МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему «РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ТА АНАЛІЗУ ДАНИХ З ВЕБ-САЙТІВ»

Виконав: студент 4 курсу, групи КНТ-521
спеціальності 123 Комп'ютерна інженерія

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(код і назва спеціальності)

КОБА Б.С.
(прізвище та ініціали)
Керівник ІЛЬЯШЕНКО М.Б.
(прізвище та ініціали)
Рецензент СТЕПАНЕНКО О.О.
(прізвище та ініціали)

2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) бакалаврський

Спеціальність 123 Комп'ютерна інженерія

(код і назва)

Освітня програма (спеціалізація) Комп'ютерна інженерія

(назва)

ЗАТВЕРДЖУЮ
Зав. кафедри
_____ Кудерметов Р.К.
"____" _____ 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

КОБА Богдан Сергійович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка системи моніторингу та аналізу даних з веб-сайтів»

керівник проекту (роботи) ІЛЬЯШЕНКО М.Б. к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "08" квітня 2025 року № 151

2. Строк подання студентом проекту (роботи) 01.06. 2025року

3. Вихідні дані до проекту (роботи) інструменти Selenium та BeautifulSoup4, архітектура BaseScraper, браузер Google Chrome, JSON, платформа Django Admin

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Опис предметної області

2) Збір даних з вебсайтів

3) Робота з даними

4) Проєктування системи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ІЛЬЯШЕНКО М.Б. к.т.н.. доцент		
Нормоконтроль	ПОЛЬСЬКА О.В., ст. викладач		

7. Дата видачі завдання 01.04. 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Опис предметної області	05.04.2025	
2	Збір даних з вебсайтів	10.05.2025	
3	Вибір програмного забезпечення	17.04.2025	
4	Збір та аналіз даних	23.04.2025	
5	Проєктування системи	18.05.2025	
6	Використання даних для прийняття рішень	25.05.2025	
7	Оформлення отриманих результатів у ПЗ	26.05.2025	
8	Оформлення графічного матеріалу	29.05.2025	
9	Захист роботи		

Студент _____ **Богдан КОБА**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **Матвій ІЛЬЯШЕНКО**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 60 с., 21 рис., 5 табл., 16 джерел.

АВТОМАТИЗОВАНА СИСТЕМА, АНАЛІЗ ДАНИХ, ВЕБ-САЙТИ ЗБІР ДАНИХ, ПРИЙНЯТТЯ РІШЕНЬ

Об'єктом дослідження виступає процес розробки, впровадження та тестування автоматизованої системи.

Предмет дослідження охоплює технічні аспекти реалізації системи та методи її аналізу.

Мета роботи полягає у всебічному дослідженні проблематики збору та обробки веб-даних для потреб бізнесу, а також у розробці й впровадженні автоматизованої системи, що вирішує ці завдання. Зокрема, мета включає такі складові:

Аналіз проблеми: дослідити сучасний стан бізнес-середовища, визначити актуальні проблеми, які можливо вирішити шляхом автоматизації збору й аналізу даних з веб-ресурсів.

Розробка системи: створити концептуальну модель та архітектуру автоматизованої системи для збору й аналізу веб-даних.

Впровадження і тестування: реалізувати систему в умовах реального бізнесу та перевірити її працездатність і ефективність.

Збір і аналіз даних: здійснити збирання інформації з веб-сайтів, провести її обробку з метою отримання інсайтів, що сприятимуть прийняттю управлінських рішень.

ABSTRACT

Explanatory note to the master’s work: 60 p., 21 figures, 5 tables, 16 sources.

AUTOMATED SYSTEM, DATA COLLECTION, WEBSITE DATA ANALYSIS, DECISION MAKING, BUSINESS INTELLIGENCE, MARKET RESEARCH

The object of research is the process of developing, implementing and testing an automated system.

The subject of the study covers the technical aspects of the system implementation and methods of its analysis.

The purpose of the study is to comprehensively investigate the issues of collecting and processing web data for business needs, as well as to develop and implement an automated system that solves these problems. In particular, the goal includes the following components:

Analysis of the problem: to study the current state of the business environment, to identify current problems that can be solved by automating the collection and analysis of data from web resources.

System development: create a conceptual model and architecture of an automated system for collecting and analyzing web data.

Implementation and testing: to implement the system in a real business environment and test its performance and efficiency.

Data collection and analysis: to collect information from websites, process it to obtain insights that will facilitate management decision-making.

ЗМІСТ

ВСТУП

У сучасному інформаційному суспільстві дані стали одним із ключових ресурсів, що впливають на прийняття рішень у бізнесі, науці, державному управлінні та багатьох інших сферах. Постійне зростання обсягів інформації, що публікується у відкритому доступі на веб-ресурсах, створює необхідність у розробці ефективних інструментів для її автоматизованого збору, обробки та аналізу. Особливої актуальності набувають системи, які здатні в реальному часі відстежувати зміни, оновлення та тенденції, що відображаються на веб-сайтах.

Тема має на меті створення технічного рішення, яке дозволить автоматизовано збирати, зберігати та аналізувати інформацію з інтернет-ресурсів з подальшою візуалізацією результатів. Така система є актуальною в умовах висококонкурентного середовища, де своєчасне отримання та правильне трактування даних надає значну перевагу.

Розробка подібної системи вимагає врахування низки технічних, юридичних та етичних аспектів, пов’язаних із веб-скрапінгом, структурою цільових веб-сайтів, обсягами оброблюваних даних, а також потреб користувачів у зручному інтерфейсі для взаємодії з результатами аналізу. В рамках дослідження розглядаються підходи до побудови гнучкого та масштабованого рішення, заснованого на сучасних технологіях, зокрема фреймворку Django, інструментах обробки даних і бібліотеках для експорту у зручні формати.

Отже, робота спрямована на реалізацію комплексного підходу до створення системи, здатної забезпечити ефективний моніторинг та аналітику веб-даних, що може бути застосовано у широкому спектрі задач — від моніторингу цін в електронній комерції до відстеження змін у новинних стрічках чи державних реєстрах.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вступ до теми

У сучасних умовах стрімкого розвитку інформаційних технологій та Інтернету збір і аналіз даних з веб-сайтів набуває все більшої значущості для прийняття стратегічних бізнес-рішень. Веб-скрапінг - процес автоматизованого витягування інформації з веб-ресурсів - перетворюється на ключовий інструмент забезпечення конкурентоспроможності підприємств у динамічному ринковому середовищі. Цей підхід дає змогу отримувати значні обсяги інформації з різноманітних джерел, що відкриває можливості для глибокого аналізу, виявлення закономірностей і тенденцій, важливих для формування обґрунтованої стратегії розвитку бізнесу [1].

Водночас бізнес-аналітики зіштовхуються з низкою проблем, пов’язаних зі збором та обробкою великих масивів даних: від неоднорідності структури веб-сторінок і складностей доступу через технічні або правові обмеження до проблем валідації, фільтрації та очищення інформації. Ці виклики потребують комплексного підходу до пошуку ефективних рішень.

З огляду на це, метою даної роботи є розробка автоматизованої системи збору та аналізу веб-даних, яка дозволить ефективно вирішувати зазначені проблеми та забезпечить стабільну роботу з великими обсягами інформації з мережі. Система передбачає використання мови програмування Python, яка дозволяє реалізувати високопродуктивні інструменти збору та обробки даних у автоматичному режимі.

Запропоноване рішення стане дієвим інструментом для підприємств, оскільки забезпечить швидкий доступ до релевантної, достовірної інформації, необхідної для ухвалення стратегічно обґрунтованих рішень. Крім того, розробка враховує технічні, правові та етичні аспекти взаємодії з веб-ресурсами, а також гарантує безпеку й цілісність оброблюваної інформації.

Таким чином, робота спрямована на дослідження, проєктування та

створення комплексної автоматизованої системи збору та аналізу даних з веб-сайтів, яка сприятиме підвищенню ефективності бізнес-процесів і зміцненню конкурентних позицій підприємств на ринку.

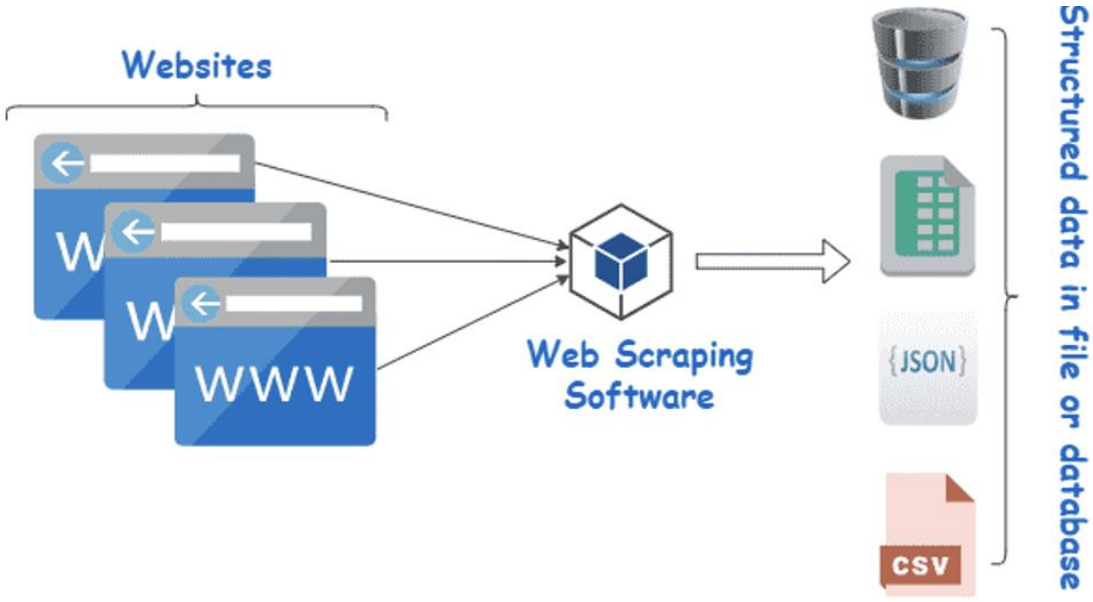


Рисунок 1.1 – Базова структура оглядємої системи

У сучасних умовах ведення бізнесу доступ до якісної та актуальної інформації є критично важливим фактором для успішного функціонування та збереження конкурентних позицій на ринку. Веб-скрапінг забезпечує широкі можливості для автоматизованого збору даних з різноманітних джерел, таких як сайти конкурентів, соціальні мережі, новинні ресурси тощо. Зібрана інформація може бути використана для аналізу ринку, прогнозування попиту на товари та послуги, а також для виявлення нових тенденцій і змін у поведінці споживачів.

Автоматизація процесів збору та обробки даних дозволяє значно скоротити витрати часу і ресурсів, які раніше були потрібні для ручного аналізу. Це забезпечує можливість оперативного реагування на зміни в ринковому середовищі, своєчасного виявлення потенційних ризиків і можливостей, що сприяє ухваленню ефективних рішень. Таким чином, дана робота не лише спрямована на розробку дієвої системи збору та аналізу веб-даних, а й покликана надати бізнесу практичний інструмент для підвищення ефективності управління

та зміцнення конкурентоспроможності.

У наступних розділах дисертації буде детально розглянуто технічні аспекти веб-скрапінгу, методи збору, обробки та аналізу отриманої інформації, а також варіанти її практичного застосування в умовах реального бізнес-середовища. Особлива увага буде приділена правовим та етичним аспектам використання веб-даних, оскільки дотримання законодавчих норм та етичних принципів є невід'ємною складовою легітимного впровадження подібних систем.

Розвиток і вдосконалення методів збору й аналізу інформації з веб-ресурсів має великий потенціал для застосування в різних галузях — від фінансів і маркетингу до наукових досліджень та управлінських рішень у державному секторі. Отримання доступу до структурованої інформації через веб-скрапінг може стати визначальним чинником для впровадження інновацій та досягнення успіху.

Розробка автоматизованої системи збору та аналізу даних також є кроком до цифрової трансформації підприємств, забезпечуючи оптимізацію бізнес-процесів та підвищення продуктивності. У рамках цієї дисертаційної роботи будуть проаналізовані відповідні техніки та інструменти, які можуть бути інтегровані у діяльність компаній для ефективного використання інформації з веб-ресурсів.

Отже, робота охоплює як технічну реалізацію системи збору та аналізу даних, так і практичне застосування її результатів у реальних бізнес-сценаріях. Вона також створює підґрунтя для подальших досліджень у сфері веб-скрапінгу, сприяючи покращенню аналітичних можливостей підприємств і підтримці прийняття обґрунтованих рішень.

1.2 Мета та задачі роботи

Основною метою цієї роботи є створення та впровадження автоматизованої системи збору й аналізу даних з веб-ресурсів для підвищення ефективності

прийняття стратегічних бізнес-рішень. Така система повинна бути функціональною, стабільною та здатною працювати з різними джерелами інформації в Інтернеті. Окрему увагу приділено вивченню правових та етичних аспектів застосування технологій веб-скрапінгу [2].

Робота над проєктом передбачає досягнення низки конкретних завдань:

– розробка програмного інструменту для веб-скрапінгу - створення програмного рішення, яке забезпечує зручний і стабільний збір інформації з різних веб-сайтів. Інструмент повинен бути адаптивним до різних типів джерел та мати можливість швидкого налаштування під нові сайти;

– аналіз особливостей різних типів веб-джерел - вивчення структури та специфіки функціонування різних сайтів, що є потенційними джерелами даних, з метою розробки ефективних методів збору та обробки інформації для кожного з них;

– розробка алгоритмів обробки, аналізу та візуалізації дан**их** - створення алгоритмів, здатних працювати з великими обсягами даних, виділяючи ключові закономірності, тренди та залежності, що сприятимуть прийняттю рішень;

– дослідження правових і етичних аспектів - вивчення чинного законодавства щодо збору даних з відкритих джерел, а також етичних норм, що регламентують використання інформації з веб-ресурсів.

Робота всебічно розглядає наведені завдання, щоб продемонструвати потенціал автоматизованої системи збору та аналізу даних для застосування в бізнес-середовищі та наукових дослідженнях. Окремо буде проаналізовано труднощі, з якими можуть зіткнутися користувачі веб-скрапінгу, та запропоновані можливі шляхи їх подолання.

Слід підкреслити, що створення універсального рішення для збору даних з усіх типів веб-сайтів є практично неможливим через унікальність кожного ресурсу. Веб-скрапінг - це ефективний, але складний інструмент, який потребує індивідуального підходу до кожного сайту. Це пов'язано з відмінностями у структурі HTML-коду, методах захисту, доступі до даних та використанні

динамічного контенту.

Процес скрапінгу включає вилучення даних з веб-сторінок, їх трансформацію в структурований формат та збереження для подальшої обробки. Врахування унікальних особливостей веб-сайтів — таких як структура розмітки, використання JavaScript, автентифікація та інші механізми обмеження — є критичним етапом у розробці парсера.

Розуміння індивідуальності веб-ресурсів вимагає глибоких знань у сфері веб-технологій: HTML, CSS, JavaScript, а також досвіду у роботі з інструментами для автоматизації, як-от Selenium або Puppeteer, у випадках динамічних сайтів. У свою чергу, для простіших, статичних веб-ресурсів можуть бути застосовані базові бібліотеки для парсингу.

