ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЧНОГО ЗБИРАННЯ ТА СИНТАКСИЧНОГО АНАЛІЗУ ДАНИХ З САЙТІВ МОВОЮ PYTHON»**

Виконав: студент 4 курсу, групи ПД–44 спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)
Дорошин Н.А.
(прізвище та ініціали)
Керівник Садовенко В.С.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Київ –2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О.В.
“____” _____2023 року

ЗАВДАННЯ
НА БАКАЛАВРАТСЬКУ РОБОТУ СТУДЕНТА

ДОРОШИНУ НАЗАРУ АНДРІЄВИЧУ
(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для автоматичного збирання та синтаксичного аналізу даних з сайтів мовою Python»

   Керівник роботи: Садовенко Володимир Сергійович, доцент, К.ф.-м..н.
   (прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

   Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26.
2. Строк подання студентом роботи « 01 » червня 2023 року
3. Вхідні дані до роботи

   3.1. Парсинг та аналіз інформації
   3.2 Науково-технічна література з питань, пов’язаних з програмним забезпеченням щодо автоматичного збирання та синтаксичного аналізу даних з сайтів

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
   4.1 Системи розпізнавання та вилучення текстової інформації з сайту.
   4.2 Вимоги та оцінка якості системи.

4.3 Опис проектування системи.
4.4 Опис використаних технологій.

Перелік демонстраційного матеріалу (назва основних слайдів)

1. Мета, об’єкт та предмет дослідження
2. Задачі дипломної роботи
3. Аналіз аналогів
4. Вимоги до додатку
5. Програмні засоби реалізації
6. Діаграма діяльності
7. Екранні форми
8. Апробація результатів дослідження
9. Висновки

Дата видачі завдання « 25 » лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
	Підбір науково-технічної літератури	25.02-06.03	Виконано
	Вимоги до системи	07.03-14.03	Виконано
	Аналіз літератури	15.03-30.03	Виконано
	Збір та обробка даних	01.04-21.04	Виконано
	Концепція та архітектура програмного забезпечення	22.04-02.05	Виконано
	Вступ, висновки, реферат	03.05-12.05	Виконано
	Розробка обов‘язкових демонстраційних матеріалів	13.05-20.05	Виконано
	Попередній захист роботи	25.05	Виконано
	Здача роботи	01.06.2023	

Студент _____ Дорошин Н.А ___
(підпис)(прізвище та ініціали)
_____
Керівник роботи _____ Садовенко В.С. _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Постановка задачі

У контексті зростаючої кількості даних, доступних в Інтернеті, автоматичне збирання та аналіз цих даних стає все більш важливим завданням для різних сфер діяльності, включаючи науку, бізнес та маркетинг. У цьому контексті, метою даної роботи є розробка програмного забезпечення, яке надасть можливість ефективно та автоматично збирати дані з різних веб-сайтів та виконувати їх синтаксичний аналіз з використанням мови програмування Python.

Мета дослідження

Мета дослідження полягає у розробці функціонального та ефективного програмного забезпечення, яке допоможе автоматизувати процес збирання та аналізу даних з веб-сайтів з використанням мови програмування Python.

Результати дослідження

В результаті було досліджено процес розробки та створення програмного забезпечення для збирання та синтаксичного аналізу даних з сайтів. Також в ході розробки було реалізовано наступні функції: автоматичне збирання даних з різних веб-сайтів, забезпечує обробку різних типів даних, таких як текст, зображення, таблиці тощо, має можливість виявлення та обробки помилок аналізу, таких як синтаксичні помилки або неправильний формат даних.

Висновки та перспективи

Розроблене програмне забезпечення для автоматичного збирання та синтаксичного аналізу даних з веб-сайтів мовою Python виявилося ефективним і корисним інструментом. Воно забезпечує автоматизацію процесу збирання даних з веб-сайтів та надає можливість аналізувати зібрані дані для отримання корисної інформації.

Перспективи розробленого програмного забезпечення для автоматичного збирання та синтаксичного аналізу даних з веб-сайтів мовою Python є досить широкими : дослідницьке застосування, моніторинг соціальних мереж, моніторинг веб-магазинів.

ЗМІСТ

ВСТУП

В даний час розвиток інформаційних технологій має великий вплив на всі сфери людської діяльності, так чи інакше пов'язані з накопиченням і обробкою інформації. Інформаційні технології (IT) – це процеси, які використовують набір засобів і методів збору, обробки та передачі даних (первинної інформації) для отримання нової якісної інформації про стан об’єкта, процесу чи явища (інформаційного продукту). Інформаційна технологія - це процес, що складається з чітко регламентованих правил виконання операцій, дій, етапів різного ступеня складності над даними, що зберігаються в комп'ютерах.

Найпоширенішим джерелом інформації в сучасному світі є Інтернет - всесвітня система взаємопов'язаних комп'ютерних мереж. Він заснований на стеку протоколів TCP/IP. Інтернет базується на Всесвітній павутині (WWW) та багатьох інших системах передачі даних.

Саме особливості Інтернету зумовлюють його величезний потенціал і всезростаючу роль у сучасному світі. Інтернет - це такий технічний засіб і канал зв'язку, який характеризується відсутністю централізованої організаційної структури. Також цей канал характеризується високою швидкістю розповсюдження інформації. Головною відмінною рисою Інтернету є своєчасне оновлення контенту. І головна вимога, яку суспільство висуває до мережі, це наявність актуальної інформації.

Постійне оновлення контенту призводить до збільшення темпів зростання інформації. Згідно з прогнозами IDC (International Data Corporation), кількість даних на планеті щонайменше подвоюватиметься кожні два роки.

У зв'язку з цим виникає необхідність впровадження спеціальних інформаційно-пошукових систем для пошуку необхідної інформації. Інформаційно-пошукова система — програмний комплекс, який забезпечує пошук

і відбір необхідних даних у спеціальній базі даних з описом джерел інформації (індекс) на основі інформаційно-пошукової мови та відповідних правил пошуку.

Основним завданням будь-якої інформаційно-пошукової системи є пошук інформації, релевантної інформаційним потребам користувача.

Релевантність - це відповідність результатів пошуку сформульованому запиту. Такими програмами пошуку інформації є пошукові системи та парсери.

Парсинг (від англійського «parsing» — «аналіз, розбір») — це лінійне порівняння послідовності слів з правилами мови. Поняття «мова» розглядається в найширшому контексті. Це може бути людська мова, яка використовується для людського спілкування. А може і формалізована мова, зокрема будь-яка мова програмування.

Парсинг веб-сайту — це послідовний синтаксичний аналіз інформації, розміщеної на сторінках Інтернету. Збирання веб-сайтів є найефективнішим рішенням для автоматизації збору та зміни інформації.

Для написання парсерів підходять будь-які мови програмування, які використовуються для створення програм для роботи з всесвітньою павутиною. Веб-програми для парсингу зазвичай пишуться на C++, Delphi, Perl, Ruby, Python, PHP.

Порівняно з людиною, комп’ютерна програма аналізатора:

1. Здатний швидко проаналізувати більше тисячі веб-сторінок (за кілька секунд);

2. Точно підібрати необхідну інформацію;

3. Оформіть підсумкові дані в необхідній формі (база даних або таблиця).

Порівняно з пошуковою системою, комп’ютерна програма аналізатора:

1. Має більш високу швидкість відбору необхідної інформації;

2. Відрізняється підвищеною точністю: містить конкретні дані за заданими критеріями пошуку;

3. Можна універсально налаштувати під вимоги та запити будь-якого користувача.

У зв'язку з цим в даній роботі було поставлено завдання створити парсер на мові Python для навігаційних та інформаційних технологій.

1 ПРОГРАМИ І МЕТОДИ ПОШУКУ І ЗБОРУ ДАНИХ. ПАРСИНГ

1.1 Огляд сучасних інформаційних технологій і апаратно-програмного забезпечення для пошуку і збору інформації

Інформаційні технології - сукупність комп'ютерної техніки, телекомунікаційного обладнання, каналів передачі даних та інформаційних систем, засобів комутації та управління інформаційними потоками, а також організаційних структур, правових і нормативних механізмів, що забезпечують їх ефективне функціонування.

Аналізуючи загальне розуміння інформаційних технологій, можна відзначити, що ІТ охоплюють усі сфери передачі, зберігання, сприйняття інформації. Але в більшості випадків ІТ асоціюється з комп'ютерними технологіями, оскільки поява комп'ютерів вивела інформаційні технології на новий рівень розвитку.

Тобто з появою персонального комп’ютера розпочався новий етап у розвитку інформаційних технологій. Головною метою ІТ є задоволення особистих інформаційних потреб людини, як для професійної сфери, так і для повсякденного використання.

У розвитку інформаційних технологій можна виділити певні етапи:

1 етап (перші 60-70-ті роки) - основним напрямком була автоматизація рутинних дій людини, обробка даних в обчислювальних центрах в режимі колективного користування, в умовах обмежених можливостей апаратури, характеризується проблемою обробки великих обсягів інформації. .

2 етап (70-ті рр.) – поява персональних комп’ютерів, розповсюдження комп’ютерів серії IBM/360, орієнтація на окремого користувача, використання централізованої обробки даних, і децентралізованої, яка базується на вирішенні локальних завдань і роботі з локальними. бази даних на робочому місці користувача.

3 етап (80-ті роки) – починають використовувати комп’ютер непрофесіонали, створення інформаційних технологій, спрямованих на вирішення завдань, одна з яких – максимально задовольнити потреби користувача та створити відповідний інтерфейс для роботи в комп’ютерному середовищі.

4 етап (90-ті роки) - створення сучасної технології між організаційними комунікаціями та інформаційними системами, організація захисту та безпеки інформації, організація доступу до стратегічної інформації, розробка угод і встановлення стандартів, протоколів для комп'ютерних комунікацій.

Метою інформаційних технологій є виробництво інформації для аналізу людиною та прийняття на її основі рішень щодо виконання будь-яких дій.

1.2 Збір даних за допомогою засобів емуляції поведінки користувача у браузері

Одним із засобів імітації поведінки користувача в браузері є Selenium.

Selenium – це проект, який розробляє серію програмних продуктів з відкритим кодом:

Selenium WebDriver,

Selenium RC,

Selenium Server,

Selenium Grid,

Selenium IDE.

Selenium WebDriver — це бібліотека програмного забезпечення для керування браузерами. Також часто використовується коротша назва WebDriver.

Це ціле сімейство драйверів для різних браузерів, а також набір клієнтських бібліотек на різних мовах, які дозволяють працювати з цими драйверами (рис. 1.1).

