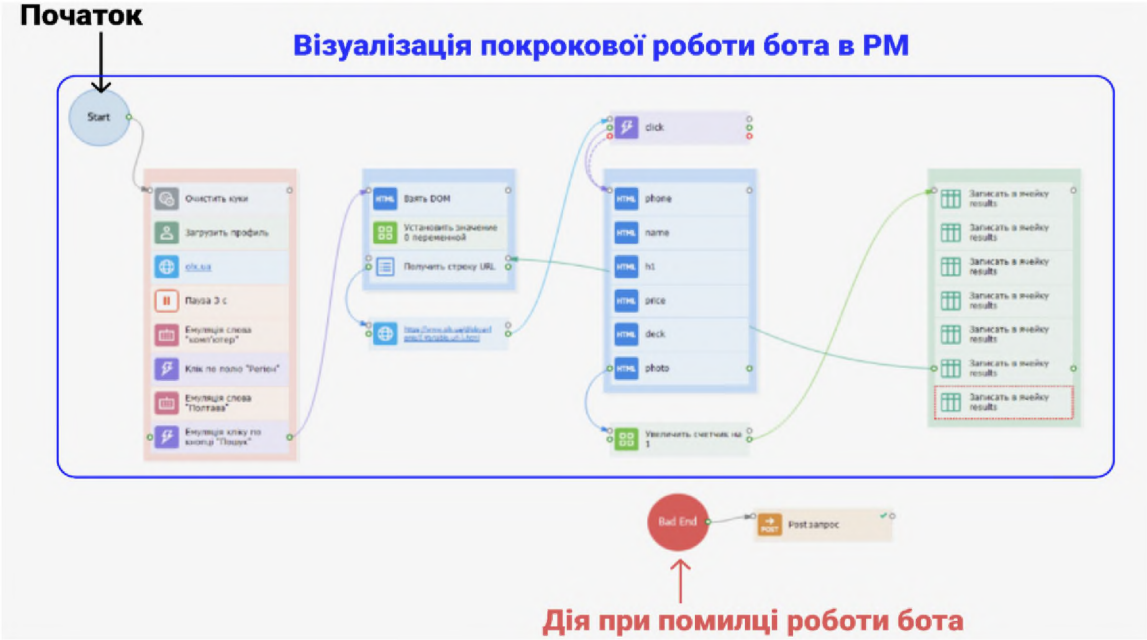


Рисунок 3.13 – Візуалізація покрокової роботи бота в PM

Для попередження про помилку в роботі боту була застосована функція «Bad End» і розроблений сценарій, в якому під час активації функції до мого Telegram Bot буде надіслано повідомлення про помилку та Log системи.

Log – це спеціальний файл у якому зберігається службова інформація про події в системі та інформація про помилки в виконанні дій [27] зоб. на рис. 3.14.

Після перегляду Log-файлу можна чітко зрозуміти, на якому етапі відбулася помилка, що її викликало та наслідки помилки. Завдяки цьому можна оперативно розв'язувати проблему та запобігти її появи у майбутньому.