Отже, дана робота поєднує в собі технічні, правові та практичні аспекти розробки ефективної автоматизованої системи збору даних, яка здатна враховувати індивідуальні характеристики джерел та забезпечити корисну інформацію для бізнес-аналітики.

Розробникам веб-скраперів необхідно постійно враховувати зміни, що відбуваються на веб-сайтах, з яких здійснюється збір даних. Оскільки сайти регулярно оновлюють структуру, оформлення та технології, це може негативно впливати на стабільність та точність роботи скрапера. Тому регулярна підтримка, моніторинг і своєчасне оновлення інструментів скрапінгу є невід’ємною складовою процесу їхньої експлуатації.

Успішна розробка веб-скрапера вимагає не лише глибоких технічних знань, а й гнучкості, уваги до деталей та здатності швидко адаптуватися до змін у зовнішньому середовищі. Особливо важливим є правильний вибір інструментів та бібліотек, які найкраще підходять для роботи з конкретним веб-ресурсом. Тип сайту, його структура, використовувані технології та спосіб представлення даних безпосередньо впливають на технічні рішення, що приймаються під час розробки.

Для статичних сайтів із простою HTML-структурою можуть використовуватись базові інструменти для аналізу HTML-коду. Натомість динамічні сайти, що активно застосовують JavaScript для завантаження контенту,

часто потребують складніших рішень, таких як браузерна автоматизація (наприклад, через Selenium або Puppeteer), щоб забезпечити доступ до необхідних даних.

Окремі сайти можуть використовувати захисні механізми, такі як CAPTCHA, обмеження доступу за IP-адресами, автентифікацію або інші методи протидії автоматизованому збору даних. У таких випадках можуть знадобитися спеціалізовані рішення або додаткова логіка для обходу цих обмежень. Якщо сайт надає офіційне API для доступу до інформації, цей варіант зазвичай є найбільш ефективним і безпечним. API, як правило, забезпечують структуровані дані у форматах JSON або XML, що значно спрощує процес їх обробки.

Окрему увагу слід приділити юридичним та етичним аспектам. Під час здійснення веб-скрапінгу необхідно дотримуватися чинного законодавства і поважати правила використання контенту, визначені власниками сайтів, щоб уникнути правових наслідків та конфліктів.

Оскільки веб-сайти постійно змінюються, розробка ефективного веб-скрапера вимагає адаптивного та індивідуального підходу до кожного ресурсу. Неможливо створити універсальний скрапер, який би працював однаково добре для всіх сайтів, тому кожен проект потребує окремого аналізу, налаштування та подальшої підтримки. У цьому контексті глибоке розуміння архітектури веб-ресурсів і гнучкість інструментів є ключовими факторами успішної реалізації проекту.

1.3 Завдання та актуальність роботи

Завдяки веб-скрапінгу вирішується широкий спектр завдань у різних галузях, де необхідний доступ до великої кількості актуальної інформації з Інтернету. Ось основні приклади застосування цього інструменту:

- аналіз ринку - у сфері бізнесу веб-скрапінг використовується для

моніторингу цін, рекламних пропозицій і наявності товарів на сайтах конкурентів. Це дає змогу компаніям формувати ефективну цінову політику, оперативно реагувати на зміни ринку та оцінювати попит на продукцію чи послуги.

– соціологічні дослідження - дослідники застосовують веб-скрапінг для збору та аналізу даних із соціальних мереж, форумів і новинних ресурсів. Це дозволяє виявляти суспільні настрої, домінуючі думки, актуальні теми та загальні тенденції поведінки споживачів і громадськості.

– медіа-моніторинг - представники ЗМІ та інформаційних агентств використовують веб-скрапінг для автоматичного збирання новинних матеріалів, статей, зображень і відео з різних інтернет-ресурсів. Це забезпечує швидкий доступ до контенту для подальшого аналізу, систематизації та публікацій.

– наукові дослідження - у багатьох наукових галузях веб-скрапінг став важливим інструментом для збирання великих обсягів даних. Зокрема, в медицині цей підхід дозволяє аналізувати медичні публікації, бази даних (із дотриманням норм конфіденційності), спеціалізовані форуми та інші джерела інформації щодо захворювань, методів лікування та профілактики. Такі дані використовуються для аналізу епідеміологічних трендів, моніторингу реакції суспільства на медичні події або кампанії.

Окрім медицини, веб-скрапінг широко застосовується:

– в екології - для моніторингу змін у навколишньому середовищі;

– в урбаністиці - для аналізу міського розвитку та транспортної інфраструктури;

– у соціальних науках - для вивчення моделей поведінки, культурних змін і суспільної динаміки.

Автоматизація збору даних за допомогою веб-скрапінгу значно розширює можливості дослідників, надаючи доступ до таких обсягів інформації, які були б неможливі для ручного збору (рис.1.2). Це відкриває нові перспективи для глибокого аналізу, наукових відкриттів та обґрунтованих висновків у різних галузях знань.

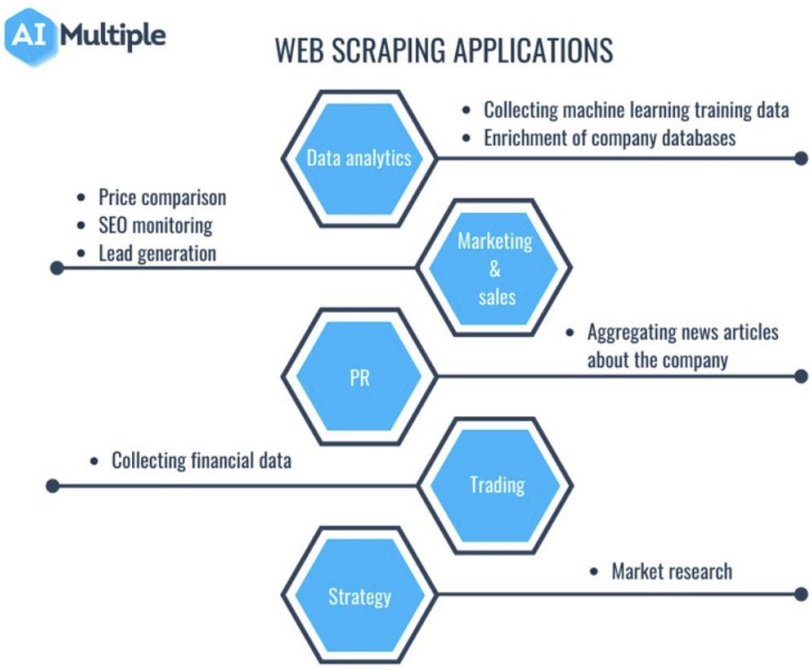


Рисунок 1.2 – Додаток для вебскрапінгу

Розробка Інтернет-агрегаторів: Веб-скрапінг застосовується для побудови Інтернет-агрегаторів, які збирають новини, оголошення, рецепти та іншу інформацію з різноманітних джерел і об'єднують її в одному зручному для користувача ресурсі (рис.1.3).

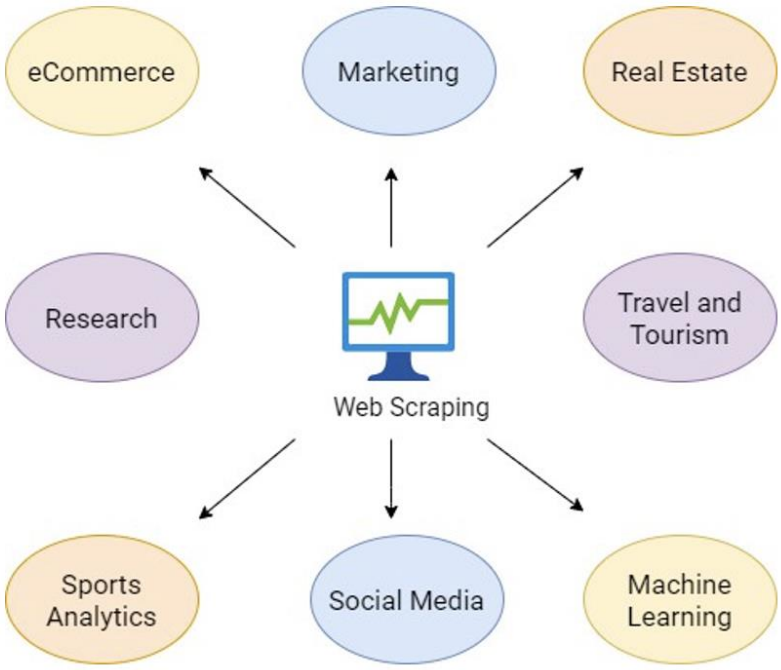


Рисунок 1.3 – Принцип роботи веб-скрапінгу

Актуальність веб-скрапінгу. Веб-скрапінг набуває все більшого значення в умовах сучасного інформаційного середовища.

Постійне зростання обсягів даних в Інтернеті: Інтернет забезпечує доступ до величезних масивів інформації, і веб-скрапінг слугує ефективним інструментом для їх збору та подальшого аналізу. Із розвитком цифрових технологій та онлайн-контенту обсяги даних, доступних для веб-скрапінгу, стрімко зростають (рис. 1.4).

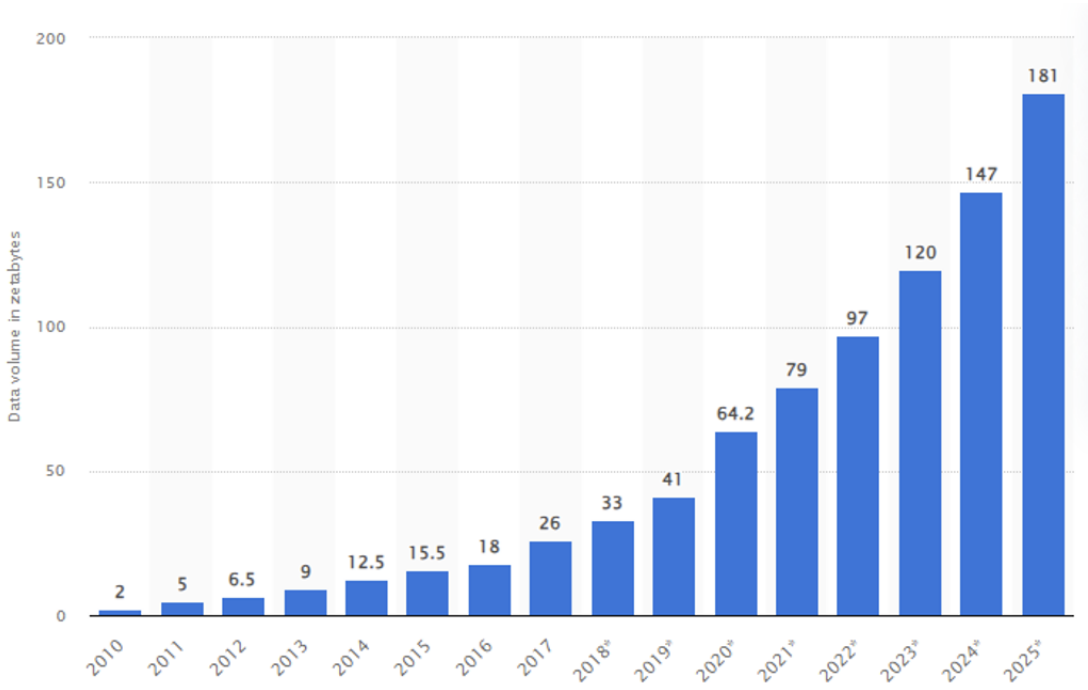


Рисунок 1.4 – Збільшення обсягів даних у мережі Інтернет у зетабайтах

Веб-скрапінг відкриває широкі можливості для збору та обробки даних у різних сферах - від маркетингових досліджень і конкурентного аналізу до академічних досліджень і громадських ініціатив. Зі збільшенням обсягу доступної онлайн-інформації його роль стає дедалі важливішою для тих, хто прагне працювати з великими масивами даних. Автоматизований збір інформації забезпечує високу точність і швидкість аналізу тенденцій, виявлення закономірностей і формування прогнозів, що значно перевищує можливості традиційних ручних методів.

Разом з тим, зростання обсягів доступної інформації створює нові виклики.

Окрім складності обробки великих даних, важливим аспектом є дотримання юридичних та етичних норм. Використання веб-скрапінгу потребує чіткого розуміння правових обмежень, пов’язаних із доступом до даних, а також відповідального ставлення до конфіденційності та авторських прав.

Таким чином, у цифрову епоху веб-скрапінг залишається ключовим інструментом, що забезпечує доступ до цінної інформації з різноманітних джерел для покращення бізнес-процесів, наукових досліджень та суспільних ініціатив.

Значення аналізу даних**.** Сучасні процеси прийняття рішень у бізнесі, науці та громадянському суспільстві дедалі більше базуються на глибокому аналізі даних. Веб-скрапінг є ефективним способом автоматизованого збору інформації з численних веб-ресурсів, що значно підвищує обсяги та швидкість обробки порівняно з традиційними методами.

У бізнес-середовищі це дозволяє отримувати актуальні дані про ринок, конкурентів, ціни та споживчі настрої, що сприяє оперативній адаптації до змін, оптимізації стратегій і підвищенню ефективності маркетингу. Також веб-скрапінг корисний для моніторингу репутації бренду й збору клієнтського фідбеку, що є важливим для збереження високої лояльності клієнтів.

У науковій сфері метод широко застосовується для збору великих обсягів інформації з наукових публікацій, баз даних і академічних порталів. Це полегшує вивчення тенденцій, перевірку гіпотез і здійснення наукових відкриттів, особливо у сферах, де потрібна швидка обробка масивів даних, таких як біоінформатика, екологія та соціологія.

У громадському секторі веб-скрапінг використовується для аналізу публічних заяв, новин, соціальних мереж та урядових повідомлень. Це дає змогу активістам, науковцям і політикам ефективно відслідковувати соціальні, політичні та екологічні процеси, сприяючи більшій прозорості та підзвітності в управлінні.

Отже, веб-скрапінг стає ключовим ресурсом для збору, дослідження й інтерпретації даних у різних галузях. Він дозволяє глибше розуміти складні процеси та формувати ефективні стратегії й політики, спрямовані на сталий

розвиток і прийняття обґрунтованих рішень.

Необхідність моніторингу та конкурентної розвідки**.** У бізнес-середовищі критично важливо своєчасно відслідковувати дії конкурентів і зміни на ринку. Веб-скрапінг забезпечує компаніям інструменти для оперативного збору великих обсягів інформації з онлайн-ресурсів — про ціни, нові продукти, маркетингові стратегії та відгуки споживачів. Такий підхід дозволяє швидко аналізувати ринкові тенденції та адаптувати бізнес-стратегії відповідно до змін у поведінці клієнтів і дій конкурентів, підтримуючи конкурентоспроможність компанії.

Перспективи розвитку веб-скрапінгу**.** З огляду на стрімке зростання обсягів цифрової інформації та потребу в її обробці, веб-скрапінг стає все більш актуальним. Цей тренд зумовлює інтерес до нових підходів до збору даних, що враховують динаміку цифрового середовища та орієнтуються на створення інноваційних інструментів для ефективного аналізу великих масивів інформації.