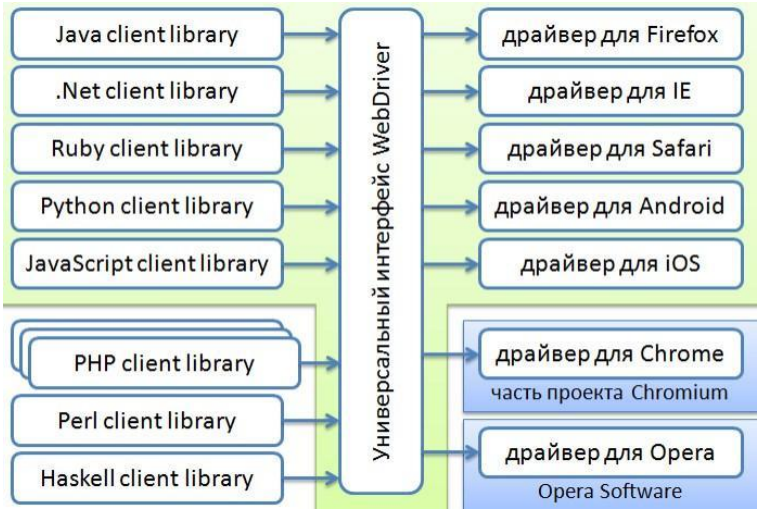


Рисунок 1.1 - Selenium WebDriver

Selenium RC — це попередня версія бібліотеки керування браузером. Абревіатура RC в назві цього продукту розшифровується як Remote Control, тобто це інструмент «віддаленого» керування браузером.

Сервер Selenium — це сервер, який дозволяє керувати браузером з віддаленої машини через мережу.

Selenium Grid — це кластер, що складається з кількох серверів Selenium. Він призначений для організації розподіленої мережі, що дозволяє запускати багато браузерів паралельно на великій кількості машин. Selenium Grid має топологію «зірка», тобто включає виділений сервер, який називається «хаб» або «комутатор», а решта серверів називаються «вузлами» або «вузлами» (рис. 1.2).

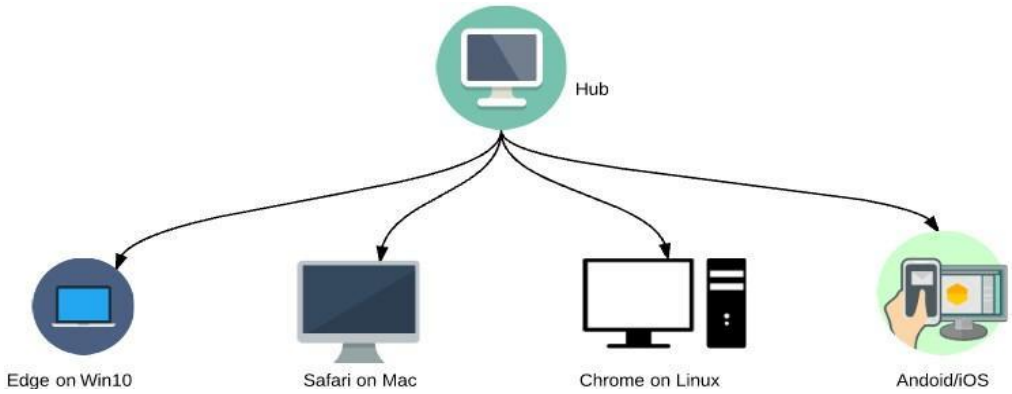


Рисунок 1.2 - Зірчаста топологія для Selenium Grid

Мережа може бути неоднорідною, тобто комутатор і вузли можуть працювати з різними операційними системами, на них можуть бути встановлені різні браузери. Одне із завдань Selenium Grid — «підібрати» відповідний вузол, коли під час запуску браузера задані вимоги до нього — тип браузера, версія, операційна система, архітектура процесора та ряд інших атрибутів.

Selenium IDE — це плагін для браузера Firefox, який може записувати дії користувача, відтворювати їх, а також генерувати код для WebDriver або Selenium RC, у якому виконуються ті самі дії.

1.3 Збір даних за допомогою API

API (від англ. application programming interface) — інтерфейс для взаємодії сайту зі сторонніми програмами і серверами, набір готових класів, процедур, функцій, структур і констант, що надаються додатком (бібліотекою, сервісом) або операційна система для використання у зовнішніх програмних продуктах.

Вхід і реєстрація на різних онлайн-сервісах або платформах через облікові записи соціальних мереж є використанням API. У цьому випадку сервіси або програми використовують бази даних соціальних мереж. При цьому сервіс може отримувати інформацію про користувача та маніпулювати нею у своїх цілях.

Open API — це система для розробників сторонніх сайтів, яка надає можливість легко авторизувати користувачів соціальної мережі на сайті. Крім того, за згодою користувачів ви можете отримати доступ до інформації про їхніх друзів, фотографій, аудіозаписів, відео та інших даних для більш глибокої інтеграції з проектом.

Незважаючи на всю зручність використання API, є одне обмеження - соціальна мережа не може видати всі дані, які користувачі бачать в інтерфейсі. Ці обмеження мають дві причини:

соціальні мережі намагаються зберегти конфіденційність своїх користувачів;

деякі функції надто сильно навантажують серверну частину програми. Щоб подолати це обмеження, слід використовувати такий механізм, як веб-сайту.

1.4 Збір даних за допомогою семантичного аналізу веб-сторінок

Парсинг веб-сайту — це послідовний синтаксичний аналіз інформації, розміщеної на сторінках Інтернету.

Для розбору html найчастіше використовуються наступні варіанти:

Регулярні вирази. Вони є найбільш універсальним і настроюваним семантичним аналізатором. Однак використання виключно їх є досить складним завданням для системного дизайнера, через те, що кожен регулярний вираз потребуватиме дуже спеціалізації, і вони створюють додаткове навантаження на ОС.

BeatifulSoup, lxml — найпопулярніші бібліотеки для парсингу html-сторінок, і вибір однієї з них визначається особистими вподобаннями розробника. Крім того, ці бібліотеки тісно пов’язані: BeautifulSoup почав використовувати lxml як внутрішній парсер для прискорення, а модуль soupparser було додано до lxml.

Синтаксичний аналізатор — це скрипт або програма, яка використовується для збору інформації з веб-сайтів для подальшої обробки та представлення. Його можна написати будь-якою мовою, яка працює з веб-контентом. Веб-програми для парсингу зазвичай пишуться на C++, Delphi, Perl, Ruby, Python, PHP.

Найпростіше написати синтаксичний аналізатор мовами високого рівня, оскільки він уже містить модулі або бібліотеки для роботи з веб-контентом. Мова програмування високого рівня - це мова програмування, призначена для швидкості та простоти використання програмістом. Основною особливістю мов високого рівня є абстракція, тобто введення семантичних конструкцій, які коротко описують такі структури даних і операції над ними, описи яких у машинному коді (або іншій мові програмування низького рівня) дуже довгі і важко зрозуміти.

Термін «комп'ютерна програма» має два визначення. По-перше, це комбінація комп’ютерних інструкцій і даних, що дозволяє апаратному забезпеченню обчислювальної системи виконувати обчислення або керувати функціями. По-друге, це синтаксична одиниця, яка відповідає правилам певної мови програмування. Він складається з визначень і операторів або інструкцій, необхідних для вирішення конкретної функції, завдання або проблеми.

Програма може містити як машинний код, що виконується процесором для досягнення якоїсь мети, так і необхідні для цього дані. Відмінною особливістю програми є її розташування в пам'яті та виконання процесором.

Образ програми зберігається на комп'ютері як виконуваний модуль. Він представляє один файл або групу файлів. З цього зображення завантажувач програмного забезпечення може створити виконувану програму в оперативній пам’яті.

Процес розробки програмного забезпечення складається з кількох етапів, з яких лише безпосереднє створення програмного коду називається «програмуванням».

Запис вихідного коду програм з використанням мов програмування полегшує розуміння та редагування коду. Програмісту допомагають коментарі, які дозволені в синтаксисі більшості мов. Для виконання на комп’ютері готовий текст програми перетворюється (компілюється) у машинний код.

Деякі мови програмування дозволяють динамічну компіляцію. Це означає, що ви можете обійтися без попередньої компіляції програми, а відразу перевести її в інструкції машинного коду безпосередньо під час виконання. Цей процес дозволяє досягти більшої переносимості програм між різними апаратними та програмними платформами, зберігаючи при цьому багато переваг компіляції.

Інтерпретовані програми, для яких не застосовується процес компіляції та які інтерпретуються операційною системою або спеціальними програмами-інтерпретаторами, називаються сценаріями або «сценаріями».

Вихідні тексти комп’ютерних програм на більшості мов програмування містять вбудований алгоритм. Такий підхід у програмуванні називають імперативним.

Також застосовуються інші методики. Опис вихідних і необхідних характеристик, даних, що обробляються, і надання вибору відповідного алгоритму рішення спеціалізованій програмі-інтерпретатору називається декларативним програмуванням. Він включає функціональне та логічне типи програмування.

Програми можуть бути створені в текстовому вигляді та візуально. У першому випадку вихідний код набирається вручну. У другому функціонал програми задається за допомогою елементів графічного інтерфейсу користувача, а текст програми формується автоматично. При такому підході код може бути доступний для ручної модифікації або повністю прихований від програміста.

Парсери дуже часто використовуються в таких областях:

де інформація швидко втрачає свою актуальність і стає непридатною через кілька хвилин;

де копіювання вручну неможливе або вимагає величезних людських витрат (наприклад, щоб відобразити курс обміну, вартість цінних паперів або інсайдерську інформацію);

де відбувається повне або часткове копіювання матеріалів сайту з подальшим розміщенням цих матеріалів на своїх ресурсах.

Отже, можна уявити собі приблизну сферу застосування парсерів:

маркетингові дослідження;

моніторинг ЗМІ в реальному часі;

оновлення новинних порталів (в цьому випадку текст може бути попередньо пропущений через синонімайзер або оброблений рерайтером для підвищення унікальності);

аналіз громадської думки;

автоматичне ціноутворення на основі аналізу цін конкурентів;

побудова списку потенційних користувачів на основі інформації про користувачів ресурсів конкурентів;

розробка мобільних додатків; - аналіз соціальних зв'язків;

збір та обробка спортивної статистики;

аналіз документів у сфері судочинства;

також аналізуються сайти з оглядами фільмів і книг, а також сайти з рецептами, текстами пісень і віршами.

Розбір HTML-сторінки — це процес, який можна розбити на три етапи:

Етап 1: отримання вихідного коду веб-сторінки.

У різних мовах для цього передбачено різні способи, наприклад, у мові програмування Python найчастіше використовується бібліотека «requests», яка зберігає дерево сайту у змінній.