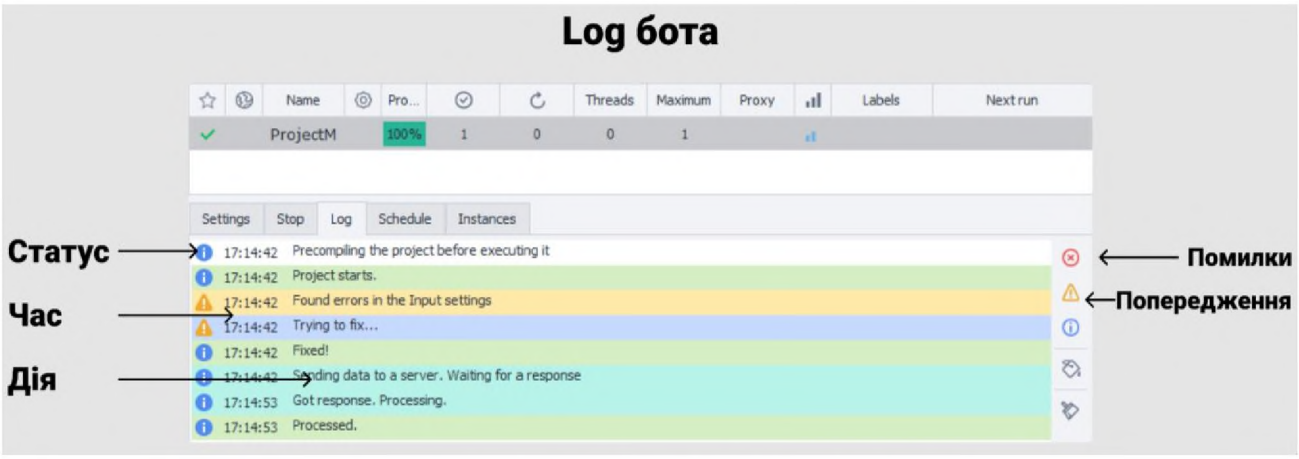


Рисунок 3.14 – Log системи в PM

На вкладці Log записуються повідомлення про хід виконання проєкту: помилки, відмітки про успішне/невдале виконання проєкту.

За допомогою кнопок у правій частині вкладки можна очистити журнал, увімкнути автоматичне прокручування, фільтрувати повідомлення за типом та/або кольором.

Завдяки тому, що PM має чітку покрокову візуалізацію дій, користувач має змогу швидко вносити зміни у налаштування бота.

## 3.3 Техніко-економічне обґрунтування ефективності застосування парсингу

Область застосування парсингу не має кордонів: парсинг застосовується майже у всіх галузях інформаційної діяльності людини, від аграрних підприємств до атомної промисловості. Окремі, найбільш популярні області застосування парсингу, описані далі.

1. Товари з інтернет магазинів. Парсинг часто використовується для роботи із товарами інтернет-магазину. Товари збираються в Excel-базу та успішно імпортуються в інтернет-магазин. Під час імпорту буде завантажено зображення, видалено старі товари та оновлено існуючі. Так, каталог сайту завжди буде актуальним.

Навіть у невеликих інтернет-магазинах парсинг значно полегшує ручну роботу менеджера. Це особливо відчувається, коли товарів кілька тисяч.

Нерідко парсинг застосовується для завантаження товарів із зарубіжних ресурсів. Під час парсингу додатково підключається автоматизований перекладач, який формує практично унікальні тексти;

2. Порівняння цін конкурентів. Успішність бізнесу залежить від багатьох факторів, у тому числі від цінової політики компанії. Клієнти можуть надавати пріоритет тим компаніям, які надають товари або послуги дешевше, ніж у вас.

Тому потрібно постійно моніторити конкурентів та знати, за якими цінами надаються послуги на ринку.

Наразі послуги з моніторингу цін конкурентів надаються повсюдно. Принцип таких сервісів є досить простим. Дані про ваші товари зберігаються у Excel-файлі. Парсинг додає в цей же файл інформації про ціни з інших інтернет-магазинів. У результаті у вас є порівняльна таблиця, орієнтуючись на яку ви можете своєчасно приймати рішення.

Виконувати таку роботу вручну на сьогодні, як мінімум, нерозумно. Ця діяльність займає дуже багато часу, неефективна і нерентабельна;

3. Демпінг цін. Демпінг – штучне зниження цін на товари та послуги з метою проникнення, зміцнення та витіснення конкурентів на ринку [29].

Такі товари значно дешевші за товари конкурентів, тому користуються великим попитом. Демпінгові товари продаються масово, тим самим збільшуючи рейтинг магазину, кількість продажів, відгуків, популярність тощо. Після цього магазин відновлює ціни на товари, а зароблений рейтинг працює на магазин.

Витрати на парсинг та збитки з продажу товарів за демпінговими цінами окупаються за рахунок того, що в майбутньому популярний магазин зможе мати більшу кількість клієнтів, продажу та прибутку відповідно;

4. Парсингу особливо потребують сайти з «гарячим» контентом і таким, що часто оновлюється. Як правило, це міські та новинні портали, кіносайти, сайти-агрегатори компаній, майданчики з результатами спортивних змагань, сайти для бронювання готелів та купівлі квитків тощо.

Інформація для таких сайтів збирається із різних джерел. Курси валют та прогноз погоди можна без проблем отримати через публічний API. Однак, API не передбачений для більшості неоднорідних даних, тому парсинг є найбільш вдалим способом їх отримання.