Одним із ключових напрямів розвитку є інтеграція веб-скрапінгу з технологіями штучного інтелекту та машинного навчання. Це дозволяє не лише підвищити ефективність збору даних, а й виконувати глибший аналітичний розбір - виявляти закономірності, тренди та приховані зв’язки, які раніше були недоступні. Такі технології також допомагають обходити анти-скрапінгові механізми, що дедалі частіше впроваджуються на сучасних веб-сайтах.

Водночас питання правового й етичного характеру залишаються актуальними. Регулювання у сфері захисту персональних даних, зокрема такі акти, як GDPR у ЄС, істотно впливають на способи збору та використання інформації. Компанії повинні дотримуватися встановлених норм, щоб уникнути юридичних ризиків, водночас балансуючи між необхідністю збору даних і захистом прав користувачів на приватність.

Іншим перспективним вектором розвитку є впровадження технологій блокчейн. Вони здатні підвищити прозорість і безпеку обробки даних, забезпечити їх достовірність і простежуваність, що особливо важливо при використанні зібраної інформації для прийняття рішень або досліджень.

На тлі зростання значення обробки великих даних веб-скрапінг набуває

особливої важливості не лише в комерційній сфері, а й у науковій, освітній та соціальній діяльності. Компанії та організації продовжуватимуть інвестувати в розвиток інструментів веб-скрапінгу, що стане запорукою конкурентних переваг у сфері збору й аналізу інформації.

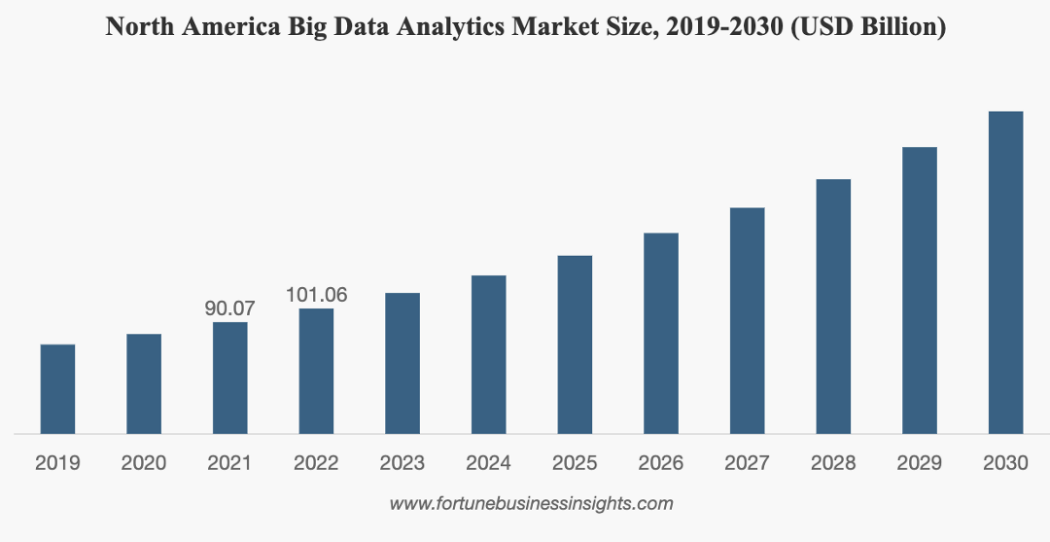


Рисунок 1.5 – Рівень зростання попиту вебскраппінгу

Підсумовуючи, майбутнє веб-скрапінгу прогнозується як динамічне, з новими можливостями та викликами. Ця сфера й надалі впливатиме на розвиток бізнесу, технологій і суспільства, підкреслюючи ключову роль даних у сучасному світі. За оцінками експертів, ринок програмного забезпечення для веб-скрапінгу зросте з 0,54 млрд доларів у 2021 році до 1,15 млрд доларів у 2027-му (зростання на 113%). Дані сьогодні - це нова нафта, і компанії будь-якого масштабу обробляють їх у величезних обсягах.

1.4 Висновки до розділу

Веб-скрапінг, як сучасний метод автоматизації збору даних, відкриває нові можливості для аналітичних відділів компаній і стає необхідною навичкою дослідників у цифрову епоху. Автоматичний збір інформації зі різних веб-

ресурсів дозволяє глибоко аналізувати ринкові умови, що є важливою складовою стратегічного управління бізнесом [3].

Використання веб-скрапінгу для аналізу ринку дає компаніям унікальну можливість відстежувати зміни в попиті, асортименті та цінових стратегіях конкурентів, а також вивчати поведінку споживачів для розробки більш ефективних маркетингових кампаній і оптимізації логістики.

У соціальних науках цей метод змінює підхід до досліджень, дозволяючи збирати детальні дані про громадські настрої та суспільні рухи, що розширює науковий потенціал і відкриває нові перспективи, недоступні традиційними методами.

Науковці використовують веб-скрапінг для збору великих масивів даних із відкритих баз, журналів і публікацій, що прискорює наукові відкриття та підтримує міждисциплінарні дослідження, особливо в галузях, де потрібні великі датасети — біоінформатиці, соціальних науках тощо.

Разом з перевагами веб-скрапінг викликає важливі правові та етичні питання, що будуть розглянуті у наступному розділі. Захист приватності, дотримання законів на кшталт GDPR та інтелектуальної власності вимагають відповідального підходу. Розробка політик і процедур регулювання використання веб-скрапінгу є пріоритетом для забезпечення довіри та безпеки.

Майбутнє веб-скрапінгу пов’язане з подальшою інтеграцією у бізнес-процеси і наукові дослідження, адаптацією до змін цифрового середовища та розвитком нових методів аналізу даних. Він залишатиметься ключовим інструментом для доступу до актуальної інформації і прийняття обґрунтованих рішень на всіх рівнях.

Сучасні сервіси та інструменти полегшують веб-скрапінг, роблячи його доступним для користувачів без глибоких технічних знань, що сприяє інноваціям і підвищенню конкуренції, дозволяючи більшій кількості організацій використовувати дані для вдосконалення продуктів і послуг.

У межах цієї роботи веб-скрапінг розглядається як стратегічний ресурс для підвищення інноваційності та конкурентоспроможності компаній і організацій, з

акцентом на вдосконаленні методів збору і обробки даних, а також на етичних і правових засадах їх використання.

Очікується, що результати дослідження сприятимуть кращому розумінню та застосуванню веб-скрапінгу у різних сферах — від повсякденних потреб до масштабних бізнес-рішень і наукових проєктів, а також вкажуть нові напрямки розвитку інструментів і стратегій його використання.

2 ЗБІР ДАНИХ З ВЕБСАЙТІВ

2.1 Поняття веб-скрапінгу як методу збору даних

Веб-скрапінг - це ефективний інструмент автоматичного збору даних з веб-сайтів, який дозволяє отримувати великі обсяги інформації для бізнес-аналізу, досліджень ринку, моніторингу цін та інших задач. У цьому розділі розглянуто основні технічні аспекти веб-скрапінгу, зокрема вибір інструментів і методологію збору даних [4].

Серед популярних інструментів для веб-скрапінгу - бібліотеки Python, такі як BeautifulSoup і Scrapy, що забезпечують гнучкість у виборі елементів сторінок для збору інформації. Одним із викликів є обхід обмежень веб-сайтів, зокрема лімітів на частоту запитів та захисту від ботів. Для уникнення блокувань використовують методи ротації IP-адрес і затримки між запитами.

Наступний етап - парсинг, що полягає в аналізі HTML-коду та вилученні потрібних даних. Це ускладнюється на динамічних сайтах, де контент формується за допомогою JavaScript. Останнім кроком є збереження отриманих даних у структурованих форматах - CSV, JSON або базах даних, що дозволяє подальший аналіз і підтримку бізнес-рішень.

Основні переваги веб-скрапінгу:

– автоматизація та ефективність - автоматизований збір даних значно підвищує ефективність у порівнянні з ручним збором, який є трудомістким і

схильним до помилок. Веб-скрапінг дозволяє швидко збирати, обробляти та класифікувати великі обсяги інформації з різних джерел, знижуючи трудовитрати й витрати, особливо у великих проектах. Автоматизація мінімізує людські помилки, забезпечуючи високу точність і надійність даних.

– доступність даних - веб-скрапінг забезпечує простий доступ до величезної кількості інформації в інтернеті з різних сайтів і платформ. Автоматизовані системи можуть регулярно оновлювати зібрані дані, підтримуючи їх актуальність і релевантність. Наприклад, для моніторингу цін на фондовому ринку оновлення може відбуватися щохвилини, тоді як для звичайних товарів достатньо оновлення раз на тиждень.

– гнучкість та адаптивність - веб-скрапінгові системи легко пристосовуються до різних форматів веб-сторінок і типів даних, включно зі сторінками з частими змінами дизайну чи контенту. Вони можуть інтегруватися з іншими системами та аналітичними інструментами, підвищуючи ефективність бізнес-процесів. Великі компанії активно використовують такі системи для аналізу ринку, моніторингу конкурентів та інших цілей.

При виборі інструментів для веб-скрапінгу слід враховувати:

– масштаб задачі:

– типи даних:

– складність веб-сайтів;

– вимоги до частоти оновлення даних.

2.2 Вибір програмного забезпечення для системи

Для невеликих проєктів достатньо простих засобів, як-от BeautifulSoup або скриптів на Python, які ефективно обробляють десятки чи сотні сторінок. Для великих завдань із тисячами сторінок потрібні складніші рішення, наприклад, Scrapy або Selenium (табл..2.1) [5,6].

Таблиця 2.1 - Порівняння програмного забезпечення

Framework	Selenium	Scrapy
Конфіденційність	Тестування браузера	API веб-скрапінгу
Підтримка JavaScript	Дуже гарно підтримує	Багато часу для перевірки і розробки для стимулювання запросів
Типи даних	Добре для невеликих даних	Добре працює з великим даними
Швидкість	швидкий	Дуже швидкий
Вимоги до оновлення	Не досить вимогливий	проста розробка проміжного програмного забезпечення або додавання функцій користувача
Складність	Досить легко для новачків	Досить важка для новачків

Selenium створений для тестування веб-додатків, але широко використовується для веб-скрапінгу завдяки здатності імітувати дії користувача в реальному браузері, виконувати JavaScript і працювати зі складними динамічними сайтами. Його недоліки - високе споживання ресурсів і повільніша робота.

Selenium - це потужний інструмент з відкритим кодом, який широко використовується для автоматизації взаємодії з веб-браузерами. Він дозволяє імітувати дії користувача на веб-сторінці, такі як клік по кнопках, заповнення форм, навігація між сторінками, прокручування, очікування елементів тощо. Завдяки цьому Selenium став незамінним у сферах автоматизованого тестування веб-додатків, збору даних (веб-скрапінгу), перевірки функціональності сайтів та моделювання поведінки користувачів.

Проєкт Selenium був започаткований у 2004 році Джейсоном Хагінсом (Jason Huggins) у компанії ThoughtWorks. З часом він набув широкого поширення й підтримується великою спільнотою розробників. Його гнучкість і багатоплатформенність дозволяють запускати автоматизовані сесії в різних браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari тощо) та операційних системах (Windows, macOS, Linux).

До складу Selenium входить кілька основних компонентів, кожен із яких має свої завдання.

Selenium WebDriver - основний інтерфейс, який дозволяє керувати браузером з різних мов програмування, таких як Python, Java, C#, JavaScript тощо. WebDriver напряму комунікує з браузером через драйвери, специфічні для кожного браузера (наприклад, ChromeDriver для Google Chrome або GeckoDriver для Firefox).

Selenium IDE - інтерактивне середовище для запису, відтворення й редагування тестових сценаріїв у браузері. Це розширення для браузерів Chrome або Firefox, яке дозволяє швидко створювати прості автоматизовані сценарії без написання коду. Хоча цей інструмент зручний для початківців, його функціональність є обмеженою порівняно з WebDriver

Selenium Grid - система, яка дозволяє виконувати тести паралельно на кількох машинах та браузерах, що особливо корисно для великих проєктів або розподілених систем. Grid забезпечує масштабованість і прискорює процес тестування чи збору інформації.

У контексті даної дипломної роботи Selenium розглядається як інструмент веб-скрапінгу — автоматизованого збору даних з веб-сайтів. На відміну від простих бібліотек для HTTP-запитів (наприклад, `requests`), Selenium дозволяє взаємодіяти з динамічними сторінками, що генеруються JavaScript-кодом, які не можна повністю обробити без запуску повноцінного браузера. Це особливо важливо у випадках, коли контент з'являється лише після натискання кнопок, прокручування сторінки чи асинхронного завантаження через AJAX.

Завдяки тому, що Selenium виконує дії так само, як це робить реальний користувач, він дозволяє обходити деякі технічні обмеження сайтів: взаємодіяти з капчами, динамічними меню, спливаючими вікнами, логінами тощо. Водночас використання Selenium вимагає значно більше ресурсів у порівнянні з "легкими" рішеннями, адже кожна сесія виконується у повноцінному браузері, що підвищує навантаження на систему та потребує додаткових засобів оптимізації.

Selenium підтримує різні режими запуску браузера, включаючи так званий «headless mode», коли браузер працює у фоновому режимі без графічного інтерфейсу. Це значно пришвидшує виконання скриптів, знижує використання

ресурсів та робить процес автоматизації менш помітним з боку систем виявлення ботів.

Інтеграція Selenium із такими бібліотеками, як `BeautifulSoup`, `pandas` або `openpyxl`, дозволяє не лише зібрати дані з веб-сторінки, а й обробити їх, зберегти у вигляді таблиць, побудувати звіти та візуалізувати зміни у динаміці. У поєднанні з Django або іншими фреймворками, Selenium може стати частиною більших інформаційних систем, що використовуються для моніторингу ринку, цін, наявності товарів або змін у веб-контенті.

Однак важливо також зважати на етичні й правові аспекти використання Selenium. Автоматизований збір даних не завжди дозволений власниками веб-сайтів і може суперечити їхнім умовам використання. Тому при розробці скриптів слід враховувати правила сайту, уникати надмірного навантаження на сервери та дотримуватись чинного законодавства.

Підсумовуючи, Selenium - це універсальний і потужний інструмент, який забезпечує гнучку автоматизацію взаємодії з веб-ресурсами, є незамінним як у тестуванні веб-додатків, так і в задачах з веб-скрапінгу. Його використання в дипломному проєкті дозволяє реалізувати надійну систему моніторингу та аналізу веб-даних із широкими можливостями для масштабування та адаптації до змін у структурі сайтів.

Scrapy - спеціалізований і високопродуктивний фреймворк для збору даних, який швидко обробляє запити, налаштовує парсинг і збереження даних, має вбудовані засоби обходу блокувань (користувацькі агенти, проксі). Однак Scrapy не може виконувати JavaScript і взаємодіяти зі сторінками, що потребують дій користувача, як-от прийняття cookie або згоди. Крім того, Scrapy - виключно Python-фреймворк, тоді як Selenium підтримує кілька мов.

Scrapy — це потужний, високопродуктивний фреймворк з відкритим кодом, розроблений спеціально для веб-скрапінгу, автоматичного збору інформації з веб-сайтів та створення ботів для обробки веб-контенту. Він написаний мовою програмування Python і є одним із найпопулярніших інструментів для побудови систем збору даних із веб-джерел завдяки своїй гнучкості, масштабованості та

ефективності.