Етап 2: вилучення необхідних даних з html коду. Отримавши сторінку, її необхідно обробити:

відокремити простий текст від гіпертекстової розмітки;

побудувати ієрархічне дерево елементів документа;

правильно реагувати на недійсний код;

зробити вибірку необхідної інформації зі сторінки.

Для цього можна використовувати регулярні вирази або спеціалізовані бібліотеки.

3 етап: закріплення результату.

Безпечно обробивши дані на сторінці, необхідно зберегти їх у необхідному вигляді для подальшої обробки. Існує кілька способів маніпулювання отриманою інформацією:

ввести в базу даних; - запис у файл CSV;

будувати ієрархічні структури JSON;

конвертувати в excel-таблицю;

створити динамічний rss-канал.

Аналоги парсера

В даний час в Інтернеті доступні два типи парсерів:

1. Сервіс парсингу ресурсів, що цікавлять.
2. Використання універсальних програм-парсерів.

У першому випадку клієнт робить запит на збір необхідних даних. В результаті він отримує необхідну інформацію в необхідному вигляді і форматі.

Переваги:

можливість отримання до 10000 записів;

парсинг будь-якого веб-ресурсу;

немає необхідності реєструватися на сайті та використовувати VPN;

надання інформації у зручному форматі.

Недоліки:

висока вартість послуги парсингу сайту;

тривалий час збору даних (від 1 до 3 днів);

додаткові витрати за необхідність доповнення або оновлення даних.

У разі придбання ліцензії на використання готової універсальної програми-парсера замовник отримує можливість аналізувати ресурси різного типу. Прикладами таких готових аналізаторів є X-Parser Light і uParser.

Переваги:

автоматичне розпізнавання тегів на сторінках будь-якого сайту;

наявність редактора для ручної перевірки обраного контенту за допомогою менеджера обробки контенту.

можливість парсингу контенту навіть за відсутності тегів і з формуванням базової розмітки на основі оригінальної;

можливість фільтрувати статті та окремі абзаци за допомогою власних фільтрів на етапі збору контенту;

можливість перевірки статей на наявність ключової інформації в тілі статті;

використання доступних пошукових систем;

можливість парсингу контенту будь-якою мовою.

Недоліки:

немає можливості інтеграції в програму або сервер;

досить висока вартість.

Однак, крім цих двох способів збору даних за допомогою семантичного аналізу веб-сторінок, існує альтернативний варіант, який полягає в розробці власної програми аналізатора. При цьому замовник формує вимоги до джерел інформації та форми її запису після процедури парсингу.

У цьому випадку поєднується збереження достоїнств перших двох видів і позбавлення від їх недоліків. Однак є деякі мінуси:

наявність більш точних налаштувань програми;

чітке визначення джерел інформації.

Виходячи з вищесказаного, можна зробити висновок, що якщо необхідно збирати інформацію, то необхідно спиратися на цілі та вимоги замовника. Не менш важливо фактором вибору є аналіз переваг і недоліків представлених методів парсингу сайту.

Висновки

У першому розділі роботи були проаналізовані та вивчені різні методи пошуку та збору інформації. Кожен із представлених методів має свої переваги, недоліки та особливості роботи. Парсинг сайту виявився найбільш оптимальним способом збору та обробки інформації, оскільки він поєднує в собі переваги представлених методів.

При цьому створення власної програми-парсера допомагає замовнику заощадити значну суму грошей. За часовими витратами розробка парсера може зайняти до 10 днів, але інформація буде оновлюватися миттєво.

2. РЕАЛІЗАЦІЯ АЛГОРИТМІВ ПАРСИНГУ МОВОЮ ПРОГРАМУВАННЯ PYTHON

2.1 Особливості алгоритму автоматичного збирання та синтаксичного аналізу даних

Перед проектуванням архітектури модуля необхідно сформувати всі вимоги:

1. Як вхідні дані модуль повинен приймати повні робочі посилання на сторінки з товарами, що цікавлять користувача;
2. Модуль повинен витягувати дані про актуальні новини з веб-ресурсів;
3. Всі знайдені дані повинні звірятися з наявними в таблицях бази даних, у разі наявності нових даних, бази даних повинна оновлюватися.
4. Модуль повинен легко налаштовуватись на різні новинні ресурси.
5. Вихідний код розробленого модуля має бути відкритим та легким для розуміння;
6. Модуль повинен працювати навіть із невалідним HTML-кодом;
7. Модуль повинен мати високу продуктивність навіть у разі більших обсягів даних;
8. Необхідна орієнтованість на включення до інших проектів (системи інтелектуального аналізу даних);
9. Модуль має бути кросплатформним.

На початку роботи модуля відбувається підключення бібліотеки QueryList для створення парсерів, потім підключення до створеної БД, після чого модуль очікує введення користувачем посилання на сторінку, що його цікавить. Обов'язковим пунктом є перевірка введеної адреси на валідність, використовуючи шаблон, що включає всі обов'язкові посилання на веб-ресурс. Після успішної перевірки посилання за допомогою функції мови PHP parse_url визначаємо інформаційний ресурс для парсингу, відповідно до якого вибираються налаштування для парсеру, новина та її акутальність.

Вивантаження даних починається з надсилання HTTP-запиту браузером до сервера торгового майданчика із встановленими заголовками для завантаження потрібної сторінки.

На запит браузера сервер видає відповідь із кодом стану, що допомагає зрозуміти результат запиту. Успішне завантаження сторінки HTTP відповідь буде визначатися отриманням в HTTP-відповіді коду 200, однак, на даному етапі є ймовірність отримання в HTTP-відповіді коду з помилкою.

Найпоширеніші коди станів наведені в таблиці 2.1. Відповідно до отриманого коду стану користувачеві буде виведено відповідне повідомлення з поясненнями, достатніми для розуміння причини помилки.

Таблиця 1 – Коди станів HTTP

Код стану	Пояснення
200 OK	Успішний запит
404 Not found	Запит до неіснуючого ресурсу
500 Internal server error	Наявність внутрішньої помилки сервера
401 Unauthorized	Для запиту потрібна аутентифікація
403 Forbidden	Доступ до ресурсу заборонено
429 Too Many Requests	Запит відхиляється через огр

У разі успішного запиту модуль завантажує сторінку необхідного ресурсу та за допомогою встановлених параметрів для парсингу зчитує необхідну інформацію про новину. Далі БД перевіряється на наявність отриманої інформації в таблиці item і за необхідності оновлюється.

У циклі відбувається вивантаження відгуків із поточної сторінки та формування URL наступної.

2.2 Програмна реалізація алгоритмів збору та аналізу даних

Для отримання даних у роботі використана бібліотека QueryList.

Серед альтернативних варіантів було детально розглянуто використання регулярних виразів, модулів DOM та XPath, бібліотеки Simple HTML DOM.

Регулярні вирази є інструментом для отримання інформації за допомогою шаблонів. У разі використання регулярних виразів перехід на новий торговий майданчик супроводжувався б створенням великої кількості нових складних регулярних виразів.

Отже, чим більше торгових майданчиків охоплював би модуль, тим більше регулярних виразів довелося б зберігати і навіть при незначній зміні коду веб-ресурсу регулярні висловлювання теж доводиться коригувати. До того ж, при великих обсягах даних робота регулярних виразів займає багато часу.

Згідно DOM-моделі, документ є ієрархією (деревом).

Кожен HTML-тег утворює вузол дерева із типом «елемент». Вкладені у нього теги стають дочірніми вузлами. Для представлення тексту створюються вузли із типом «текст». Використовуючи даний метод необхідні дані можна отримати за ідентифікатором, ім'ям, іншими атрибутами елемента дерева або ж за допомогою унікального шляху, спускаючись вниз по дереву. Однак шлях до елемента може виявитися занадто складним і його також необхідно змінювати при зміні структури веб-ресурсу.

XPath - це мова запитів до елементів XML або XHTML документа. Щоб отримати дані, що цікавлять, необхідно створити запит, що описує ці дані.

Використання DOM та XPath поступається обраній бібліотеці відсутністю можливості роботи з невалідним html-кодом.

Бібліотеки QueryList і Simple HTML DOM, окрім відсутності вищезгаданих мінусів, мають зручні для пошуку елементів HTML-коду можливості, наприклад, звернення до HTML-елементів за допомогою CSS селекторів, що спрощує парсинг нових торгових майданчиків.

2.3 Області застосування програм-парсерів

Парсери дуже часто використовуються в таких областях:

Веб-скрапінг: Парсери використовуються для витягування даних з веб-сторінок, таких як новини, відгуки, ціни товарів, рейтинги тощо. Це може бути корисно для створення власної бази даних, аналізу конкурентів, моніторингу ринку та іншого.

Пошук інформації: Парсери використовуються для пошуку та збору інформації з різних джерел, таких як бази даних, файлові системи, API тощо. Це може бути корисно для знаходження специфічних даних, створення звітів, аналізу даних та іншого.

Аналітика даних: Парсери можуть допомагати у витягуванні та підготовці даних для аналізу. Вони можуть витягувати та обробляти великі обсяги даних з різних джерел, забезпечуючи зручну форму для подальшого аналізу та візуалізації.

Автоматизація задач: Парсери можуть бути використані для автоматизації повсякденних задач, які вимагають обробки або аналізу даних. Наприклад, вони можуть автоматично збирати дані звіти з різних джерел та генерувати зведені звіти або повідомлення.

Інтернет-маркетинг: Парсери використовуються для аналізу ринку, конкурентів та цільової аудиторії. Вони можуть витягувати дані про продукти, ціни, відгуки, рейтинги та іншу інформацію, що допомагає приймати стратегічні рішення у маркетингу та рекламі.

Отже, можна уявити собі приблизну сферу застосування парсерів:

- маркетингові дослідження;

моніторинг ЗМІ в реальному часі;

оновлення новинних порталів (в цьому випадку текст може бути попередньо пропущений через синонімайзер або оброблений рерайтером для підвищення унікальності);

аналіз громадської думки;

автоматичне ціноутворення на основі аналізу цін конкурентів;

побудова списку потенційних користувачів на основі інформації про користувачів ресурсів конкурентів;

розробка мобільних додатків; - аналіз соціальних зв'язків;

збір та обробка спортивної статистики;

аналіз документів у сфері судочинства;

також аналізуються сайти з оглядами фільмів і книг, а також сайти з рецептами, текстами пісень і віршами.

2.4 Особливості розробленого ПЗ для реалізації алгоритмів парсингу і його місце серед існуючих аналогів

Серед наявних на даний момент засобів збирання даних з вебресурсів розглянемо 2 найближчих конкурентних рішення – програму для парсингу різних сайтів Datacol та сервіс збору та агрегації відгуків Mneniya.pro.