На сайтах-агрегаторах сконцентровано, упорядковано та зручно представлено дійсно багато інформації, тому такі сайти мають велику

відвідуваність. За допомогою парсингу сторінки таких сайтів створюються в автоматичному режимі;

5. Початкові дані для легкого старту. Існують такі категорії сайтів, запуск яких неможливий без великих обсягів контенту ще на початкових етапах. Без цього сайт абсолютно непотрібний для користувачів. До таких сайтів відносяться дошки оголошень, каталоги компаній, форуми, портали новин, інформаційні сайти і т.д. Але де взяти одразу багато контенту, коли сайт ще не має аудиторії? Рішенням буде використання парсингу, який автоматично збере потрібні дані з різних джерел. Щоб заповнити такий майданчик оголошень і зробити видимість активності на ній, досить просто зробити парсинг, імпортувати ці дані на сайт, запустити сайт в роботу, а потім поступово витісняти завантажені оголошення своїми;

6. SEO-аналіз ефективності вебзастосувань: парсинг є хорошим помічником для SEO-фахівців, вебстудій та онлайн-сервісів з просування сайтів.

Для просування в інтернеті можна і потрібно скористатися рекомендаціями для вебмайстрів від Google. Однак, лише цих рекомендацій замало для ефективного просування, оскільки поради описані надто узагальнено. Щоб насправді дізнатися, як працює пошукова видача і які показники найважливіші для ранжування, використовується парсинг.

Наприклад, можна запустити парсер, зібрати статистичну інформацію сотень тисяч сайтів та проаналізувати її. Приклад таких даних: довжина текстів, заголовків, описів, водність тексту та частота ключових слів, наявність зображень, маса посилань і т.д;

7. Соціальні мережі. Сьогодні у соціальних мережах багато підприємців будують цілий бізнес. Адже в них зосереджено величезну аудиторію різних інтересів, поглядів, потреб. Навіть простий пошук усередині майданчика дозволяє досить легко отримати необхідних вам потенційних клієнтів.

Але для того, щоб вичавити з соціальних мереж максимум, потрібно вийти за межі інтерфейсу соціальної мережі. Парсинг дозволяє отримати набагато більш необхідну та схильну до покупки аудиторію.

Зрештою, розглянемо окремі економічні аспекти щодо вартості та окупності розроблення такого програмного прдукту, як парсер. Загальна вартість підтримки бота для парсингу визначається з урахуванням розробки повноцінного бота для парсингу на базі ПЗ ZennoPoster на один місяць та використання обладнання і ПЗ протягом року за формулою:

$$V_p = V_s + V_{os} + V_{ZP} + V_{MS} + V_{pc}, \quad (3.1)$$

де $V_s$ – вартість оренди сервера;

$V_{os}$ – вартість операційної системи;

$V_{ZP}$ – вартість пакету програмного комплексу ZennoPoster;

$V_{MS}$ – вартість ліцензії на пакет MS Office;

$V_{pc}$ – вартість роботи з написання програмного коду.

Актуальні значення показників для розрахунків за наведено в табл. 3.1.

Таблиця 3.1 – Вартість програмного забезпечення розробки парсера

| Найменування елементу програмного забезпечення | Вартість використання, грн/міс. | Вартість використання на рік, грн |
|---|---|---|
| Оренда сервера на міс у постачальника DigitalOcean [30] | 150 | 900 |
| Операційна система Windows 10 Professional x64 | 400 | 4800 |
| Ліцензія на програмний комплекс ZennoPoster | 1087 | 13044 |
| Ліцензія на програмне забезпечення MS Office Excel | 242 | 1452 |
| Разом, грн | 1879 | 20196 |

Окрім вартості програмного забезпечення (див. табл. 3.1), необхідно врахувати заробітну плату програміста. В середньому програміст витрачає часу на розробку бота для парсингу сайтів приблизно 24 години (протягом 3 робочих днів). Середня заробітна плата програміста рівня Junior за даними сайту [31] складає ≈ 590 грн. за годину. Тобто оплата роботи програміста буде складати одноразово ≈ 14 160 грн. Утримання бота на перший рік відповідно 34356.

Але це одноразові витрати, оскільки надалі необхідно витрачати кошти виключно на щомісячну підтримку бота, що на рік складає ≈ 20 196 грн.

Для повноцінного розуміння вигоди у використанні ботів запропоновано розглянути завдання по збору оголошень з сайту olx.ua в порівнянні бота з людською працею. Людина за годину людина збирає 24 оголошення, в той час як бот збирає 61 оголошення за годину, тобто бот на 60 % ефективніший за людину.

Згідно з законом, мінімальний розмір заробітної плати станом на 01.04.2024 р. складає 8000 грн за 8 годин праці в день та 22 робочих днів на місяць. Якщо брати цю цифру до розрахунків, то виходить що за рік ручного збору буде зібрано приблизно 50 688 оголошень, і на це буде витрачено 96000 грн з урахуванням податків та відрахувань у вигляді заробітної плати.