Scrapy дозволяє автоматизувати процеси навігації сторінками, збору HTML-вмісту, обробки структурованих і неструктурованих даних, а також подальшої їх передачі у формати CSV, JSON, XML або збереження у базах даних. Основним завданням цього фреймворку є витяг релевантної інформації зі сторінок за допомогою механізмів парсингу та селекторів XPath або CSS.

Ключовими перевагами Scrapy є:

– швидкодія та асинхронність - Scrapy побудований на основі асинхронної бібліотеки Twisted, що дозволяє виконувати сотні запитів паралельно, не блокуючи процес, це значно пришвидшує скрапінг великих обсягів даних і мінімізує час простою між запитами до серверів;

– гнучка архітектура - Scrapy реалізує архітектуру на основі «павуків» (spiders) окремих класів, які описують логіку збору даних з конкретного ресурсу, завдяки цьому легко розширювати функціональність, адаптувати логіку під різні структури сайтів або реалізовувати повторне використання компонентів;

– розвинена система середніх шарів (middlewares) - Scrapy підтримує можливість використання користувацьких або стандартних проміжних обробників, які перехоплюють та модифікують HTTP-запити та відповіді, що відкриває широкі можливості для обходу обмежень, авторизації, динамічного налаштування заголовків, чергування User-Agent, використання проксі-серверів тощо;

– інтеграція з інструментами обробки даних - зібрані дані можна зручно передавати в сторонні бібліотеки, як-от `pandas`, або ж відправляти до баз даних MySQL, PostgreSQL, MongoDB, Elasticsearch, що дозволяє легко інтегрувати Scrapy у більш складні інформаційні системи;

– розширення та підтримка - Scrapy активно розвивається, має велике співтовариство та безліч додаткових плагінів і розширень, що дозволяють, наприклад, організовувати зберігання кешу, вести логування, обробляти JavaScript-сторінки через інтеграцію з Selenium або Splash, а також працювати з API.

У контексті дипломного проєкту Scrapy виступає як основа для створення системи моніторингу даних з веб-сайтів. Завдяки гнучкості Scrapy можна легко реалізувати окремі класи скраперів для різних сайтів, визначивши структуру елементів, з яких слід збирати дані (наприклад, назва продукту, ціна, посилання, дата оновлення тощо). Важливою перевагою є можливість адаптації до змін у DOM-структурі сайтів без повного переписування логіки скрапінгу.

Одним із найважливіших аспектів у роботі зі Scrapy є правильне використання механізмів затримки запитів (download delay) та обмеження кількості одночасних підключень, щоб уникнути блокування з боку серверів. Фреймворк дозволяє централізовано керувати такими налаштуваннями у файлі конфігурації `settings.py`, що забезпечує контрольовану та відповідальну взаємодію з джерелами даних.

Також Scrapy має вбудовану підтримку логування, повторних спроб при помилках, перенаправлень, обробки cookies, а також формування черги запитів — усе це значно спрощує роботу над проєктами, що потребують стабільного й надійного збору даних у великих обсягах.

Для обробки сайтів із динамічним контентом Scrapy може працювати у зв’язці з браузерами або рендеринг-сервісами, наприклад, Splash або Selenium, що дозволяє завантажувати сторінки з JavaScript та отримувати вміст, недоступний у звичайних HTTP-запитах. Це робить Scrapy придатним навіть для найскладніших задач веб-скрапінгу.

Завдяки своїй архітектурі Scrapy також зручно використовувати для побудови систем постійного моніторингу сайтів. Наприклад, можна реалізувати автоматичний запуск скраперів за розкладом, зберігання історії цін на товари та генерацію звітів про зміни. Таким чином, Scrapy є надійною технологічною базою для побудови розумних систем аналізу ринку, конкурентного моніторингу та інших бізнес-задач.

У підсумку, Scrapy - це повноцінний фреймворк, який поєднує в собі потужні засоби збору, обробки та збереження веб-даних, а також широкі можливості для інтеграції, масштабування та автоматизації. Його використання в

дипломному проєкті дозволяє реалізувати ефективну, гнучку та розширювану систему збору інформації з веб-ресурсів, що є важливим інструментом в епоху даних.

Враховуючи ці фактори та специфіку нашого дослідження ринку українських магазинів техніки, де оновлення не потрібне щохвилини, обмежень на ресурси немає, а сайти використовують JavaScript та мають динамічний контент із розділами, що вимагають імітації дій користувача, Scrapy не підходить. Тому оберемо комбінацію BeautifulSoup4 та Selenium як оптимальний інструментарій.

Збір текстової інформації відносно простий і підтримується більшістю інструментів веб-скрапінгу. Для завантаження зображень, відео чи аудіо потрібні спеціалізовані методи та бібліотеки, які забезпечують збереження мультимедійних файлів.

Сайти зі складними JavaScript-фреймворками потребують спеціальних інструментів для збору даних [7]. Складна структура та захист сторінок впливають на методи скрапінгу. Завдяки використанню Selenium ці проблеми для нас мінімальні. Водночас викликом може бути динамічна обфускація даних (рис. 2.1) на сайті.

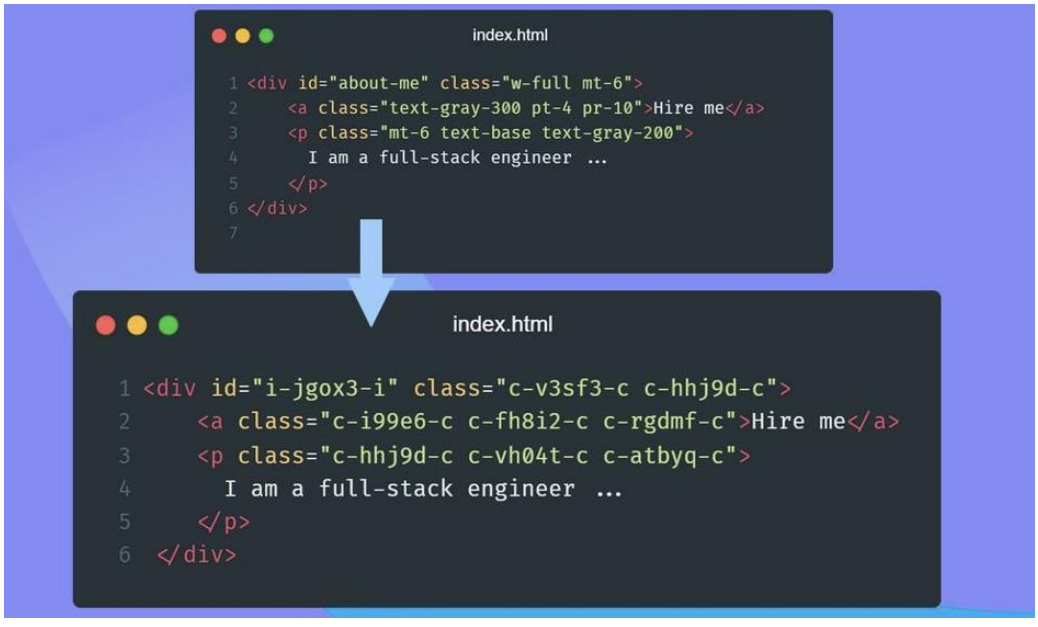


Рисунок 2.1 – Приклад захисту даних

Обфускація коду - це процес перетворення зрозумілого коду у важкочитаний формат, що ускладнює його аналіз, копіювання чи зміну. Розробники сайтів застосовують цю техніку для захисту коду, зокрема шляхом перейменування змінних на незрозумілі символи, зміни структури коду без втрати функціональності або додавання зайвих фрагментів [8]. Обфускований код створює значні труднощі для веб-скрапінгу, оскільки складно визначити потрібні елементи сторінки для збору даних. Крім того, обфускація може приховувати антискрапінгові заходи, як-от пастки для ботів або підставні дані. Ефективним способом роботи з такими сайтами є аналіз DOM дерева, що відображає структуру веб-сторінки. Оскільки обфускація здебільшого стосується скриптів, а не HTML-структури, аналіз DOM дозволяє отримувати доступ до даних навіть при замаскованих скриптах. Сучасні браузери надають інструменти розробника для перегляду, аналізу та роботи з DOM, включно з вибором елементів, переглядом властивостей і копіюванням XPath або CSS-селекторів.

Частота оновлення визначає необхідну інфраструктуру та стратегію збору. Одноразовий збір підходить для проектів з разовим отриманням даних і не потребує складних рішень. Для регулярного оновлення потрібна автоматизація, планувальники завдань або системи неперервного збору. У нашому випадку, щоб відстежувати зміни цін, веб-скрапінг виконуватиметься приблизно раз на тиждень за допомогою cron-job на віддаленому сервері.

2.3 Юридичний та етичний аспект моніторінгу даних з сайтів

Веб-скрапінг часто супроводжується правовими та етичними питаннями, що потребують уваги та дотримання певних рамок.

– авторське право та інтелектуальна власність - дані, що збираються, можуть бути захищені авторськими правами або іншими правами інтелектуальної власності. Веб-сайти часто обмежують копіювання чи поширення свого контенту.

– закони про конфіденційність і захист даних - збір персональної інформації може порушувати законодавство про захист приватності. Важливо дотримуватися правил обробки та використання таких даних.

– умови використання та запити - багато сайтів мають політику, що обмежує або забороняє веб-скрапінг. Ігнорування цих правил може призвести до юридичних наслідків, блокування IP або судових позовів.

– транспарентність - необхідно повідомляти користувачів про мету та обсяг збору даних;

– заборона на скрапінг - якщо сайт забороняє скрапінг, це треба поважати. Порушення призводить до правових та етичних проблем, включно з блокуванням і судовими діями.

– етичність - збір даних має відповідати етичним нормам і законодавству, зокрема GDPR. Збирати слід лише необхідну інформацію, уникаючи зайвих або чутливих даних без дозволу. Забезпечуйте захист зібраних даних (шифрування, безпека), щоб запобігти несанкціонованому доступу.

– приклад LinkedIn - суворо забороняє автоматизований збір даних, що регламентується їх політикою. Порушення призводить до блокування акаунтів і юридичних проблем.

– етика взаємодії з серверами - необхідно уникати перевантаження серверів частими запитами. Важливо дотримуватись інтервалів між запитами, зберігати зібрані дані для мінімізації повторних звернень, коректно використовувати HTTP-хедери (User-Agent), враховувати вказівки файлу robots.txt. Оптимально виконувати запити в періоди мінімального навантаження (наприклад, вночі) та розподіляти трафік через різні IP для уникнення блокування. Такий підхід забезпечує етичність, стабільність і надійність процесу збору даних.

2.4 Висновки до розділу

У цьому розділі розглянуто ключові аспекти веб-скрепінгу - від вибору інструментів до методів подолання складнощів під час збору даних.

Вибір інструментів є важливим етапом: Selenium краще підходить для складних сайтів з JavaScript і імітації користувацьких дій, тоді як Scrapy ефективніший для масштабного збору зі статичних або простіших сторінок.

Фактори, що впливають на підхід до скрепінгу: масштаб задачі, типи даних, складність сайту та частота оновлення інформації - були детально проаналізовані.

Особливу увагу приділено обфускації коду як виклику для збору даних та стратегіям подолання цього, зокрема аналізу DOM, використанню інструментів дебагінгу та інших методів, які дозволяють ефективно отримувати інформацію незважаючи на складнощі.

Таким чином, розділ забезпечує комплексне розуміння технічних нюансів веб-скрепінгу, що необхідне для розробки надійних систем збору даних, підкреслюючи важливість адаптивності і постійного оновлення знань у динамічному середовищі інтернет-технологій.

Виходячи з аналізу, обрано комбінацію Selenium та BeautifulSoup4 як стратегічне рішення, що поєднує гнучкість і потужність обох інструментів. Selenium використовується для навігації та взаємодії зі сторінками, а BeautifulSoup - для парсингу HTML і вилучення необхідних даних.

3 РОБОТА З ДАНИМИ

3.1 Збір та аналіз даних

Цей розділ присвячено розробці універсального класу BaseScraper, який слугуватиме основою для збору даних із різних джерел, зокрема онлайн-

магазинів.

Клас BaseScraper створений як гнучкий інструмент, що дозволяє легко адаптувати процес скрепінгу під різні інтернет-магазини завдяки модульній структурі та параметризації ключових компонентів. Його універсальність і адаптивність забезпечують ефективний збір даних із різних сайтів із врахуванням їхніх особливостей [9].

Методи класу BaseScraper та короткий опис їх функцій та можливостей наведені у таблиці 3.1.

Таблиця 3.1 - Методи класу BaseScraper та короткий опис їх функцій та можливостей

Метод	Опис функції	Можливості
`__init__(self, config)`	Ініціалізує об'єкт класу з конфігурацією скрепінгу	Приймає параметри для налаштування (URL, таймаути, тощо)
`start_requests(self)`	Запускає процес надсилання початкових HTTP-запитів	Генерує початкові URL для скрепінгу
`fetch_page(self, url)`	Завантажує HTML-сторінку за вказаною URL	Використовує Selenium або requests для отримання сторінки
`parse(self, html)`	Аналізує HTML-код та витягує потрібні дані	Використовує BeautifulSoup для парсингу
`extract_data(self, parsed_html)`	Витягує конкретні елементи даних із розібраного HTML	Повертає структуровані дані (наприклад, словник)
`save_data(self, data)`	Зберігає зібрані дані у потрібному форматі	Підтримка збереження у файл, базу даних або API
`handle_pagination(self, html)`	Обробляє пагінацію для переходу між сторінками	Генерує URL наступних сторінок для подальшого скрепінгу

Продовження таблиці 3.1

Метод	Опис функції	Можливості
`run(self)`	Запускає повний цикл скрепінгу від отримання до збереження даних	Координує роботу всіх інших методів
`set_headers(self, headers)`	Налаштовує HTTP-заголовки для запитів	Дозволяє імітувати різні браузери або користувацькі агенти
`wait_for_element(self, selector)`	Очікує завантаження певного елемента на сторінці (для динамічних сайтів)	Використовує Selenium для роботи з динамічним контентом

Основні Особливості BaseScraper:

– гнучкість конфігурації - параметри `start_url` і `selectors` дозволяють налаштовувати клас для збору даних з різних сайтів, що робить BaseScraper універсальним інструментом без потреби створювати окремого робота для кожного ресурсу;

– використання Selenium та BeautifulSoup - поєднання цих інструментів забезпечує взаємодію з динамічними сайтами та ефективний парсинг і обробку необхідних даних;

– автоматизація браузера - Selenium дозволяє імітувати дії користувача, що необхідно для роботи з сайтами, які застосовують складні методи аутентифікації чи навігації, реалізуючи специфічну поведінку через наслідування класів;

– гнучка обробка даних - BeautifulSoup у Django дозволяє парсити та структурувати складні HTML/XML документи, витягуючи текст, посилання, теги та атрибути, що полегшує роботу з неструктурованим контентом і швидко виділяти важливу інформацію;

– масштабованість - завдяки ефективній архітектурі та використанню веб-драйверів BaseScraper легко масштабується для збору великих обсягів даних,

підходить для проектів будь-якого розміру;

– адаптація під конкретні завдання - приклади класи RozetkaScraper (рис.3.1) і ALLOScraper (рис.3.2), налаштовані на специфічні селектори та URL цих сайтів, демонструють, як BaseScraper просто пристосовується до різних джерел даних.

```
ROZETKA_SELECTORS = {
    'product_wrappers': 'div.goods-tile__content',
    'price_selector': 'p.product-price__big',
    'name_selector': 'h1',
    'photo_selector': 'li:nth-of-type(2) img.picture-container__picture'
}
```

Рисунок 3.1 – Скрапер магазину Розетка

```
ALLO_SELECTORS = {
    'product_wrappers': 'div.product-card__img',
    'price_selector': "div[itemprop='offers']",
    'name_selector': 'h1',
    'photo_selector': ".active img.main-gallery__image"
}
```

Рисунок 3.2 – Скрапер магазину АЛЛО

Застосування RozetkaScraper:

– збір даних про продукти - RozetkaScraper автоматизує збір інформації про товари (назви, ціни, зображення з сайту Rozetka);

– аналіз і моніторинг ринку - зібрані дані використовуються для вивчення ринкових тенденцій, споживацьких вподобань та конкурентного середовища;

– відстеження змін цін у часі - збереження дати збору даних дозволяє аналізувати динаміку цін, виявляти сезонні коливання та адаптувати бізнес-стратегії. Це сприяє кращому розумінню ринку, оптимізації ціноутворення та покращенню споживацького досвіду. Збереження дати є ключовим для ефективного збору і аналізу даних.