Парсер Datacol працює з регулярними виразами і XPath-запитами, має налаштування для парсингу великої кількості майданчиків.

Інтерфейс програми дозволяє користувачеві самостійно налаштувати парсер. Однак, для вищевказаних цілей даний продукт має такі недоліки, як закритий вихідний код, ручне налаштування парсера та обмеженість безкоштовної версії.

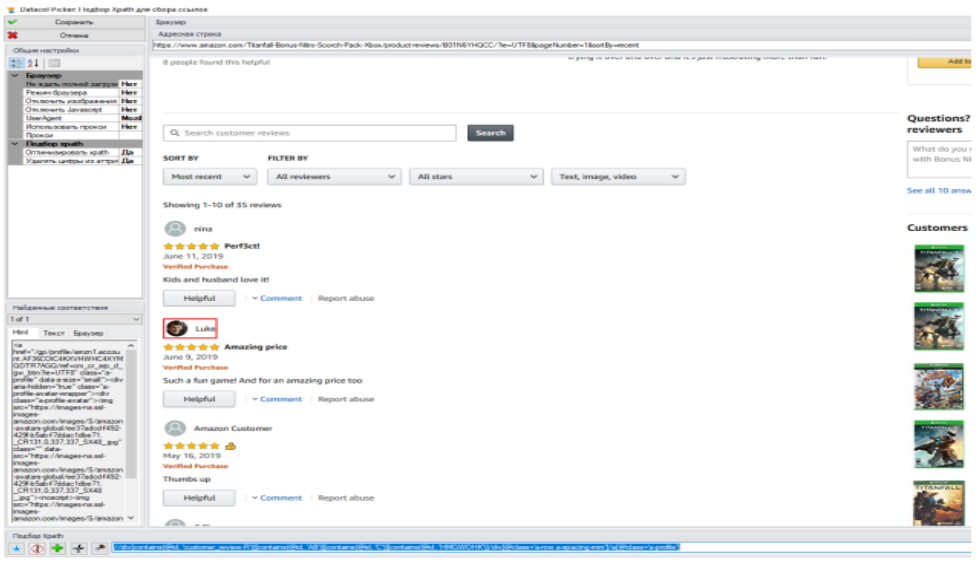


Рисунок 1 – Інтерфейс налаштування XPath-виразів парсера Datacol

Сервіс Mneniya.pro також є платним, не має відкритого вихідного коду і крім цього орієнтований на інші завдання – виведення відгуків на сайт клієнта.

Більшість аналогічних рішень є парсерами з графічним інтерфейсом, які є окремим продуктом, в нашому випадку необхідно рішення для легкого включення в складніші системи, без зайвого функціоналу.

Модуль, що розробляється, буде виконаний більш низькорівневим для більшої гнучкості при включенні в системи інтелектуального аналізу і з відкритим вихідним кодом, доступним для модифікації та розширення.

В другому розділі розглянуто особливості алгоритму автоматичного збирання та синтаксичного аналізу даних. Також розглянуто програмна реалізація алгоритмів збору та аналізу даних. Визначено області застосування програм-парсерів. А також описується особливості розробленого ПЗ для реалізації алгоритмів парсингу і його місце серед існуючих аналогів.

3. МЕТОДИКА ПРОЕКТУВАННЯ СИСТЕМ ЗБОРУ І СИНТАКСИЧНОГО АНАЛІЗУ ДАНИХ ІНСТРУМЕНТАЛЬНИМИ ЗАСОБАМИ МОВИ PYTHON

3.1 Етапи розробки СЗСАД (системи збору і синтаксичного аналізу даних)

Розробка СЗСАД (системи збору і синтаксичного аналізу даних) включає наступні етапи:

1. Визначення вимог: На цьому етапі встановлюються вимоги до системи збору і синтаксичного аналізу даних. Це можуть бути вимоги до функціональності, продуктивності, безпеки, масштабованості та інші.
2. Проектування архітектури: На цьому етапі визначається загальна архітектура системи. Вона включає складові, модулі, комунікацію між ними та інші аспекти системи.
3. Розробка інтерфейсу: На цьому етапі розробляються інтерфейси, які дозволяють взаємодіяти з системою збору і синтаксичного аналізу даних. Це можуть бути графічні інтерфейси, API або інші засоби комунікації.
4. Розробка збірника даних: Цей етап включає розробку компонентів, які забезпечують збір даних з різних джерел. Це можуть бути веб-скрапери, API-інтеграції, розширення для браузерів та інші механізми збору даних.
5. Розробка модуля синтаксичного аналізу: На цьому етапі створюються модулі для синтаксичного аналізу зібраних даних. Це можуть бути моделі машинного навчання, алгоритми обробки мови, статистичні методи або комбінація різних підходів.
6. Інтеграція та тестування: На цьому етапі розроблені компоненти і модулі інтегруються в єдину систему. Проводяться тестування для

перевірки функціональності, продуктивності, стійкості та інших аспектів системи.

7. Впровадження: Після успішного тестування система готова до впровадження. Здійснюється інсталяція системи на необхідному обладнанні або хостинг-сервері, налаштування параметрів і початок експлуатації.
8. Підтримка і поновлення: Після впровадження система потребує підтримки, включаючи моніторинг, виявлення та виправлення помилок, поновлення системи з урахуванням нових вимог та технологічних розробок.

3.1.1 Опис технічного завдання

1. Мета і цілі.

Необхідно розробити скрипт для програми-парсера. Письмова програма має проаналізувати необхідні джерела та зібрати актуальні дані щодо акутальних новин.

2. Запис результатів

Запис та збереження результатів аналізу даних про актуальні події здійснювати у файлах із розширенням csv. А для даних про інформацію що до новини створюється додаткова таблиця в базі.

3. Джерела даних

Джерела даних для парсингу новин повинні бути актуальними і постійно оновлюватися. Для цього було обрано наступний сайт:

- «https://vechirniy.kyiv.ua/», де вкладка Основна сторінка містить список актульнтх нещодавно опублікованих новин (рис. 3.1).

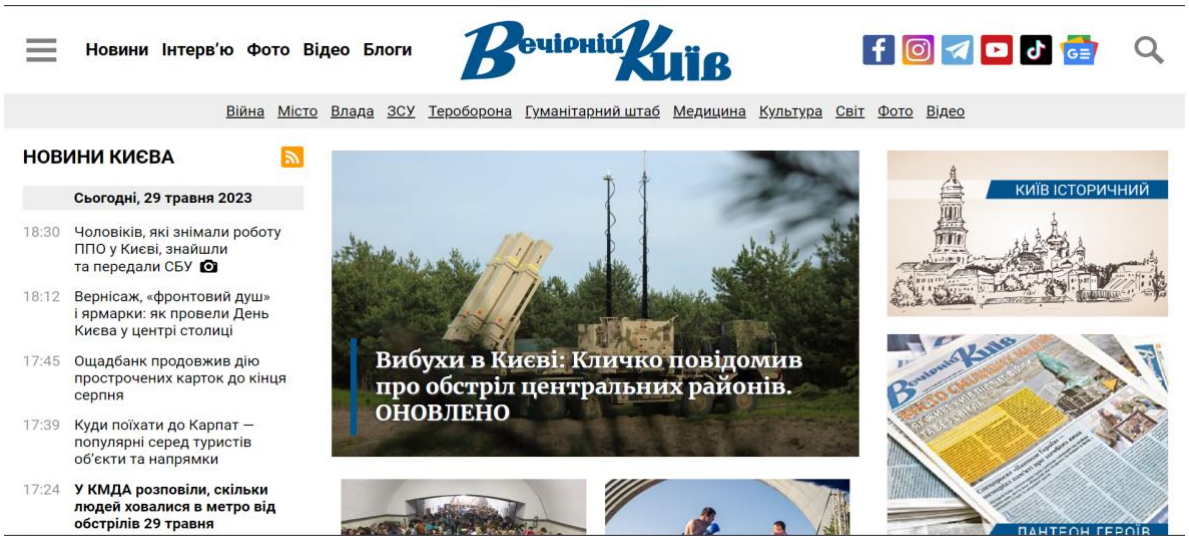


Рисунок 3.1 - Відображення сайту для парсингу новин в “лайв-режимі”

3.1.2 Вибір інструментальних засобів розробки

Програми для веб-скопіювання зазвичай пишуться мовами високого рівня, такими як C++, Delphi, Perl, Ruby, Python, PHP. В ході роботи було проведено ретельне дослідження переваг і недоліків кожної з представлених мов програмування.

В результаті мова Python3 стала найбільш оптимальною мовою програмування для написання програми-парсера для вебсайтів.

Такий вибір був обумовлений наявністю наступних переваг цієї мови програмування:

простота синтаксису;

достатня зрозумілість мови (це часто сприяє прийняттю очевидних рішень навіть для неочевидних завдань);

наявність готових модулів і бібліотек;

наявність багатьох книг з практичними прикладами, які спрощують написання коду і полегшують розуміння мови.

Як наслідок, цією мовою легко створювати програми. На Python було створено багато перевірених бібліотек, які допомогли в написанні цієї роботи.

Python — мова програмування високого рівня загального призначення, орієнтована на підвищення продуктивності розробника та читабельності коду. Головною особливістю є мінімалістичний синтаксис ядра Python з одночасною наявністю великої кількості корисних функцій у стандартній бібліотеці.

Python підтримує кілька парадигм програмування:

структурну;

об'єктно-орієнтований;

функціональні;

наказовий;

аспектно-орієнтований.

Основні архітектурні особливості:

динамічна типізація;

автоматичне управління пам'яттю;

повна інтроспекція

наявність механізму обробки винятків;

підтримка багатопоточних обчислень;

зручні високорівневі структури даних.

Код у Python організований у функції та класи, які можна об’єднати в модулі, які, у свою чергу, можна запакувати.

Довідковою реалізацією Python є інтерпретатор CPython, який підтримує більшість поширених платформ. Він поширюється за безкоштовною ліцензією Python Software Foundation, що дозволяє використовувати його без обмежень у будь-якій програмі. Існують реалізації інтерпретатора для JVM (компілюємо), MSIL (компілюємо), LLVM та інші. Проект PyPy забезпечує реалізацію Python за допомогою JIT-компіляції, що значно прискорює виконання програм Python.

3.1.3 Опис середовища та системи

У роботі були використані такі бібліотеки:

Requests - Запити на надсилання HTTP/HTTPS запитів;

BeautifulSoup для отримання даних із файлів HTML та XML;

Lxml для обробки сторінок XML і HTML на Python;

Databases Libraries.

Модуль Python, який надає програмісту можливість виконувати структурний аналіз файлів із розширенням csv (Comma Separated Values – змінні, розділені комами);

Тепер розглянемо докладніше кожну з використовуваних бібліотек окремо.