У той час, як бот за рік збере 527 040 оголошень, якщо буде працювати 24 години 7 днів на тиждень, то на це буде витрачено 34356 грн. Результати розрахунків відображені в табл. 3.2.

Таблиця 3.2 –Порівняльні витрати на збір оголошень людиною і ботом

| Варіант збору оголошень | Продуктивність, оголошень шт./день | Витрати, грн/день | Продуктивність, оголошень шт./ місяць | Витрати, грн / місяць | Оголошень, шт./рік | Витрати, грн/рік | Витрати на 1 оголошення, грн |
|---|---|---|---|---|---|---|---|
| Людина, 8 год./день | 192 | 363,64 | 4224 | 8000 | 50 688 | 96000 | 1,89 |
| Бот, 24 год./день | 1464 | 94,13 | 44 530 | 2 823,78 | 534 360 | 34356 | 0,06 |

Отже, за розрахунками (див. табл. 3.1 - 3.2), можна зробити висновок про високу ефективність роботи бота у порівнянні з ручною працею, а також швидку окупність. Додатковим аргументом є наступний: одна з причин збору оголошень є отримання бази контактів для рекламних чи спам-розсилок. На інформаційному ринку можна купити базу з 10000 контактів за 100 доларів США (1$ = 38,7 грн за курсом НБУ станом на 10.04.24). Тобто, за рік людської праці (збору вручну) вийде заробити 19350 грн, в той час, як за допомогою бота заробіток складе 206658 грн. при продажу річної бази контактів одному замовнику.

# ВИСНОВКИ

В ході виконання кваліфікаційної роботи було ґрунтовно досліджено процес, завдання і технології парсингу як сучасного явища в інформаційних технологіях, результату використання всесвітньої мережі інтернет. Було здійснено розробку та випробовування власного варіанту парсингу. По завершенні виконання завдань кваліфікаційної роботи можна сформулювати наступні висновки.

1. Парсинг – це процес автоматичного збору даних та їх структурування. Спеціальні програми або сервіси-парсери «обходять» сайт та збирають дані, які відповідають заданій умові.

2. Головним поштовхом у розробці програмного забезпечення та технологій для парсингу стало те, що компанії розробники побачили як необхідність в швидкому автоматичному збору даних та їх структурування так і вигоду в розробці таких продуктів.

Серед основних технологій для парсингу є:

– розробка власних парсерів традиційним програмуванням з застосуванням інструменту, наприклад, Selenium та мови програмування Python;

– використання готових рішень у вигляді програмного забезпечення для парсингу;

– використання готових рішень у вигляді вебдодатків та розширень для веббраузера.

Одним із лідерів серед програмного забезпечення для парсингу є продукт від виробника ZennoLab, програмний комплекс ZennoPoster, що вміщує в себе два програмних забезпечення, таких як ProjectMaker, ZennoPoster.

3. Головним інструментом для парсингу є використання регулярних виразів (RegEx), це працює як спеціальна мова для опису шаблонів рядків. Використання цього інструменту може відрізнятися в різних мовах програмування. Завдяки регулярним виразам можна знайти данні, які підходять до регулярного виразу.

4. Зазвичай зібрану за допомогою парсерів інформацію зберігають до табличних процесорів MS Office Excel чи Google Sheets, щоб надалі взаємодіяти з інформацією, використовуючи влаштовані в табличний процесор інструменти. Цю дію також можна автоматизувати завдяки інструментам ПЗ ProjectMaker.

5. Розробка ботів для парсингу - це важливий та необхідний крок для людства, адже шукати, обробляти, аналізувати та розповсюджувати інформацію набагато легше, швидше та ефективніше, автоматизувавши цей процес.

6. Серед основних проблем парсингу є те, що цей процес потребує надлишковості розрахункових можливостей від сервера чи ПК користувача. Через це розробники ботів для парсингу все частіше звертаються до сервісів з оренди віртуальних серверів VPS, що значно економить ресурси, створює планомірну надлишковість потужності та забезпечує надійне зберігання даних.

7. Використання парсингу та ботів є більш ефективним рішенням в порівнянні з аналогічними діями людини. У багатьох задачах парсинг як рішення ефективніший від людини від 60 до 90 відсотків.

Результати, отримані в ході виконання кваліфікаційної роботи можуть бути корисними для бізнес-компаній, підприємств, де доводиться збирати та аналізувати величезні обсяги інформації, яка надходить через вебканали.

Досвід розробки і впровадження можливо розглянути під час виробничих або навчальних практик здобувчів вищої освіти, або лабораторних робіт.