3.2 Збереження даних системи

Збереження даних у форматі JSON. У веб-скрапінгу важливо не лише збирати дані ефективно, а й належно їх зберігати. JSON став стандартом для серіалізації та зберігання структурованих даних завдяки своїй простій, читабельній людині та машині формі, а також широкій сумісності з різними мовами програмування [10].

Приклад коду показує, як легко перетворити зібрані дані у структурований формат JSON, що полегшує їх подальшу інтеграцію у складніші системи обробки.

Оскільки важливо відстежувати зміни цін з часом, збереження дати для кожного запису є критичним для аналізу ринкової динаміки. Це дозволяє відслідковувати цінові коливання, аналізувати тренди і реагувати на зміни в попиті та конкурентному середовищі. Веб-скрапінг також може збирати та аналізувати великі обсяги відгуків і оцінок споживачів, що допомагає бізнесу краще розуміти потреби клієнтів і покращувати маркетингові стратегії та продукти.

Переваги JSON:

– структурованість і читабельність - чітка структура полегшує аналіз та обробку даних;

– сумісність - широко підтримується більшістю мов програмування і платформ;

– гнучкість - легко змінювати або доповнювати структуру даних;

– компактність - зазвичай менший обсяг порівняно з XML, що знижує витрати на передачу та пришвидшує обробку;

– легкість парсингу - більшість мов програмування мають вбудовані бібліотеки або прості інструменти для парсингу та генерації JSON, що забезпечує зручність обробки та використання даних у програмах;

– сумісність з restful api - JSON є стандартним форматом передачі даних у RESTful API, що широко застосовуються у веб- та мобільній розробці.

Після збору й первинного збереження даних ефективним рішенням є їх відправка на сервер для зберігання в базі даних. Це забезпечує централізоване управління інформацією та спрощує доступ для подальшого аналізу та обробки.

Передача даних на сервер і їх збереження у базі є ключовим етапом для забезпечення безпеки, централізації та доступності. Встановлення з'єднання, передача JSON або іншого формату, а також фіксація дати збору відкривають широкі можливості для глибокого аналізу (табл..3.2).

Таблиця 3.2- Порівняння даних JSON та XML

JSON	**XML**
Має тип	Не мають типу
Типи: рядок, число, масив, логічний	Усі дані мають бути рядками
Дані легко доступні	Дані потрібно проаналізувати.
Підтримується більшістю браузерів	Розбір у різних браузерах може бути складним
Не має можливості відображення.	Пропонує можливість відображати дані, оскільки це мова розмітки.
Підтримує лише текстові та числові типи даних.	Підтримує різні типи даних, такі як числа, текст, зображення, діаграми, графіки тощо, надає параметри для передачі структури або формату даних із фактичними даними.
Отримати значення легко	Відновити цінність важко
Підтримується багатьма наборами інструментів Ajax	Не повністю підтримується набором інструментів Ajax
Повністю автоматизований спосіб десеріалізації/серіалізації JavaScript	Писати мають розробники JavaКод сценарію для серіалізації/десеріалізації з XML
Рідна підтримка об'єкта	Об'єкт має бути виражений конвенціями переважно пропущеним використанням атрибутів і елементів.
Підтримує лише кодування UTF-8.	Підтримує різні кодування.
не підтримує коментарі.	Підтримує коментарі.
Файли легше читати порівняно з XML.	Документи відносно складніше читати та інтерпретувати.
Не підтримує простори імен.	Підтримує простори імен.
Менш захищено.	Безпечніший, ніж JSON.

Цей крок особливо важливий при роботі з великими обсягами даних, що збираються через скрапінг. Зберігання на сервері гарантує надійність, швидкий доступ і зручність використання у бізнес-процесах, що є основою ефективного управління даними.

3.3 Основні показники для продуктивності системи

Обмежене використання даних і ресурсів. У розробці системи веб-скрепінгу ми акцентуємо увагу на ефективності та оптимізації ресурсів. Один із ключових підходів - вибірковий збір даних, зосереджений на основному вмісті веб-сторінки, що міститься в тегу .

Цей метод має кілька переваг:

- економія ресурсів - замість завантаження та обробки всієї сторінки, включно зі скриптами, стилями та метаданими, що зазвичай розміщені в шапці (header), ми працюємо лише з основним контентом. Це значно зменшує обсяг завантажених даних і, відповідно, навантаження на мережу та обчислювальні потужності (таблиця 3.3).

- підвищення ефективності обробки - аналіз і вилучення даних тільки з основного вмісту сторінки дозволяє прискорити обробку, що особливо важливо при масштабному зборі інформації, де кожна секунда має значення. Фокус на релевантних даних: Найчастіше ключова інформація міститься саме в тілі сторінки, тому цей підхід допомагає відсіяти зайвий контент і зосередитись на цінних даних.

Впровадження такої методики в нашій системі веб-скрепінгу відображає прагнення до максимальної ефективності та раціонального використання ресурсів, забезпечуючи при цьому високу точність і якість збору даних. Обсяг зекономлених ресурсів у відсотках наведено рисунку 3.3.

Таблиця 3.3 - Навантаження на мережу та обчислювальні потужності

Номер випробовування	Повний розмір сторінки (у байтах)	Розмір тегу body (у байтах)
1	777083	616289
2	873447	707934
3	708226	547763
4	230319	226847
5	257573	251597
6	255968	250062
7	1212865	974931
8	1147187	910194
9	1133872	902219

На рисунку 3.3 показано середню економію ресурсів на трьох різних сайтах.

– для перших трьох випробувань (Rozetka) середня економія склала 20,77%, що підтверджує ефективність вибіркового збору даних із фокусом на тег <body>;

– на сайті IStore (випробування 4-6) середня економія була лише 2,04%, що значно менше і може бути зумовлено відмінностями у структурі та дизайні цих сторінок;

– для останніх трьох випробувань (ALLO) середня економія становила 20,24%, що також свідчить про ефективність застосованої методики.

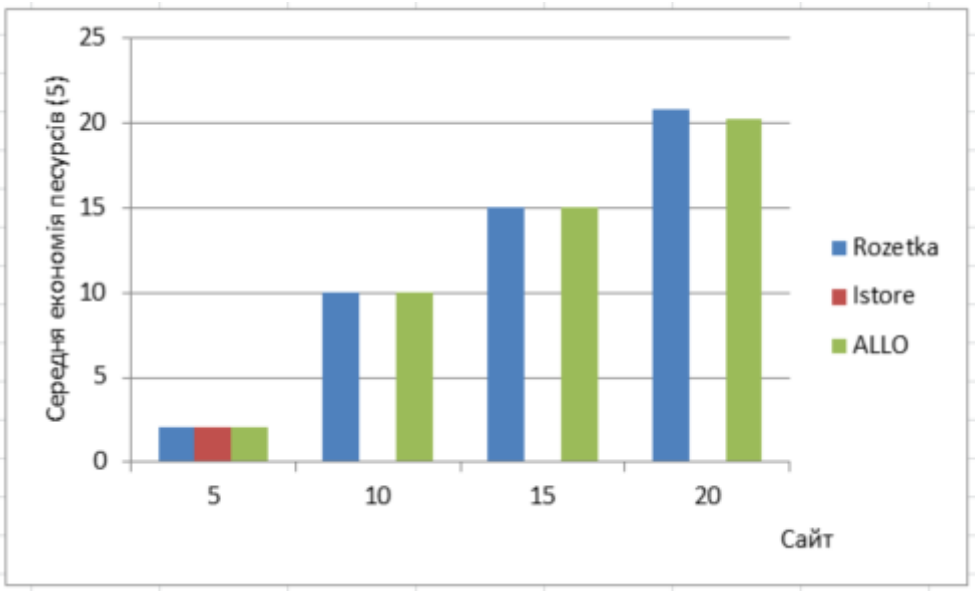


Рисунок 3.3 - Обсяг зекономлених ресурсів у відсотках

Ці дані демонструють, як структура веб-сторінки впливає на ефективність скрепінгу, підкреслюючи необхідність індивідуального підходу до кожного ресурсу при розробці стратегій збору даних.

Порівняння браузерів**.** Також варто порівняти швидкодію браузерів (рис.3.4). Найпопулярнішими для веб-скрепінгу є Google Chrome та Mozilla Firefox [11,12]. Випробування будуть проведені на тих самих сторінках, що і в таблиці 3.4.

Рисунок 3.4 – Порівняння швидкості запитів

Таблиця 3.4 – Випробування веб-скрепінгу у Google Chrome та Mozilla Firefox

Номер запиту	Час відповіді Firefox (мс)	Час відповіді Chrome (мс)
1	535.87	466.50
2	517.99	470.38
3	609.32	394.80
4	496.93	396.87
5	467.20	538.35
6	474.12	376.82
7	520.09	488.05
8	494.71	497.62
9	469.42	530.30

Ця таблиця відображає результати кожного з дев’яти запитів, демонструючи варіації у часі відповіді для обох браузерів (рис.3.5).

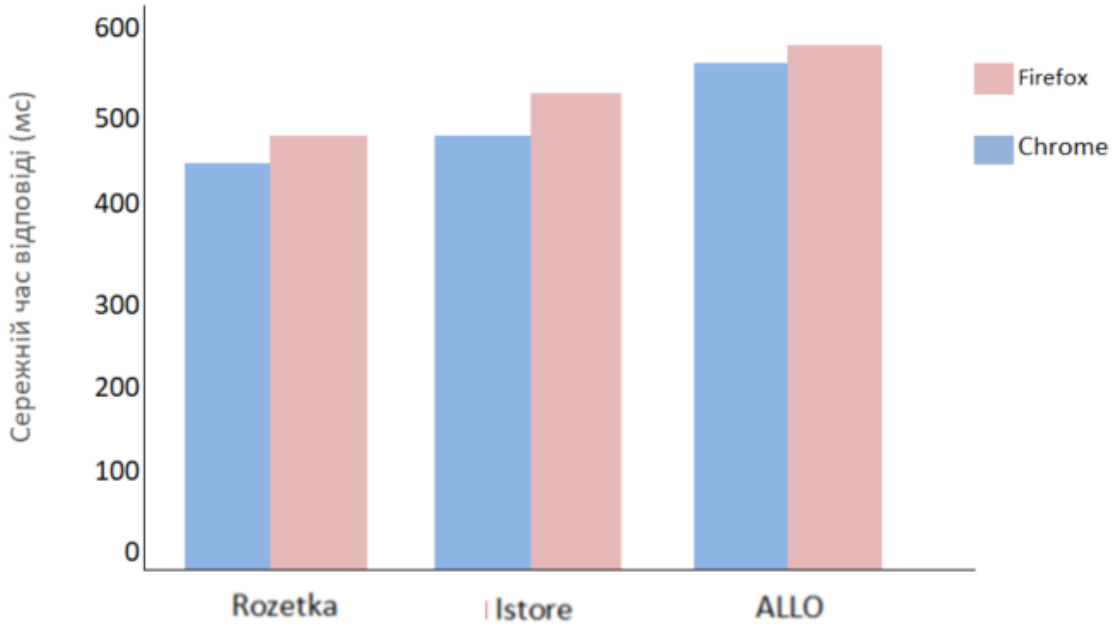


Рисунок 3.5 – Середній час відповіді браузер на запит

З графіка можна зробити такі висновки:

- швидкість відповіді в Chrome в середньому дещо вища, ніж у Firefox;
- обидва браузери демонструють варіативність часу відповіді, що залежить від ваги сторінок та навантаження мережі.

Отже, Google Chrome є кращим вибором не лише через швидкодію, а й через популярність. Розробники більше уваги приділяють тестуванню сайтів у цьому браузері, оскільки різні віджети та JavaScript можуть працювати по-різному у різних браузерах. Використовуючи Chrome, ймовірність зіткнутися з непередбачуваними проблемами мінімальна.

3.4 Висновки до розділу

У третьому розділі підкреслюється важливість розробки гнучких інструментів збору даних, які мають стратегічне значення у веб-аналітиці. Клас BaseScraper виступає фундаментом веб-скрепінгу, забезпечуючи автоматизацію, підвищену ефективність і точність збору інформації.

Архітектура BaseScraper модульна та адаптивна, з можливістю налаштування початкового URL і селекторів, що робить його універсальним рішенням для різних джерел даних. Це важливо для швидкого реагування на зміни в структурах веб-ресурсів.

Використання Selenium та BeautifulSoup дозволяє ефективно обробляти динамічний контент, взаємодіяти зі складними JavaScript-елементами і AJAX-запитами сучасних сайтів. Автоматизація через BaseScraper знижує людські помилки, підвищує продуктивність і дозволяє обходити захист від скрепінгу.

Порівняння швидкодії браузерів Google Chrome та Mozilla Firefox показало перевагу Chrome у часі відповіді, обробці сторінок та виконанні JavaScript, що робить його оптимальним для веб-скрепінгу.

Методика парсингу була вдосконалена шляхом фокусування на основному контенті сторінки (тег), ігноруючи бічні панелі, футери та рекламу. Це значно підвищує ефективність, скорочує час збору та економить ресурси, покращуючи якість і релевантність зібраних даних.

Структуроване збереження даних у форматі JSON забезпечує легку інтеграцію в аналітичні системи завдяки своїй гнучкості, читабельності та сумісності. Збереження часових міток відкриває можливості для аналізу ринкових трендів і прогнозування, що важливо для оптимізації бізнес-процесів.

Вибір ефективних браузерних двигунів, зокрема Google Chrome, є ключовим фактором успіху збору даних і якості веб-аналітики.

Таким чином, третій розділ розкриває як технічні, так і стратегічні аспекти веб-скрепінгу. Універсальність і адаптивність BaseScraper разом із застосуванням

JSON формують міцну основу для систем збору та аналізу даних, що дозволяє компаніям оперативно реагувати на зміни ринку і отримувати конкурентні переваги.

4 ПРОЕКТУВАННЯ СИСТЕМИ

4.1 Розробка інформаційних панелей та звітів для відображення результатів аналізу

4.1.1 Вибір Django Admin для відображення даних

Під час вибору платформи для відображення та керування даними було обрано Django Admin завдяки її функціональності, зручності використання та гнучкості. Це рішення базується на низці переваг, які роблять Django Admin ефективним інструментом для реалізації адміністративної частини веб-додатків.