Бібліотека запитів використовується для надсилання органічних запитів HTTP/HTTPS без необхідності вручну будувати дерево сайту.

Водночас програмісту немає необхідності вручну додавати рядки запиту до URL-адрес або декодувати дані. Крім того, Requests дозволяє вставляти необхідні заголовки, наприклад:

User-Agent (версія операційної системи та тип браузера);

Проксі (проксі-сервери, через які можна зайти на сайт, якщо ip користувача вже заблокований). [16]

Основні функції бібліотеки запитів: - утримання зв'язків;

міжнародні домени та URL-адреси;

сесії зі збереженням файлів cookie;

перевірка SSL у формі браузера;

автоматичне декодування контенту;

формат відповіді в кодуванні UTF-8;

підтримка http(S) proxy;

завантаження складених файлів;

потокове завантаження;

фрагментовані запити.

Бібліотека BeautifulSoup працює з будь-яким парсером. Вона надає прості та зручні способи навігації, пошуку та зміни бажаної форми. Ця бібліотека дозволяє

знаходити та зберігати дані різних типів. До них відносяться посилання, теги, текст. А також можна шукати та створювати списки з однаковими тегами.

Набір інструментів lxml є зв'язкою бібліотек libxml2 і libxslt C для Python. Він унікальний тим, що поєднує швидкість і повноту функцій XML цих бібліотек із простотою рідного API Python, який перевершує добре відомий API ElementTree. Його остання версія працює з усіма версіями CPython від 2.6 до 3.6.

Модуль CSV дозволяє виконувати структурний аналіз файлів із розширенням csv. Файл CSV – це текстовий файл, кожен рядок якого містить кілька полів, розділених комами або іншими роздільниками. можна розглядати кожен

рядок як рядок, а кожне поле як стовпець. Формат CSV не має стандарту, але ці файли досить схожі, щоб модуль csv міг розпізнати більшість із них. Цей модуль дозволяє робити наступне:

створити файл із розширенням csv;

робити в ньому записи;

створювати різні роздільники;

читати файли з розширенням csv.

3.2.1 Основні вимоги до запуску та роботи з веб-сайтом

До програмного забезпечення висуваються наступні вимоги:

можливість зробити запит на статус реквесту;

реалізація алгоритму вибору сайту для парсингу;

реалізація методу підключення до бази даних, пошук інформації;

візуалізація результатів роботи програми у вигляді діалогового вікна.

Вхідними параметрами є URL (адреса, що ідентифікує ресурс у інтернеті

Вихідними даними для розробленого програмного продукту є база даних з інформацією про актуальні новини.

3.2.2 Вимоги до інтерфейсів інформаційної системи

Інтерфейс програмного продукту повинен бути декларативним. Користувач повинен мати можливість вводити URL в системі, переглядати заголовки на сайтах, та дивитися вміст статтей.

Взаємодія із застосунком здійснюється через інтерфейс пк додатку.

Окрім цього, у користувача повинна бути можливість розширювати функціональність розробленого програмного продукту, маючи доступ до відкритого коду.

3.2.3 Вимоги до технічного забезпечення

Мінімальні вимоги до апаратного забезпечення комп'ютера або мобільного пристрою для коректної роботи розробленого програмного продукту:

Процесор: 2 x Intel Xeon E5-2690, 8 ядер/16 потоків, тактова частота 2.9 ГГц (або еквівалентний процесор того часу)

Обсяг оперативної пам'яті: 32 ГБ або більше (рекомендується 64 ГБ для більш продуктивної роботи)

Дискова підсистема: 4 x 300 ГБ SAS або 4 x 500 ГБ SATA (залежно від потреб і можливостей)

Для ПК користувача:

Процесор: Intel Core i3-2100, 2 ядра/4 потоки, тактова частота 3.1 ГГц (або еквівалентний процесор того часу)

Обсяг оперативної пам'яті: 4 ГБ або більше (рекомендується 8 ГБ для більш продуктивної роботи)

Дискова пам'ять: 500 ГБ SATA HDD або більше (можна також розглянути використання SSD для покращення швидкості доступу до даних)

Мережевий адаптер: до 1 Гбіт/с

Для коректного відображення застосунку необхідна наявність веббраузера із підтримкою стандартів HTML5, наприклад Google Chrome, Mozilla Firefox чи інші аналоги.

Розроблене програмне забезпечення може виконуватись на операційних системах Windows та Linux за умови, що встановлене програмне забезпечення, необхідне для роботи продукту.

3.2.4 Вимоги до програмного забезпечення

Для тестування програмного забезпечення необхідно виконати наступні дії:

1. Зареєструватися в системі;
2. Зробити вхід в систему;
3. Обрати сайт;
4. Спарсити дані з сайту.

При виконанні вище перерахованих дії для тестування роботи програмного забезпечення, на користувач буде зареєстрований у системі, його данні з’являться у базі даних, та матиме можливість обирати сайт та парсити дані.

3.3 Розробка програми-парсингу

Розробка програми-парсера - це процес створення програмного забезпечення, яке здатне аналізувати вхідні дані і витягувати з них потрібну інформацію відповідно до визначених правил або шаблонів. Парсинг використовується для обробки структурованих або напівструктурованих даних, таких як текстові файли, HTML-сторінки, JSON-об'єкти, XML-документи тощо.

Розробка програми-парсера включає в себе такі етапи:

Визначення вимог: Визначення вхідних даних, які потрібно парсити, і визначення очікуваного результату. Встановлення формату або синтаксису вхідних даних.

Вибір підходу до парсингу: Вибір методу або технології для виконання парсингу. Це може бути використання готових бібліотек або фреймворків, створення власного парсера або використання інструментів для генерації парсерів.

Реалізація парсера: Розробка коду програми-парсера на вибраній мові програмування. Це включає реалізацію алгоритмів парсингу, обробку вхідних даних, витягування потрібної інформації та збереження результату.

Тестування: Виконання тестів для перевірки правильності роботи програми-парсера. Враховуючи різні варіанти вхідних даних і різні сценарії використання.

3.3.1 Етапи парсингу

Етапи парсингу (синтаксичного аналізу) даних включають наступні етапи:

Лексичний аналіз (Tokenization): На цьому етапі вхідний текст розбивається на лексеми або токени. Лексеми можуть бути словами, операторами, розділовими знаками тощо. Кожен токен має своє значення і тип, які використовуються на наступних етапах парсингу.

Синтаксичний аналіз (Parsing): На цьому етапі виконується синтаксичний аналіз токенів для визначення структури речень або виразів. Це включає використання граматики або правил, щоб перевірити, чи відповідають токени валідним синтаксичним правилам мови або формату.

Семантичний аналіз (Semantic Analysis): На цьому етапі проводиться перевірка семантики речення або виразу, тобто здійснюється аналіз змісту і правильності використання змінних, функцій, типів тощо. Цей етап може включати виконання типової перевірки, перевірку прав доступу, виконання оптимізацій та інші завдання.

Генерація вихідного коду (Code Generation): Якщо парсинг виконувався для компілятора, на цьому етапі генерується вихідний код на певній мові програмування або цільовому асемблері. Генерується код, який відповідає семантичному аналізу та проміжному представленню.

3.3.2 Структура роботи програми і її основні функції

У створеній програмі-парсері використовуються такі функції:

1. Код взаємодії з сервером через HTTP-запити

Цей фрагмент коду надсилає запит на сервер для отримання дерева сайту у форматі html. В якості вхідних даних функція приймає адреси сторінок в Інтернеті.

Опис роботи коду:

Виконується GET-запит до вказаної URL-адреси за допомогою requests.get(url), і результат зберігається у змінній r.

Отримується статусний код відповіді сервера за допомогою r.status_code і зберігається у змінній status_code.

Перевіряється, чи статусний код дорівнює 200. Якщо так, виводиться повідомлення про успішний запит.

Інакше, якщо статусний код дорівнює 403, виводиться повідомлення про відхилення запиту.

Викликається r.raise_for_status(), що перевіряє, чи статусний код відповіді є успішним (200) і в разі невдачі викликає виняток.(рис. 3.2).

```
r = requests.get(url) # Запит на взяття ссилки
status_code = r.status_code # Запит на присвоєння статусу запита

if status_code == 200: # Якщо статус код дорівнює 200 - запит успішний
    print("Status code connection: [200 - \"OK\"]")
elif status_code == 403: # Якщо статус код дорівнює 403 - запит відхилено
    print("Status code connection: [403 - \"Connection forbidden by host!\"]")

r.raise_for_status()
```

Рисунок 3.2 - Фрагмент коду “запит на сервер”

2. Фрагмент коду парсингу (рисунок 3.3).

У наданому фрагменті коду використовується бібліотека BeautifulSoup для парсингу HTML та регулярний вираз для знаходження елементів з певним класом. Ось пояснення кожного кроку коду:

```
try:
    main = html.find(class_=(re.compile(r"^content.*")))  # Пошук елемента з класом, що починається на "content"
    a = main.find_all('a')  # Знаходження всіх елементів <a> всередині знайденого елемента

    if content_value < set(list(a)):  # Порівняння content_value зі списком знайдених елементів <a>
        content = a  # Якщо значення content_value менше, ніж знайдені елементи <a>, присвоїти їх змінній content

except AttributeError:
    print("Parse from class \"content.*\" failed!")  # Вивести повідомлення про невдале парсинг з класу "content.*"
```

Рисунок 3.3 - Код парсингу сторінки за вказаним атрибутом

3. Опис класу Ui_MainWindow.

Цей код створює базову структуру користувацького інтерфейсу для головного вікна програми і може бути розширений додатковими функціями та дизайном (рис. 3.4).