Переваги Django Admin [13]:

– інтеграція з Django - Django Admin є вбудованим компонентом фреймворку Django, що надає графічний інтерфейс для управління даними без потреби у створенні окремих інтерфейсів для базових CRUD-операцій. Інструмент автоматично формує інтерфейс на основі Django моделей та дозволяє налаштовувати відображення полів, фільтрів, сортування тощо. Це прискорює розробку і дозволяє адаптувати інтерфейс до потреб конкретного проєкту.

– гнучка кастомізація - Django Admin підтримує широкі можливості для налаштування: від редагування шаблонів і стилів до впровадження власних функцій. Можна змінювати HTML, CSS, додавати JavaScript для розширення функціоналу. Інтерфейси форм також можна змінювати, додаючи кастомні поля, валідацію, фільтри або інтегруючи інші компоненти фреймворку. Також підтримується створення власних адміністративних дій, як-от масове оновлення або експорт даних.

– безпека - Django Admin успадковує механізми безпеки Django,

включно з аутентифікацією, авторизацією та захистом від XSS, CSRF, SQL-ін’єкцій. Система дозволів забезпечує детальне керування доступом до моделей та окремих об’єктів. Підтримуються сучасні методи хешування паролів і регулярні оновлення, що гарантує актуальний захист від вразливостей.

– інструменти для роботи з даними - Django Admin містить вбудовані функції пошуку, фільтрації, експорту та імпорту даних, що спрощує повсякденну роботу з великими обсягами інформації.

Порівняння з альтернативами. У порівнянні з phpMyAdmin, Rails Admin або кастомними рішеннями на React/Angular, Django Admin має значну перевагу завдяки повній інтеграції з фреймворком, гнучкості налаштувань і меншій потребі у додатковому коді:

– phpMyAdmin - ефективний для роботи з MySQL, але не має повноцінної інтеграції з веб-додатками та обмежений у кастомізації;

– Rails Admin - подібний за можливостями, але вимагає використання Ruby on Rails;

– JavaScript-фреймворки - надають високий рівень гнучкості, проте потребують більших ресурсів на розробку та підтримку.

У сучасному середовищі управління даними ключову роль відіграє візуалізація та аналітика, і Django Admin дозволяє ефективно реалізувати ці задачі.

Розробка API-сервісу для завантаження даних

– інтеграційний API - створення API-сервісу дозволяє автоматизувати завантаження даних з різних джерел до централізованої бази даних, що є першим етапом у системі візуалізації та аналітики;

– автоматизація процесу - API мінімізує ручне втручання, зменшує ймовірність помилок та забезпечує ефективну передачу великого обсягу інформації.

Інтеграція даних у Django Admin:

– збереження у форматі JSON - дані передаються на сервер у форматі JSON, який є легким, гнучким і зручним для взаємодії між клієнтом і сервером,

особливо при роботі з великими обсягами інформації;

– централізоване управління - Django Admin забезпечує просту візуалізацію, редагування, перегляд і видалення записів, що спрощує адміністративні процеси;

– відстеження змін - збереження дати збору дозволяє контролювати динаміку цін, змін характеристик продуктів та інших ключових параметрів, що забезпечує цінну аналітику;

– інтерактивна візуалізація - можливість створення графіків, діаграм і таблиць на базі зібраних даних покращує сприйняття інформації та сприяє прийняттю обґрунтованих рішень;

– зручний доступ - адмін-панель дозволяє авторизованим користувачам швидко працювати з інформацією - здійснювати пошук, фільтрацію та аналітику;

– надійність та захист - фреймворк Django гарантує безпечну обробку конфіденційної інформації через захищену авторизацію, сесійний менеджмент та регулярні оновлення.

Інтеграція збереження та управління даними через Django Admin оптимізує доступ до інформації, її безпеку й ефективне використання у бізнес-процесах.

4.1.2 Схема бази даних

Для якісного зберігання та обробки зібраних даних важливо створити продуману схему бази. В рамках Django це реалізується через створення моделей (рис.4.1). Наприклад, модель Product слугуватиме для зберігання даних про товари, включаючи ціни, характеристики, дату збору тощо. Це забезпечить логічну структуру даних, їх ефективне зберігання, обробку та подальший аналіз.

Ключові елементи схеми бази даних:

– назва продукту (name) - текстове поле для зберігання назви товару, що є основним ідентифікатором кожної одиниці;

– назва магазину (store_name) - текстове поле, яке вказує джерело магазин, у якому знайдено продукт;

– ціна (price) - поле типу float для фіксації вартості продукту, що є основним показником для цінового аналізу;

– фото (photo) - текстове поле з посиланням на зображення продукту, що полегшує його ідентифікацію у візуальному інтерфейсі;

– дата збору (date) - поле дати, яке фіксує момент збору інформації для відстеження змін у часі.

```
class Product(models.Model):
    name = models.CharField(max_length=256)
    store_name = models.CharField(max_length=256)
    price = models.FloatField()
    photo = models.CharField(max_length=256)
    date = models.DateField()
```

Рисунок 4.1 – Реалізація моделей

Призначення схеми: запропонована структура бази даних дозволяє ефективно управляти даними про товари, здійснювати моніторинг змін на ринку, проводити аналітику динаміки цін і попиту. Вона також забезпечує зручну візуалізацію інформації в адміністративній панелі та є основою для прийняття обґрунтованих рішень на основі даних.

4.2 Розробка API-ендпоінта для отримання даних у форматі JSON

Реалізація-ендпоінта**.** Для забезпечення інтеграції з зовнішніми джерелами даних передбачено створення API-ендпоінта, що приймає інформацію у форматі JSON. Цей формат є легким, структурованим і сумісним з більшістю сучасних технологій [14].

Захист через API-ключ. Доступ до ендпоінта захищено API-ключем, що забезпечує автентифікацію запитів. Тільки авторизовані клієнти матимуть змогу надсилати дані, що гарантує захист системи від несанкціонованого доступу.

Послідовність роботи API:

– прийом запиту - сервер отримує POST-запит із JSON-даними, що містять інформацію про продукти;

– перевірка ключа - система перевіряє валідність API-ключа перед обробкою запиту;

– збереження даних - після успішної перевірки дані обробляються та зберігаються у базі;

– підтвердження - у відповідь клієнт отримує повідомлення про успішне прийняття та збереження даних.

Переваги підходу:

– безпека - захист API-ключем зменшує ризики несанкціонованого доступу;

– керований доступ - можливість контролювати, хто має доступ до надсилання інформації;

– сумісність - JSON забезпечує просту інтеграцію з різними системами та платформами

4.2.1 Кастомізація Django Admin для зручного управління даними

Для ефективної роботи з даними у адміністративному інтерфейсі Django передбачена гнучка кастомізація, яка дозволяє адаптувати вигляд і функціональність панелі до специфіки проекту [15].

Ключові напрями налаштування включають наступні етапи.

Оформлення інтерфейсу: зміна стилю, макету, кольорів і типографіки для покращення юзабіліті та відповідності корпоративному дизайну.Такий рівень налаштування (рис.4.2) забезпечує зручну візуалізацію, оперативне управління даними та покращує взаємодію користувачів із системою.

Оформлення інтерфейсу сторінки продукту: модифікація дизайну сторінки продукту покращує сприйняття інформації та полегшує роботу з даними. Візуальні елементи, такі як відображення зображення товару (рис. 4.3), допомагають користувачеві швидко ідентифікувати продукт та краще розуміти його характеристики.

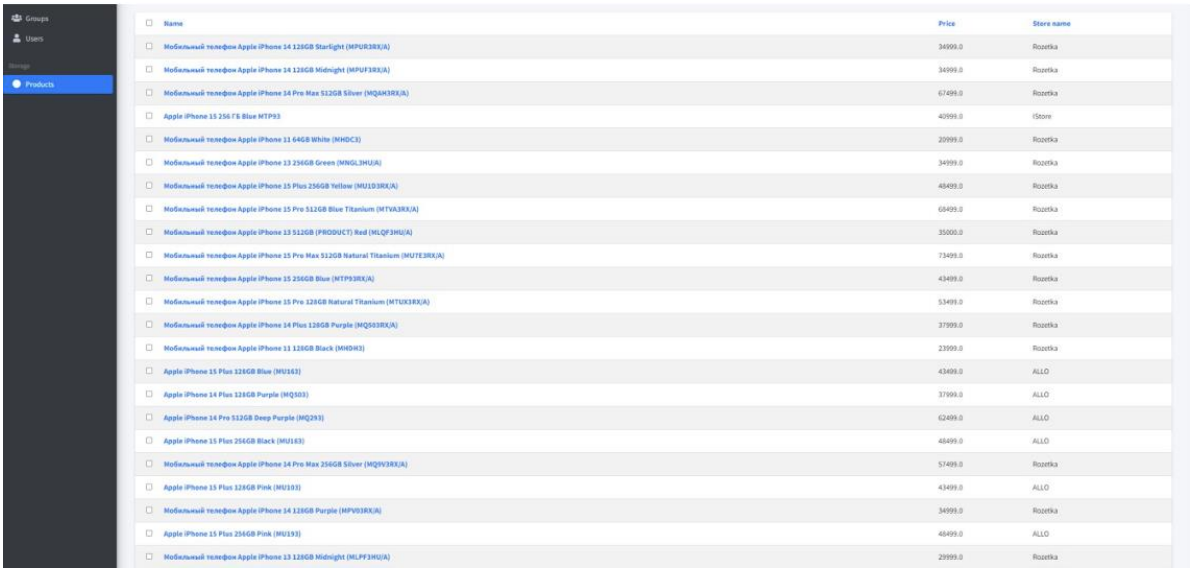


Рисунок 4.2 – Налаштування системи

Розширені функції фільтрації та пошуку**:** впровадження вдосконалених інструментів фільтрації та пошуку спрощує процес доступу до необхідної інформації та підвищує ефективність аналізу. Це дозволяє користувачам гнучко працювати з даними - зокрема, здійснювати вибірку товарів з окремого магазину або порівнювати однакові продукти чи моделі з різних магазинів за допомогою фільтрації за назвою продукту (рис.4.4 - 4.7)..

Рисунок 4.3 – Приклад роботи

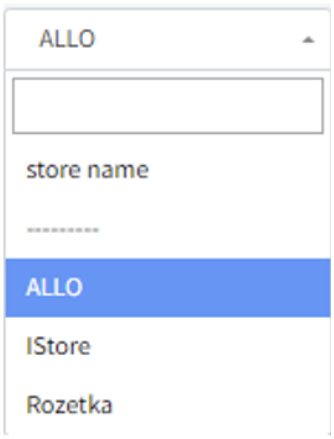


Рисунок 4.4 – Фільтрація товарів за магазином

Рисунок 4.5 – Результат фільтрації товарів за магазином

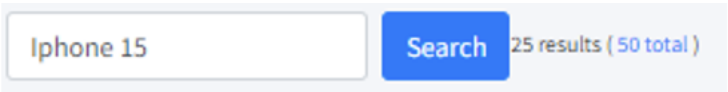


Рисунок 4.6 – Пошук товару за назвою

Рисунок 4.7– Результат пошуку за назвою

Створення користувацьких дій: інтеграція функції експорту даних у форматі XLSX є важливим етапом підвищення ефективності аналітичної роботи, оскільки цей формат зручно використовувати в таких інструментах, як Microsoft Excel та

інші програми для табличного аналізу.

Для оптимізації робочих процесів у Django Admin буде додано спеціальну кнопку експорту (рис. 4.8), яка дозволить швидко та зручно вивантажувати дані безпосередньо з адміністративної панелі. Це значно полегшить підготовку даних для подальшого аналізу.

Реалізація цієї можливості передбачає використання відповідних розширень Django з підтримкою формату XLSX, а також розробку кастомного коду, який відповідатиме за ініціацію та керування процесом експорту. Інтерфейс кнопки буде інтегровано у зручному місці адміністративної панелі, що забезпечить легкий доступ і зрозуміле керування для користувачів.

Рисунок 4.8 – Адмін панель

Кастомізація адміністративної панелі Django є ключовим етапом у формуванні ефективного та безпечного середовища для роботи з даними. Завдяки гнучким можливостям Django можна створити інтерфейс, максимально адаптований до специфіки проєкту та вимог користувачів, що забезпечує зручність і розширену функціональність у процесі керування інформацією [16].

4.3 Використання даних для стратегічного планування та прийняття рішень

У цьому розділі розглядається важливість інтеграції аналітичних інструментів у Django Admin для підвищення ефективності бізнес-процесів. Зокрема, йдеться про реалізацію експорту даних про продукти у формат XLSX із можливістю подальшого аналізу змін цін.

Для цього використовуються спеціалізовані інструменти Django у поєднанні з зовнішніми бібліотеками, такими як `pandas` та `openpyxl`, що дозволяє ефективно обробляти та зберігати інформацію. Ключовим елементом є функціональність відстеження змін цін, яка автоматично визначає їх динаміку. У згенерованому Excel-файлі зміни візуалізуються кольорами: зростання цін виділяється червоним, зниження - зеленим, що полегшує сприйняття та аналіз.

Впровадження такої аналітики в адміністративну панель Django дозволяє не лише покращити управління даними, але й значно підсилює можливості аналізу ринку, сприяючи прийняттю обґрунтованих стратегічних рішень.

У межах проєкту було розроблено клас `DownloadXLSXAdmin`, який розширює стандартний функціонал Django Admin і реалізує повноцінну систему експорту даних у форматі XLSX. Клас використовує методи `queryset_to_json` і `write_xlsx` для обробки інформації.

Метод `queryset_to_json` приймає queryset і формує список JSON-об'єктів, оцінюючи кількість записів для кожного продукту. Якщо продукт має декілька записів, метод визначає зміну ціни у відсотках, що дозволяє виявляти цінові тенденції.

Метод `write_xlsx` отримує список JSON-об’єктів та конвертує його у XLSX-файл за допомогою бібліотек `pandas` та `openpyxl`, формуючи зручну таблицю для подальшого аналізу в табличних редакторах.

Функція `make_response` об’єднує ці етапи, створюючи фінальний файл XLSX та формуючи HTTP-відповідь для його завантаження безпосередньо з

адміністративної панелі.

Клас `DownloadXLSXAdmin` містить кнопку `export_xlsx`, яка активує процес експорту. Після натискання кнопки система запускає функцію `make_response` з поточним queryset, що ініціює збір, аналіз та збереження даних. Така інтеграція значно спрощує процес роботи з даними, перетворюючи Django Admin на потужний інструмент для адміністрування та аналітики.

На представленому скріншоті зображено фрагмент Excel-таблиці з проаналізованими даними, що містять інформацію про різні моделі мобільних телефонів Apple iPhone із зазначенням обсягу пам'яті, кольору та інших характеристик (рис.4.9). У таблиці наведені назва товару, магазин, у якому його продають, дата отримання цінової пропозиції та її вартість.

	A	B	C	D	E
1		Name	Store	date	Price
2	0	Apple iPhone 15 128GB Black (MTP03)	ALLO	2023-12-27	38499
3	1	Apple iPhone 15 128GB Black (MTP03)	ALLO	2024-01-01	35000.0 (-10.0 %)
4	2	Apple iPhone 14 128GB Starlight (MPUR3)	ALLO	2023-12-27	34999
5	3	Apple iPhone 14 Plus 128GB Starlight (MQ4Y3)	ALLO	2023-12-27	37999
6	4	Apple iPhone 15 Plus 128GB Black (MU0Y3)	ALLO	2023-12-27	43499
7	5	Apple iPhone 14 Pro Max 512GB Space Black (MQAF3)	ALLO	2023-12-27	67499
8	6	Apple iPhone 14 Plus 128GB Midnight (MQ4X3)	ALLO	2023-12-27	37999
9	7	Apple iPhone 11 64GB White (MHDC3) Slim Box	ALLO	2023-12-27	20999
10	8	Apple iPhone 11 64GB White (MHDC3) Slim Box	ALLO	2024-01-01	20000.0 (-5.0 %)
11	9	Мобильный телефон Apple iPhone 13 128GB Midnight (MLPF3HU/A)	Rozetka	2023-12-27	29999
12	10	Apple iPhone 15 Plus 256GB Pink (MU193)	ALLO	2023-12-27	48499
13	11	Apple iPhone 15 Plus 128GB Pink (MU103)	ALLO	2023-12-27	43499
14	12	Apple iPhone 15 Plus 256GB Black (MU183)	ALLO	2023-12-27	48499
15	13	Apple iPhone 14 Pro 512GB Deep Purple (MQ293)	ALLO	2023-12-27	62499
16	14	Apple iPhone 14 Plus 128GB Purple (MQ503)	ALLO	2023-12-27	37999
17	15	Apple iPhone 15 Plus 128GB Blue (MU163)	ALLO	2023-12-27	43499
18	16	Apple iPhone 15 Plus 128GB Blue (MU163)	ALLO	2024-01-01	43499.0 (0.0 %)
19	17	Мобильный телефон Apple iPhone 11 128GB Black (MHDH3)	Rozetka	2023-12-27	23999
20	18	Мобильный телефон Apple iPhone 15 Pro 512GB Blue Titanium (MTVA3RX/A)	Rozetka	2023-12-27	68499
21	19	Мобильный телефон Apple iPhone 15 Plus 256GB Yellow (MU1D3RX/A)	Rozetka	2023-12-27	48499
22	20	Мобильный телефон Apple iPhone 13 256GB Green (MNGL3HU/A)	Rozetka	2023-12-27	34999
23	21	Мобильный телефон Apple iPhone 13 256GB Green (MNGL3HU/A)	Rozetka	2024-01-01	32500.0 (-7.69 %)
24	22	Мобильный телефон Apple iPhone 11 64GB White (MHDC3)	Rozetka	2023-12-27	20999
25	23	Apple iPhone 15 256 ГБ Blue MTP93	IStore	2023-12-27	40999
26	24	Apple iPhone 15 256 ГБ Blue MTP93	IStore	2024-01-01	43000.0 (4.65 %)
27	25	Мобильный телефон Apple iPhone 14 Pro Max 512GB Silver (MQAH3RX/A)	Rozetka	2023-12-27	67499
28	26	Мобильный телефон Apple iPhone 14 128GB Midnight (MPUF3RX/A)	Rozetka	2023-12-27	34999
29	27	Мобильный телефон Apple iPhone 14 128GB Starlight (MPUR3RX/A)	Rozetka	2023-12-27	34999
30	28	Мобильный телефон Apple iPhone 14 128GB Starlight (MPUR3RX/A)	Rozetka	2024-01-01	35999.0 (2.78 %)
31	29	Мобильный телефон Apple iPhone 11 128GB Black (MHDH3)	Rozetka	2024-01-01	20000
32	30	Мобильный телефон Apple iPhone 14 Pro Max 256GB Silver (MQ9V3RX/A)	Rozetka	2024-01-01	60000

Рисунок 4.9 – Результуючий файл

Кожен рядок відображає окремий товар або конкретну торгову пропозицію. Особливістю таблиці є наявність колонки з відсотковими змінами цін, що дає змогу бачити динаміку вартості. Ці зміни зазначені у форматі відсотка поряд із

актуальною ціною: знак плюс означає зростання, знак мінус - зниження. Для кращої візуалізації змін використано кольорове підсвічування: червоний - підвищення ціни, зелений - зниження. Наприклад, у 20-му рядку модель "Apple iPhone 11 128GB Black" має вартість 34 999 із позначкою "(0.0 %)", що свідчить про відсутність змін у ціні порівняно з попередньою пропозицією.

Такі дані можна використовувати для вивчення ринкових тенденцій, аналізу попиту на конкретні моделі або для стратегічного управління товарними запасами в роздрібній торгівлі.

Зміни в цінах можуть бути наслідком маркетингових кампаній, сезонного попиту чи загальних економічних коливань.

При цьому продукція Apple у таблиці представлена як приклад використання класу `BaseScraper`. Щоб почати збір даних щодо інших категорій товарів у тому самому магазині, достатньо змінити параметр `start_url` у класі `RozetkaScraper`, наприклад, на `https://bt.rozetka.com.ua/refrigerators/c80125/`, і система автоматично почне збір інформації вже не про смартфони, а про холодильники чи будь-яку іншу категорію (рис. 4.10).

```
class RozetkaScraper(BaseScraper):

    def __init__(self, start_url, selectors):
        super().__init__(start_url, selectors)
        self.start_url = 'https://bt.rozetka.com.ua/refrigerators/c80125/'
        self.selectors = ROZETKA_SELECTORS
        self.options = ["--window-size=1920,1080", "--disable-proxy-certificate-handler", "--ignore-certificate-errors"]
        self.store_name = 'Rozetka'
```

Рисунок 4.10 – Система автоматизації

4.4 Висновки до розділу

У розділі проаналізовано стратегічне значення використання даних у процесі прийняття рішень та акцентовано увагу на важливості створення

інформаційних панелей і звітів у Django Admin для ефективного представлення результатів аналітики. Django Admin обрано як основну платформу завдяки її тісній інтеграції з Django фреймворком, можливостям гнучкої кастомізації, наявності вбудованих інструментів для обробки даних і високому рівню безпеки.

Основною перевагою Django Admin є її здатність до глибокої інтеграції з моделями проєкту, що дозволяє розробникам швидко налаштовувати інтерфейс адміністрування відповідно до конкретних потреб. Це середовище значно спрощує роботу з даними й дає змогу реалізовувати складну аналітичну функціональність, включаючи моніторинг змін цін і експорт звітності для подальшого аналізу. Особливу роль відіграє гнучкість інтерфейсу Django Admin, що забезпечує адаптацію під специфіку бізнес-завдань. Від зміни вигляду інтерфейсу до інтеграції функцій глибокої обробки даних - Django Admin надає потужний інструментарій для створення індивідуального та продуктивного середовища роботи.

Запровадження функціональності експорту в XLSX формат за допомогою спеціальної кнопки розширює можливості аналітики, дозволяючи легко обробляти великі обсяги інформації та візуалізувати зміни у зручній формі. Клас `DownloadXLSXAdmin` реалізує повний цикл - від збору даних до формування звіту, забезпечуючи швидкий експорт результатів одним натисканням.

У розділі розкривається не лише технічна ефективність Django Admin, а й його стратегічна роль у бізнес-процесах, підкреслюючи значення аналітики для ухвалення управлінських рішень (додаток А). Технічні інструменти в цьому контексті виступають як основа для досягнення бізнес-цілей: аналіз ринкових тенденцій, відстеження цінових змін, адаптація до коливань попиту та оптимізація товарного асортименту що сприяє підвищенню ефективності та конкурентоспроможності.

ВИСНОВКИ

Робота, присвячена дослідженню веб-скрапінгу, становить вагомий внесок у розуміння його потенціалу та практичного застосування в сучасному цифровому середовищі. У ході роботи було розглянуто широкий спектр аспектів цієї технології - від технічної реалізації до правових та етичних питань.

Перш за все, веб-скрапінг виявився ефективним інструментом для компаній і дослідників, що дозволяє автоматизовано збирати великі обсяги інформації з різноманітних онлайн-ресурсів. Це особливо актуально в умовах високої інтенсивності обігу інформації в сучасному бізнес-середовищі.

Результати підтверджують широке застосування веб-скрапінгу в різних сферах - від комерційного аналізу ринку та поведінки споживачів до збору даних для наукових і академічних досліджень. Цей підхід активно використовується для виявлення ринкових тенденцій, моніторингу конкурентного середовища та побудови ефективних маркетингових стратегій.

У рамках цієї роботи досягнуто таких ключових результатів:

– аналіз швидкодії інструментів веб-скрапінгу;

– створено гнучкий та адаптивний інструмент BaseScraper, який дає змогу легко змінювати параметри для збору даних із різних Інтернет-магазинів, використовуючи налаштування CSS-селекторів і URL-адрес;

– реалізовано рішення на базі фреймворку Django, яке забезпечує надійне зберігання зібраних даних і надає засоби для їх візуалізації в адміністративному інтерфейсі, полегшуючи подальший аналіз;

– можливість експорту даних у формат XLSX із автоматичним підрахунком змін цін і трендів, що забезпечує зручність подальшої обробки даних за допомогою табличних редакторів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Huang, C.-J. The Synergy of Automated Pipelines with Prompt Engineering and Generative AI in Web Crawling. URL: https://arxiv.org/abs/2502.15691 (Last accessed: 15.04.2025).

2. Ališauskas, B. Web Scraping With Scrapy. URL: https://scrapfly.io/blog/web-scraping-with-scrapy/ (Last accessed: 25.05.2025

3. Kumari, J. A Comprehensive Guide to Web Scraping Using Selenium URL: https://www.analyticsvidhya.com/blog/2024/05/a-comprehensive-guide-to-web-scraping-using-selenium/ (Last accessed: 27.04.2025).

4. Wilson, R. Selenium vs BeautifulSoup in 2025: A Complete Developer's Guide to Web Scraping Tools URL: https://rebrowser.net/blog/selenium-vs-beautifulsoup-a-complete-developers-guide-to-web-scraping-tools (Last accessed: 17.04.2025)

5. Lin, J. Anywhere: A Web Crawler Automation Management Interface URL: https://arxiv.org/abs/2407.00025 (Last accessed: 03.05.2025).

6. García, B. Selenium-Jupiter: A JUnit 5 Extension for Selenium WebDriver URL: https://arxiv.org/abs/2402.01480. (Last accessed: 18.04.2025).

7. ScrapeHero. Top Open Source JavaScript Web Scraping Tools and Frameworks URL: https://www.scrapehero.com/open-source-javascript-web-scraping-tools-and-frameworks/. (Last accessed:15.05.2025).

8. Datta, S. DeepObfusCode: Source Code Obfuscation Through Sequence-to-Sequence Networks URL: https://arxiv.org/abs/1909.01837. (Last accessed: 02.05.2025).

9. Lin, J. Anywhere: A Web Crawler Automation Management Interface URL: https://arxiv.org/abs/2407.00025. (Last accessed: 30.04.2025).

10. Kumari, J. A Comprehensive Guide to Web Scraping Using Selenium URL: https://www.analyticsvidhya.com/blog/2024/05/a-comprehensive-guide-to-web-scraping-using-selenium/ (Last accessed:26.04.2025).

11. Greenberg, A. Google is finally fixing one of Chrome's biggest privacy flaws [URL: https://www.wired.com/story/google-chrome-extensions-privacy-fix. (Last accessed: 11.05.2025).

12. Newman, L. In a swipe at Chrome, Firefox now blocks ad trackers by default URL: https://www.wired.com/story/firefox-browser-cookie-blocking-default. (Last accessed: 18.05.2025).

13. Danduprolu, M. Getting Started with Django 2024: Mastering the Django Admin Interface. URL: https://medium.com/@mathur.danduprolu/django-getting-started-with-django-2024-mastering-the-django-admin-interface-part-6-16-e522bfef5a82. (Last accessed: 30.04.2025).

14. Huang, C.-J. The Synergy of Automated Pipelines with Prompt Engineering and Generative AI in Web Crawling URL: https://arxiv.org/abs/2502.15691. (Last accessed: 22.05.2025).

15. Promise, A. Mastering Django Admin Customization: Best Practices and Techniques URL: https://medium.com/@abasifreke/mastering-django-admin-customization-best-practices-and-techniques-66a6b31ad4f8. (Last accessed: 05.05.2025).

16. Strategy, M. Customizing Django Admin: Tips and Tricks for a Tailored Admin Interface URL: https://medium.com/@sampinkman41/customizing-django-admin-tips-and-tricks-for-a-tailored-admin-interface-7ad6b5d14817 (Last accessed: 18.05.2025).

ДОДАТОК А
ПРОГРАМНІ КОДИ

Лістинг А.1 – Файл base_scraper.py

```
import re
import requests
from selenium import webdriver
from selenium.webdriver import ChromeOptions
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.firefox.service import Service
from selenium.webdriver.common.by import By
from urllib.parse import urlparse
from bs4 import BeautifulSoup
HOST = "127.0.0.1"
PORT = "8000"
class BaseScraper:
def __init__(self, start_url, selectors,):
self.start_url = start_url
self.selectors = selectors
self.options = None
self.chrome_prefs = {"profile.default_content_settings":
{"javascript": 2}}
self.store_name = None
def get_driver(self):
options = ChromeOptions()
for option in self.options:
options.add_argument(option)
options.experimental_options["prefs"] = self.chrome_prefs
driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install()),
options=options)
return driver
def get_page(self, link):
page = self.get_driver()
page.get(link)
return page
def get_product_pages(self, page):
pages = page.find_elements(By.CSS_SELECTOR,
self.selectors.get('product_wrappers'))
res = []
for page in pages:
href = BeautifulSoup(page.get_attribute('innerHTML'),
'html.parser').find('a')['href']
if not href.startswith('http'):
href = "https://" + urlparse(self.start_url).netloc + href
res.append(href)
return res def scrape_items_data(self, links): items_list = []
for link in links:
item_data = {'Store_name': self.store_name}
```

```
page = self.get_page(link)
body = page.find_element(By.TAG_NAME,'body')
soap = BeautifulSoup(body.get_attribute('innerHTML'), 'html.parser')
item_data['Name'] =
soap.select(self.selectors.get('name_selector'))[0].text
item_data['Price'] = re.sub(r'[^\d]', '',
soap.select(self.selectors.get('price_selector'))[0].text)
item_data['Photo'] =
soap.select(self.selectors.get('photo_selector'))[0].get('src')
items_list.append(item_data)
return items_list
def scrape(self):
product_links =
self.get_product_pages(self.get_page(self.start_url))
products_data = self.scrape_items_data(product_links)
self.api_call(products_data)
return products_data
def api_call(self, data):
requests.post(f'http://{HOST}:{PORT}/api/upload/', json={"data":
data})
```

Лістинг А.2 – Файл rozetka_scraper.py

```
from base_scraper import BaseScraper
from selectors import ROZETKA_SELECTORS
class RozetkaScraper(BaseScraper):
def __init__(self, start_url, selectors):
super().__init__(start_url, selectors)
self.start_url = start_url
self.selectors = ROZETKA_SELECTORS
self.options = ["--window-size=1920,1080", "--disable-proxy-
certificatehandler", "--ignore-certificate-errors"]
self.store_name = 'Rozetka'
if __name__ == '__main__':
rozetka_scraper =
RozetkaScraper('https://rozetka.com.ua/mobilephones/c80003/producer=
apple/#search_text=iphone',
ROZETKA_SELECTORS)
rozetka_scraper.scrape()
```