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(581, 494)
        self.centralwidget = QtWidgets.QWidget(parent=MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pushButton = QtWidgets.QPushButton(parent=self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(30, 30, 101, 41))
        self.pushButton.setObjectName("pushButton")
        self.lineEdit = QtWidgets.QLineEdit(parent=self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(160, 40, 411, 21))
        self.lineEdit.setObjectName("lineEdit")
        self.pushButton_2 = QtWidgets.QPushButton(parent=self.centralwidget)
        self.pushButton_2.setGeometry(QtCore.QRect(30, 90, 101, 41))
        self.pushButton_2.setObjectName("pushButton_2")
        self.textEdit = QtWidgets.QTextEdit(parent=self.centralwidget)
        self.textEdit.setGeometry(QtCore.QRect(160, 90, 411, 361))
        self.textEdit.setObjectName("textEdit")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(parent=MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 581, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(parent=MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
```

Рисунок 3.4 - Код структури користувацького інтерфейсу

Користувацький інтерфейс(рис. 3.5).

У інтерфейсі розроблено 2 клавіші “Обрати” і “Спарсити” та 2 поля для введення та відображення інформації після парсингу сайту

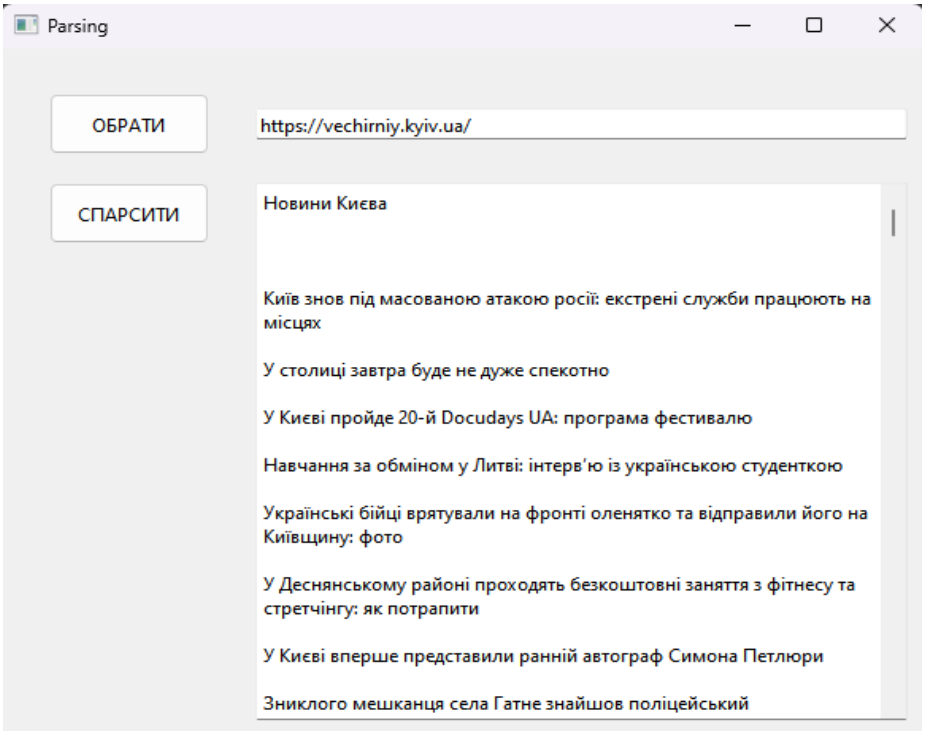


Рисунок 3.5 - Користувацький інтерфейс програми

4. Налаштування текстових елементів (рис. 3.6).

Цей фрагмент коду використовується для налаштування текстових елементів у графічному інтерфейсі, щоб забезпечити зрозумілі та інформативні надписи для користувача. Зазвичай цей код викликається під час налаштування інтерфейсу, щоб встановити початкові тексти для різних елементів.

```
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Parsing"))
    self.pushButton.setText(_translate("MainWindow", "ОБРАТИ"))
    self.lineEdit.setText(_translate("MainWindow", "https://rozetka.com.ua/notebooks/c80004/"))
    self.pushButton_2.setText(_translate("MainWindow", "СПАРСИТИ"))
```

Рисунок 3.6 - Налаштування текстових елементів

5. Головний файл програми (рис. 3.7).

Цей фрагмент коду виконується лише тоді, коли файл запускається як головний файл програми (не імпортується як модуль). Він створює інстанцію додатку QApplication, головне вікно QMainWindow та інстанцію Ui_MainWindow, яка відповідає за налаштування графічного інтерфейсу. Метод setupUi викликається для налаштування головного вікна. Після цього головне вікно відображається за допомогою show(), а потім запускається головний цикл програми за допомогою app.exec(). Після закриття вікна програма виходить з циклу та завершується за допомогою sys.exit().

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec())
```

Рисунок 3.7 - Код запуску головного вікна

6. Функція “Парсинг” (рис. 3.8).

Цей код виконує підготовчі дії перед початком парсингу. Він створює таблицю "content" у базі даних для зберігання інформації, такої як "info", "date" і "url". Якщо таблиця вже існує, вона не буде перезаписана, а якщо її не існує, вона буде створена з відповідною структурою колонок.

```
def parsing():
    global url, cl, content

    s = re.match(r'https?://(?:www\.|)([\w.-]+).*', url)
    domain = s.group(1)

    form.textEdit.setText("")

    db = sqlite3.connect(f'databases/{domain}.db')
    cursor = db.cursor()

    cursor.execute(f'''CREATE TABLE IF NOT EXISTS content
                   (id INTEGER PRIMARY KEY AUTOINCREMENT,
                   info VARCHAR(1000),
                   date VARCHAR(100),
                   url VARCHAR(200))''')
    db.commit()
    db.close()
```

Рисунок 3.8 - Опис функції

7. Функція “choose()” (рис 3.9).

Цей код виконує дії, коли користувач обирає або вводить URL-адресу в текстовому полі. Він оновлює глобальну змінну url зі значенням, введеним користувачем, і виводить це значення на консоль разом з повідомленням "choose"

```
def choose():
    global url
    url = form.lineEdit.text()
    print(url)
    print("choose")
```

Рисунок 3.9 - Код функції choose()

3.3.3 Технологічні показники і результати роботи програми

Після виконання програма створює файл з розширенням db, який оновлюються при кожному запуску програми:

Файли записів із баз даних містять інформацію про останні акутальні новини. (рис 3.10).

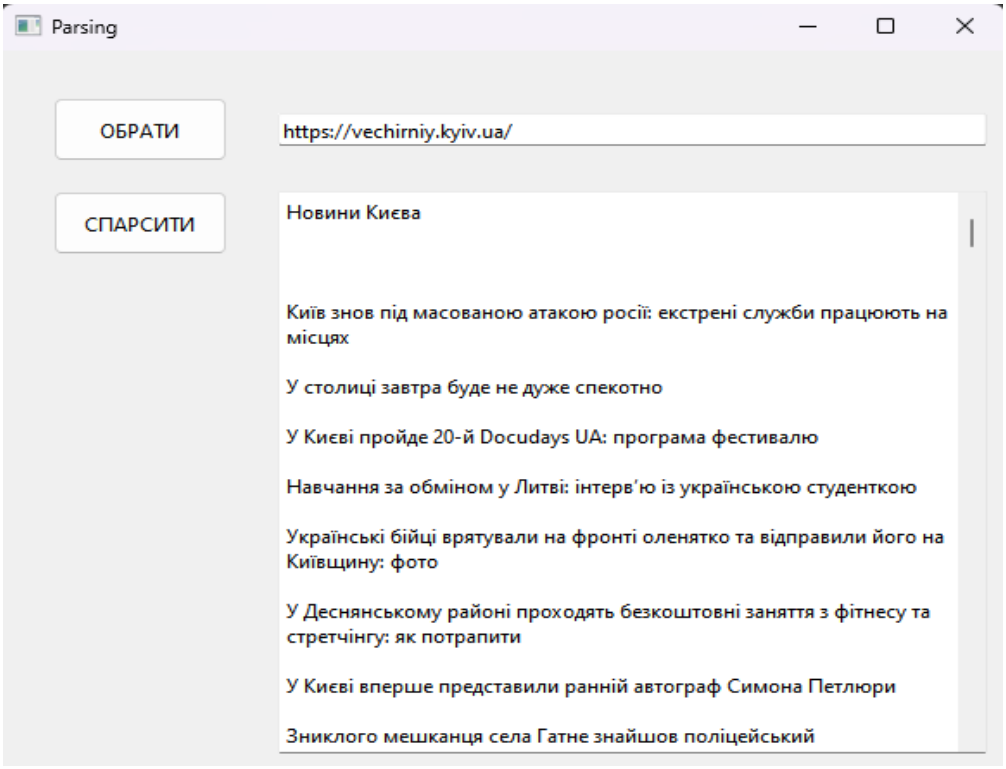


Рисунок 3.10 - Приклад виконаного парсингу

3.3.4 Області застосування програми

Програма парсера новин має широкий спектр застосувань.

Деякі з них включають:

Збір новинних статей: Програма може сканувати веб-сайти новин та витягувати заголовки, текстові відомості, дати публікацій тощо. Це може бути корисно для створення зібраної бази даних новин, аналізу тематики чи отримання актуальних новинних оновлень.

Аналіз новинного контенту: Парсер може витягувати ключові слова, теги, категорії або іншу мета-інформацію з новинних статей. Це може використовуватися для аналізу тематичних тенденцій, статистичних досліджень або побудови рекомендаційних систем.

Моніторинг новинного потоку: Програма може автоматично відстежувати та сповіщати про нові статті або оновлення на вибраних веб-сайтах новин. Це може бути корисно для актуального інформування про події у певній галузі або відстеження новин про конкретні організації чи особи.

Розробка засобів пошуку та агрегації новин: Парсер може використовуватися для розробки власних пошукових систем або новинних агрегаторів. Це дозволяє користувачам швидко знаходити новини з різних джерел та використовувати фільтри або сортування для налаштування вмісту.

Наукові дослідження: Парсери новин можуть бути використані у наукових дослідженнях, де потрібно аналізувати великі обсяги новинних даних, відстежувати події або проводити аналітику масових медіа.

Створена програма-парсер збирає актуальні дані з новинних сайтів та додає отриману у відповідну попердньо створену даних для можливої подальшої обробки.

Висновки третього розділу: перед початком роботи були ретельно проаналізовані всі вимоги замовника, а також переваги, недоліки та можливості мов програмування високого рівня. В результаті цього аналізу було вирішено працювати з використанням найбільш оптимальної для цієї мети мови програмування – Python.

В ході роботи були детально вивчені синтаксис, модулі та бібліотеки обраної мови.

У цьому розділі детально описані основні функції програми та результат її роботи. За результатами тестування парсера можна зробити висновок, що всі цілі досягнуті.

3.3.5 Тестування роботи програми

Після виконання програми виконуються записи даних у таблицю бази даних або створюються заново, якщо вони відсутні. Записи у базі даних не видаляються після чергового виконання програми, лише додаються оновлені записи, в такий спосіб маємо дані які можна надалі передавати для аналізу чи оцінки, з можливістю побудови графіків за часом.

В третьому розділі розглянуто етапи розробки СЗСАД, а саме опис технічного завдання, вибір інструментальних засобів розробки середовища та системи. Також описано вимоги до інтерфейсів, технічного та програмного забезпечення. Ведеться розробка програми для парсингу сайтів. Описуються етапи парсингу, структура роботи програми, тестування програм

ВИСНОВОК

У даній випускній кваліфікаційній роботі розглянуто розробку програми для парсингу даних з сайтів.