Лістинг А.3 – Файл admin.py

```
from django.contrib import admin
from django.utils.html import format_html
from.models import Product
from.forms import ProductForm
from.market_analysis import make_response
from admin_extra_buttons.api import ExtraButtonsMixin, button
class DownloadXLSXAdmin(ExtraButtonsMixin, admin.ModelAdmin):
@button(html_attrs={'style': 'background-
color:#28A745;color:white;borderradius:12px;padding:12px;'})
```

```
def export_xlsx(self, request):
return make_response(self.get_queryset(request))
class ProductAdmin(DownloadXLSXAdmin):
list_display = ('name', 'price', 'store_name')
fields = ['name', 'price', 'store_name', 'date', 'image_preview']
readonly_fields = ['name', 'price', 'store_name', 'date',
'image_preview']
list_filter = ['store_name', ]
search_fields = ['name', ]
form = ProductForm
def image_preview(self, obj):
return format_html(f'<img src="{obj.photo}" width="180" height="350"
/>')
admin.site.register(Product, ProductAdmin)
```

Лістинг А.4 – Файл market_analysis.py

```
import pandas
from django.http import HttpResponse
from datetime import datetime
from io import BytesIO
from .models import Product
def queryset_to_json(queryset):
res_json_list = []
already_analized = []
for item in queryset:
if item in already_analized: continue
name = item.name
store = item.store_name
all_item_records = Product.objects.filter(name=name,
store_name=store)
if len(all_item_records) == 1:
record_to_write = all_item_records[0]
res_json_list.append({"Name": record_to_write.name, "Store":
record_to_write.store_name,
"date": record_to_write.date.strftime('%Y-%m-
%d'), "Price": record_to_write.price})
already_analized.append(record_to_write)
else:
all_item_records.order_by('date')
res_json_list.append({"Name": all_item_records[0].name, "Store":
all_item_records[0].store_name,
"date": all_item_records[0].date.strftime('%Y-
%m-%d'), "Price": all_item_records[0].price})
already_analized.append(all_item_records[0])
for record_to_write in all_item_records[1::]:
prev_i = 0
price_change = f'{record_to_write.price}
({round(((record_to_write.price - all_item_records[prev_i].price) /
record_to_write.price) * 100, 2)} %)'
res_json_list.append({"Name": record_to_write.name, "Store":
record_to_write.store_name,
```

```
"date": record_to_write.date.strftime('%Y-
%m-%d'), "Price": price_change})
already_analized.append(record_to_write)
return res_json_list
def write_xlsx(json_list):
b = BytesIO()
writer = pandas.ExcelWriter(b, engine='openpyxl')
frames = pandas.DataFrame(json_list)
frames.to_excel(writer)
writer._save()
b.seek(0)
return b.getvalue()
def make_response(queryset):
json_list = queryset_to_json(queryset)
filename = f"report_{datetime.today().strftime('%Y-%m-%d')}.xlsx"
response = HttpResponse(write_xlsx(json_list),
headers={'Content-Disposition': f'attachment;'f'
filename="{filename}"'})
response['Content-Disposition'] ='attachment; filename=%s' %
filename
return response
```

Національний університет «Запорізька політехніка»
Кафедра «Комп'ютерні системи та мережі»

Дипломний проєкт

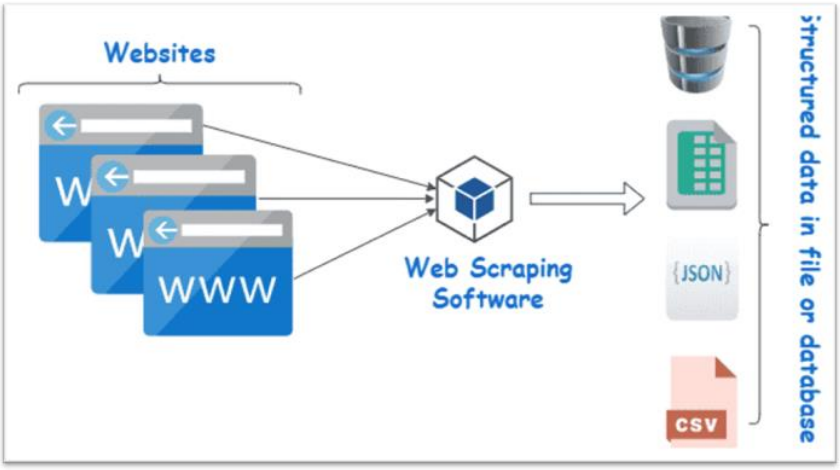
«РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ ТА АНАЛІЗУ ДАНИХ З ВЕБ-САЙТІВ»

Виконав: ст. групи КНТ – 521 Б.С. КОБА
Керівник М.Б. ІЛЬЯШЕНКО

Мета роботи - дослідження проблематики збору та обробки веб-даних для потреб бізнесу, розробці й впровадженні автоматизованої системи, що вирішує ці завдання

Робота над проєктом передбачає досягнення низки конкретних завдань:
- розробка програмного інструменту для веб-скрапінгу;
- аналіз особливостей різних типів веб-джерел ;
- розробка алгоритмів обробки, аналізу та візуалізації даних;
- дослідження правових і етичних аспектів.

Веб-скрапінг - процес автоматизованого витягування інформації з веб-ресурсів.

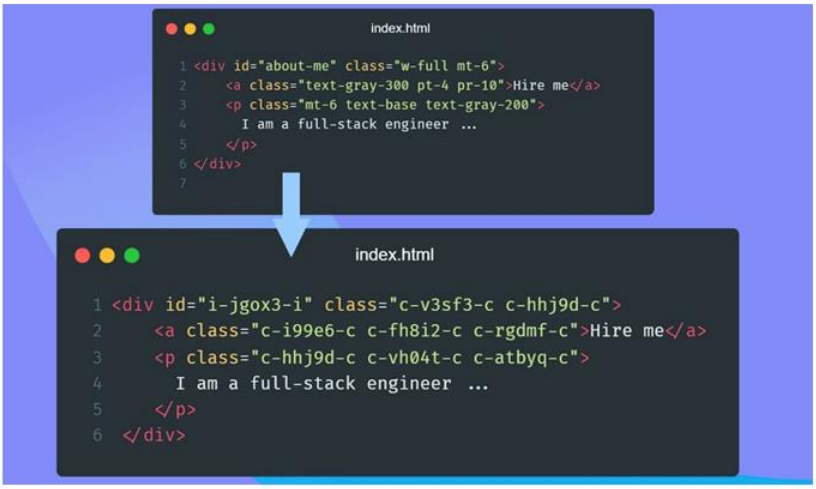


Базова структура системи

Порівняння програмного забезпечення

Framework	Selenium	Scrapy
Конфіденційність	Тестування браузера	API веб-скрапінгу
Підтримка JavaScript	Дуже гарно підтримує	Багато часу для перевірки і розробки для стимулювання запросів
Типи даних	Добре для невеликих даних	Добре працює з великим даними
Швидкість	швидкий	Дуже швидкий
Вимоги до оновлення	Не досить вимогливий	проста розробка проміжного програмного забезпечення або додавання функцій користувача
Складність	Досить легко для новачків	Досить важка для новачків

Приклад захисту даних

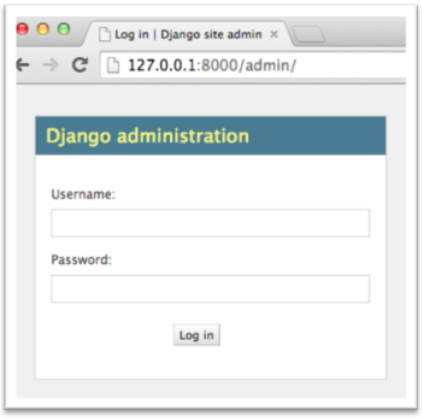


Методи класу BaseScraper

Метод	Опис функції	Можливості
`__init__(self, config)`	Ініціалізує об'єкт класу з конфігурацією скрепінгу	Приймає параметри для налаштування (URL, таймаути, тощо)
`start_requests(self)`	Запускає процес надсилання початкових HTTP-запитів	Генерує початкові URL для скрепінгу
`fetch_page(self, url)`	Завантажує HTML-сторінку за вказаною URL	Використовує Selenium або requests для отримання сторінки
`parse(self, html)`	Аналізує HTML-код та витягує потрібні дані	Використовує BeautifulSoup для парсингу
`extract_data(self, parsed_html)`	Витягує конкретні елементи даних із розібраного HTML	Повертає структуровані дані (наприклад, словник)
`save_data(self, data)`	Зберігає зібрані дані у потрібному форматі	Підтримка збереження у файл, базу даних або API
`handle_pagination(self, html)`	Обробляє пагінацію для переходу між сторінками	Генерує URL наступних сторінок для подальшого скрепінгу
`run(self)`	Запускає повний цикл скрепінгу від отримання до збереження даних	Координує роботу всіх інших методів
`set_headers(self, headers)`	Налаштовує HTTP-заголовки для запитів	Дозволяє імітувати різні браузери або користувацькі агенти
`wait_for_element(self, selector)`	Очікує завантаження певного елемента на сторінці (для динамічних сайтів)	Використовує Selenium для роботи з динамічним контентом

Випробування веб-скрепінгу у Google Chrome та Mozilla Firefox

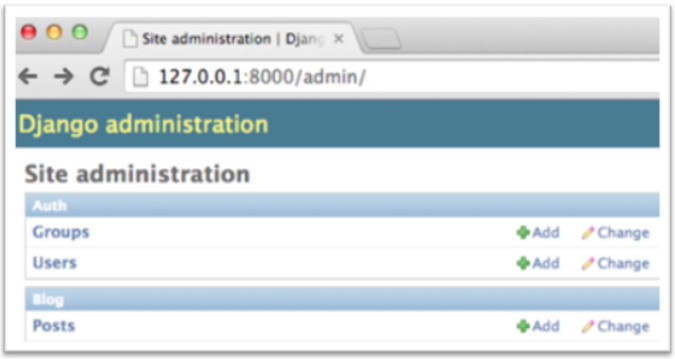
Номер запиту	Час відповіді Firefox (мс)	Час відповіді Chrome (мс)
1	535.87	466.50
2	517.99	470.38
3	609.32	394.80
4	496.93	396.87
5	467.20	538.35
6	474.12	376.82
7	520.09	488.05
8	494.71	497.62
9	469.42	530.30



проста візуалізацію, редагування, перегляд і видалення записів

зручний доступ

збереження дати збору



Django адміністратор

Дані експортовані у Excel-таблицю з проаналізованими даними

	A	B	C	D	E
1		Name	Store	date	Price
2	0	Apple iPhone 15 128GB Black (MTP03)	ALLO	2023-12-27	38499
3	1	Apple iPhone 15 128GB Black (MTP03)	ALLO	2024-01-01	35000.0 (-10.0 %)
4	2	Apple iPhone 14 128GB Starlight (MPUR3)	ALLO	2023-12-27	34999
5	3	Apple iPhone 14 Plus 128GB Starlight (MQ4Y3)	ALLO	2023-12-27	37999
6	4	Apple iPhone 15 Plus 128GB Black (MU0Y3)	ALLO	2023-12-27	43499
7	5	Apple iPhone 14 Pro Max 512GB Space Black (MQAF3)	ALLO	2023-12-27	67499
8	6	Apple iPhone 14 Plus 128GB Midnight (MQ4X3)	ALLO	2023-12-27	37999
9	7	Apple iPhone 11 64GB White (MHDC3) Slim Box	ALLO	2023-12-27	20999
10	8	Apple iPhone 11 64GB White (MHDC3) Slim Box	ALLO	2024-01-01	20000.0 (-5.0 %)
11	9	Мобильный телефон Apple iPhone 13 128GB Midnight (MLPF3HU/A)	Rozetka	2023-12-27	29999
12	10	Apple iPhone 15 Plus 256GB Pink (MU193)	ALLO	2023-12-27	48499
13	11	Apple iPhone 15 Plus 128GB Pink (MU103)	ALLO	2023-12-27	43499
14	12	Apple iPhone 15 Plus 256GB Black (MU183)	ALLO	2023-12-27	48499
15	13	Apple iPhone 14 Pro 512GB Deep Purple (MQ293)	ALLO	2023-12-27	62499
16	14	Apple iPhone 14 Plus 128GB Purple (MQ503)	ALLO	2023-12-27	37999
17	15	Apple iPhone 15 Plus 128GB Blue (MU163)	ALLO	2023-12-27	43499
18	16	Apple iPhone 15 Plus 128GB Blue (MU163)	ALLO	2024-01-01	43499.0 (0.0 %)
19	17	Мобильный телефон Apple iPhone 11 128GB Black (MHDH3)	Rozetka	2023-12-27	23999
20	18	Мобильный телефон Apple iPhone 15 Pro 512GB Blue Titanium (MTVA3RX/A)	Rozetka	2023-12-27	68499
21	19	Мобильный телефон Apple iPhone 15 Plus 256GB Yellow (MU1D3RX/A)	Rozetka	2023-12-27	48499
22	20	Мобильный телефон Apple iPhone 13 256GB Green (MNGL3HU/A)	Rozetka	2023-12-27	34999
23	21	Мобильный телефон Apple iPhone 13 256GB Green (MNGL3HU/A)	Rozetka	2024-01-01	32500.0 (-7.69 %)
24	22	Мобильный телефон Apple iPhone 11 64GB White (MHDC3)	Rozetka	2023-12-27	20999
25	23	Apple iPhone 15 256 ГБ Blue MTP93	IStore	2023-12-27	40999
26	24	Apple iPhone 15 256 ГБ Blue MTP93	IStore	2024-01-01	43000.0 (4.65 %)
27	25	Мобильный телефон Apple iPhone 14 Pro Max 512GB Silver (MQAH3RX/A)	Rozetka	2023-12-27	67499
28	26	Мобильный телефон Apple iPhone 14 128GB Midnight (MPUF3RX/A)	Rozetka	2023-12-27	34999
29	27	Мобильный телефон Apple iPhone 14 128GB Starlight (MPUR3RX/A)	Rozetka	2023-12-27	34999
30	28	Мобильный телефон Apple iPhone 14 128GB Starlight (MPUR3RX/A)	Rozetka	2024-01-01	35999.0 (2.78 %)
31	29	Мобильный телефон Apple iPhone 11 128GB Black (MHDH3)	Rozetka	2024-01-01	20000
32	30	Мобильный телефон Apple iPhone 14 Pro Max 256GB Silver (MQ9V3RX/A)	Rozetka	2024-01-01	60000

ВИСНОВКИ

У рамках цієї роботи було досягнуто такі результати:

- проаналізовано швидкодію інструментів веб-скрапінгу;
- створено гнучкий та адаптивний інструмент BaseScraper, який дає змогу легко змінювати параметри для збору даних із різних Інтернет-магазинів, використовуючи налаштування CSS-селекторів і URL-адрес;
- реалізовано рішення на базі фреймворку Django, яке забезпечує надійне зберігання зібраних даних і надає засоби для їх візуалізації в адміністративному інтерфейсі, полегшуючи подальший аналіз;
- можливість експорту даних у формат XLSX із автоматичним підрахунком змін цін і трендів, що забезпечує зручність подальшої обробки даних за допомогою табличних редакторів.