Перший розділ описує діяльність та організаційну структуру парсингу. Комплекс інформаційних систем включає численні програми для аналізу даних, документообігу та прогнозування результатів. Щоб підтримувати ці програми в актуальному стані, їм потрібна інформація безпосередньо з джерела. А для того, щоб її зібрати, необхідно розробити та впровадити сучасні методи пошуку та збору даних.

У зв'язку з цим, підготовлено та надано технічне завдання на розробку програми семантичного аналізу веб-сторінок.

Основні вимоги організації до розробленої програми:

отримання код-статусу сторінки для парсингу;

отримання актуальних даних про останні події в світі;

запис отриманої інформації в електронну таблицю;

формування баз даних на основі сформованої таблиці.

У другому розділі досліджено проблему семантичного розбору html-сторінок в Інтернеті. Проаналізовано та вивчено всі відомі методи пошуку та збору інформації з html-сторінок. В результаті порівняльного аналізу методів було вирішено створити парсер для інформаційних технологій.

Крім того, другий розділ містить опис усіх функціональних можливостей парсерів. Розглянуто та порівняно існуючі методи парсингу. Вивчено їх особливості, відмінні риси та переваги. У ході розробки також були враховані всі недоліки самостійного створення програми-парсера.

У третьому розділі наведено детальний опис розробленої програми. Для його написання була обрана високорівнева мова програмування Python3, яка

вирізняється синтаксичною простотою. У роботі наведено вичерпну характеристику використовуваних бібліотек і готових модулів цієї мови.

У цьому розділі ретельно проаналізовано функції використовуваної програми, наведено їх детальний опис, а використання проілюстровано малюнками.

На створення самого парсера пішло 14 днів. Мінімальні системні вимоги для роботи програми:

Процесор: Intel Core i3-2100, 2 ядра/4 потоки, тактова частота 3.1 ГГц

Обсяг оперативної пам'яті: 4 ГБ або більше (рекомендується 8 ГБ для більш продуктивної роботи)

Дискова пам'ять: 500 ГБ SATA HDD або більше (можна також розглянути використання SSD для покращення швидкості доступу до даних)

Мережевий адаптер: до 1 Гбіт/с

Наприкінці роботи проведено аналіз показника ефективності програми. Для цього парсер був протестований, щоб перевірити виконання всіх завдань. За результатами тестування з'ясувалося, що розроблений парсер працює коректно. Програма виконує всі вимоги коректно і в короткі терміни. Помилок під час роботи немає.

Таким чином, розроблена програма-парсер відповідає всім поставленим вимогам. Виходячи з цього, можна зробити висновок, що технічне завдання виконано успішно.

ПЕРЕЛІК ПОСИЛАНЬ

1. Інтернет. [Електронний ресурс]. Режим доступу: - https://ua.wikipedia.org/wiki/Інтернет
2. Big Data. [Електронний ресурс]. Режим доступу: - https://www.sas.com/en_us/insights/big-data/what-is-big-data.html
3. Web Scraping using Python. [Електронний ресурс]. Режим доступу: - https://www.datacamp.com/community/tutorials/web-scraping-using-python
4. Хохлова, Ю. Глосарій з інформаційного суспільства / Хохлова Ю.Є., Бунчук М.А. // Інститут розвитку інформаційного суспільства. – 2009. – 160 с.
5. Портал: Комп'ютерні технології. [Електронний ресурс]. Режим доступу: -http://ua.wikipedia.org/wiki/Портал:Комп'ютерні технології
6. Вікіпедія. Селеніум. [Електронний ресурс]. Режим доступу: https://ua.wikipedia.org/wiki/Selenium/
7. Інтернет: особливості та можливості. [Електронний ресурс]. Режим доступу: - https://studme.org/50396/menedzhment/internet_osobennosti_vozmozhnosti
8. Парсинг html-сайтів за допомогою PHP, Ruby. [Електронний ресурс]. Режим доступу: - http://parsing.valemak.com/ua/what-why-how/stages-of-parsing/
9. Суханов, А.А., Маратканов, А.С. Аналіз способів збору соціальних даних із мережі інтернет / Суханов, А.А., Маратканов, А.С. // International Scientific Review. 2017. Вип. 1. С.22-25
10. Python. [Електронний ресурс]. Режим доступу: - https://ua.wikipedia.org/wiki/Python
11. Requests: HTTP for Humans. [Електронний ресурс]. Режим доступу: - http://docs.python-requests.org/en/master/#

12. Beautiful Soup Documentation. [Електронний ресурс]. Режим доступу: - https://www.crummy.com/software/BeautifulSoup/bs4/doc/
13. Цхошвілі Д.З., Іванова Н.А. Приклади використання технології Парсингу / Цхошвілі Д.З., Іванова Н.А. // Стаття у збірнику праць конференції. Рік видання. 2017. С. 135-138
14. lxml - XML та HTML з Python. [Електронний ресурс]. Режим доступу: -http://lxml.de/index.html
15. Обробляємо csv файли - Модуль csv. [Електронний ресурс]. Режим доступу: - https://python-scripts.com/import-csv-python
16. ORM. [Електронний ресурс]. Режим доступу: - https://ua.wikipedia.org/wiki/ORM
17. X-Parser Light. [Електронний ресурс]. Режим доступу: - https://skladchik.com/threads/x-parser-light
18. Комп'ютерна програма. [Електронний ресурс]. Режим доступу: - https://ua.wikipedia.org/wiki/Комп'ютерна_програма
19. Business Intelligence. [Електронний ресурс]. Режим доступу: - https://ua.wikipedia.org/wiki/Business_Intelligence

Додаток А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЧНОГО ЗБИРАННЯ ТА СИНТАКСИЧНОГО АНАЛІЗУ ДАНИХ З САЙТІВ МОВОЮ PYTHON

Виконав студент 4 курсу
Групи ПД-44
Дорошин Назар Андрійович
Керівник роботи
к.ф.-м.н., доц., доцент кафедри ІПЗ Садовенко
Володимир Сергійович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи -** автоматизація збирання та синтаксичного аналізу даних з сайтів за допомогою програмного забезпечення мовою Python.
- **Об'єкт дослідження** - процес розробки програмного забезпечення для автоматичного збирання даних та синтаксичного аналізу мовою Python з веб-сайтів.
- **Предмет дослідження** - програмне забезпечення для автоматичного збирання даних з веб-сайтів та синтаксичного аналізу мовою Python.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Реалізувати алгоритми збирання даних з веб-сайтів за допомогою мови програмування Python.
2. Розробити функціонал для синтаксичного аналізу отриманих даних з використанням підходів та бібліотек Python.
3. Забезпечити збереження отриманих даних у Базах Даних
4. Виконати тестування та налагодження програмного забезпечення для забезпечення його роботи та надійності.
5. Провести аналіз розробленого програмного забезпечення шляхом збору та аналізу даних з веб-сайтів реального веб-простору.

АНАЛІЗ АНАЛОГІВ

	A-Parser	uParser	Розроблений Parser
Автоматизація	+	+	+
Синтаксичний Аналіз	-	-	+
Зберігає файли у бази даних	-	-	+
Функціонал	+	-	-
Кросс-платформенність	-	-	+

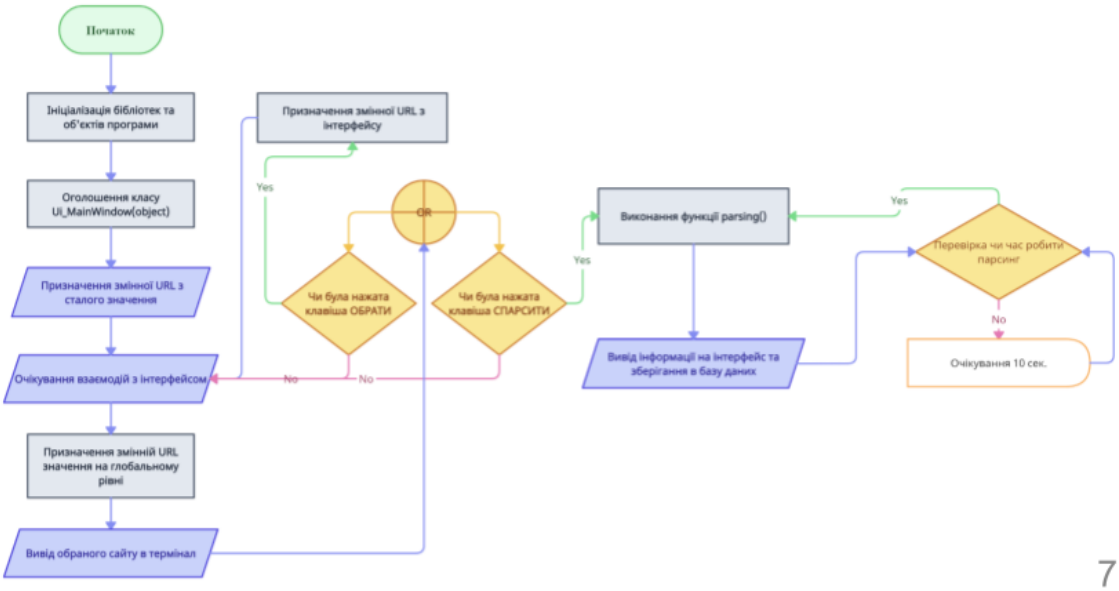
ВИМОГИ ДО ДОДАТКУ

1. Взаємодія з сайтами, отримання код-статус веб-сайту.
2. Синтаксичний аналіз веб-сайту.
3. Збирання даних з веб-сайту.
4. Пошук показників для синтаксичного аналізу.
5. Отримання актуальних даних.
6. Автоматичний запис отриманої інформації в електронну таблицю.
7. Автоматичне формування баз даних на основі отриманої інформації.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

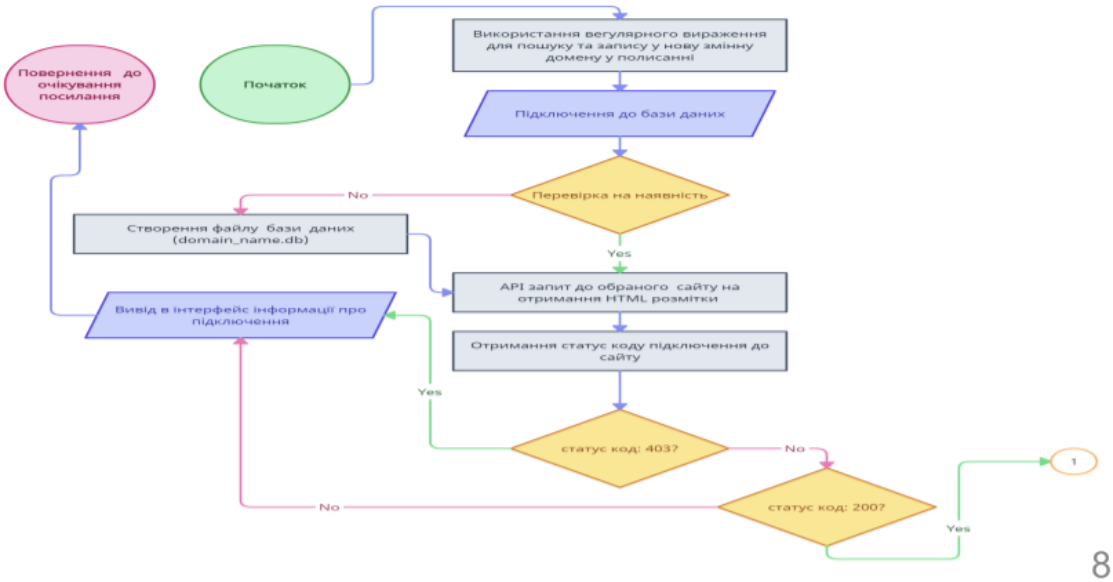
]

АЛГОРИТМ РОБОТИ ПРОГРАМИ



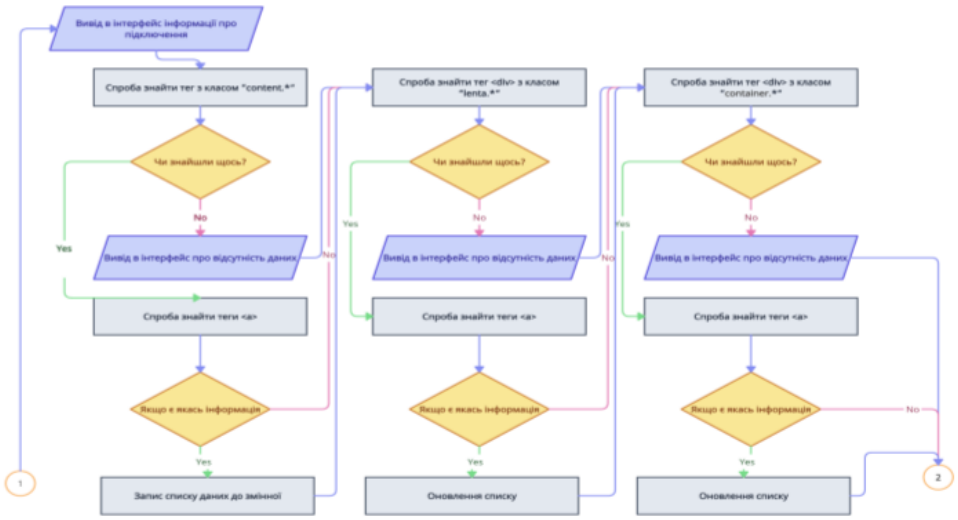
7

РОБОТА З БАЗАМИ ДАННИХ

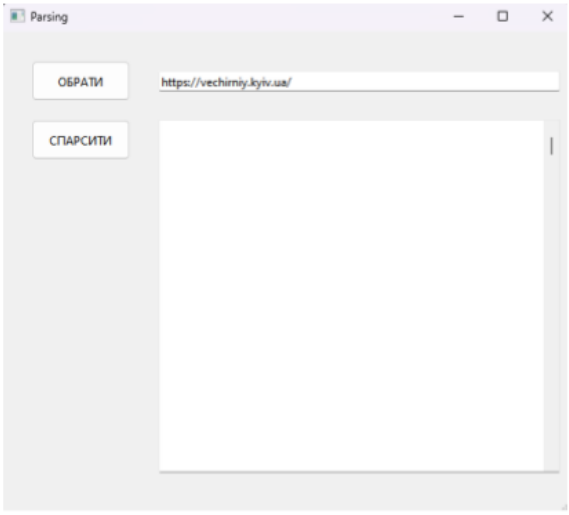


8

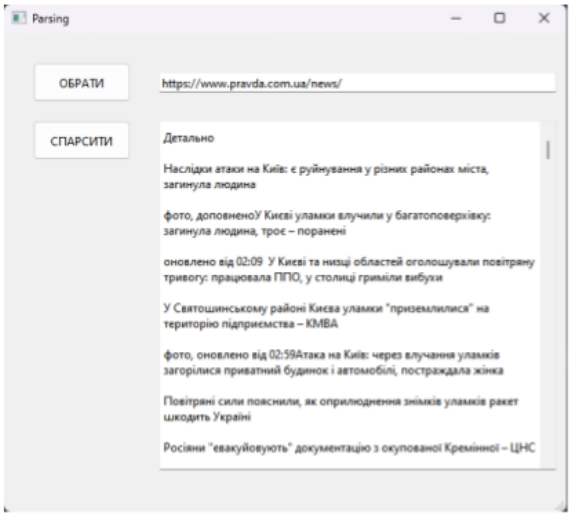
РОБОТА З ПАРСЕРОМ



ЕКРАННІ ФОРМИ

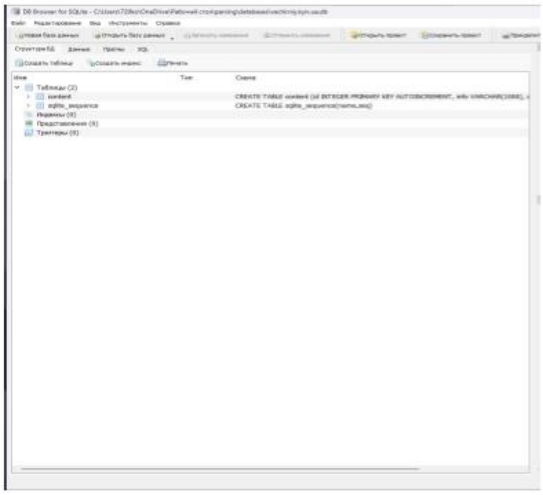


Інтерфейс

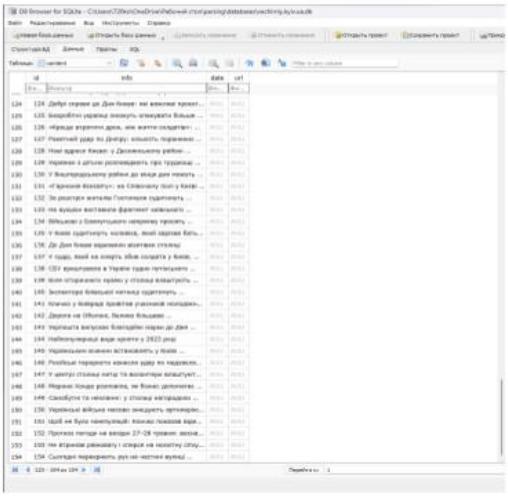


Отримана інформація

ЕКРАННІ ФОРМИ



Структура бази даних



Вміст бази даних

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Дорошин Н.А. Розробка програмного забезпечення для автоматичного збирання та синтаксичного аналізу даних з сайтів мовою Python./ Садовенко В.С., Дорошин Н.А. // "Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії", 01- 03 червня 2023р., ДУТ, м. Київ - К: ДУТ, 2023. Подано до друку.

ВИСНОВКИ

1. Реалізовано алгоритми збирання даних з веб-сайтів за допомогою мови програмування Python.
2. Розроблено функціонал для синтаксичного аналізу отриманих даних з використанням підходів та бібліотек Python.
3. Забезпечено збереження отриманих даних у відповідному форматі .db
4. Виконано тестування та налагодження програмного забезпечення для забезпечення його роботи та надійності.
5. Проведено аналіз розробленого програмного забезпечення шляхом збору та аналізу даних з веб-сайтів реального веб-простору.

ДЯКУЮ ЗА УВАГУ!

Додаток Б

Робота з базами даних та HTML запит

```
s = re.match(r'https?://(?:www\.|)([\w.-]+).*', url)
    domain = s.group(1)
    form.textEdit.setText("")
    db = sqlite3.connect(f'databases/{domain}.db')
    cursor = db.cursor()
    cursor.execute(f'''CREATE TABLE IF NOT EXISTS content
                (id INTEGER PRIMARY KEY AUTOINCREMENT,
                info VARCHAR(1000),
                date VARCHAR(100),
                url VARCHAR(200))''')
    db.commit()
    db.close()
    r = requests.get(url)
    status_code = r.status_code
    if status_code == 200:
        print(f"[{time.ctime()}] Status code connection: [200 - \"OK\"]")
    elif status_code == 403:
        print(f"[{time.ctime()}] Status code connection: [403 - \"Connection forbidden by hoster!\"]")
        raise (Exception("Not for this site!"))
    r.raise_for_status()
    html = Bs(r.text, 'lxml')
```

Робота парсера

```
try:
        main = html.find(class_=(re.compile(r"^content.*")))
```

```
            a = main.find_all('a')
            if content_value < set(list(a)):
                content = a
        except AttributeError:
            print(f"[{time.ctime()}] Parse from class \"content.*\" failed!")
        try:
            main = html.find('div', class_=(re.compile(r"^lenta.*"))

 a = main.find_all('a')
            if content_value < set(list(a)):
                content = a
        except AttributeError:
            print(f"[{time.ctime()}] Parse from tag:<div> class \"lenta\" failed!")
        try:
            main = html.find('div', class_=(re.compile(r"^container.*")))
            a = main.find_all('a')
            if content_value < set(list(a)):
                content = a
        except AttributeError:
            print(f"[{time.ctime()}] Parse from tag:<div> class \"container\" failed!")
```

Відбір нових дописів

```
    db = sqlite3.connect(f'databases/{domain}.db')
        cursor = db.cursor()
        for i in content:
            try:
                res = cursor.execute(f"SELECT info FROM content WHERE
info=\'{i.text}\'")
```

```
                out = res.fetchone()
                if out is None:
                    print(f"[{time.ctime()}] New: [{i.text}]")
                    cl_new += 1
                    form.textEdit.setText(form.textEdit.toPlainText() + i.text + '\n\n')
                    cursor.execute(
                        f"INSERT INTO content (\"info\", \"date\") VALUES (\'{i.text}\', \'{time.ctime()}\')")
                    db.commit()
            except sqlite3.OperationalError:
                print(f"[{time.ctime()}] Error Syntax from: {i.text}")
        db.close()
        if cl_new == 0:
            form.textEdit.setText(form.textEdit.toPlainText() + "No new information from this site." + '\n\n')
        cl = len(content)
        print(f"[{time.ctime()}] {cl_new} new records was added to DataBase {domain}.db")
        print(f"[{time.ctime()}] {cl} of <a>Text<a> parsed from {domain}
```