Олена Олександрівна ГРИБ'ЮК[1]**,**
доцент, старший дослідник,
ORCID: https://orcid.org/0000-0003-3402-0520
Микола Іванович ПРИЛУЦЬКИЙ[1]**,**
студент 2 курсу, групи К-41м,
ORCID: https://orcid.org/0009-0005-0141-5202

[1] *Заклад вищої освіти «Міжнародний науково-технічний університет імені академіка Юрія Бугая»*

Прийняття: 02/05/2026
Рецензія: 16/05/2026
Публікація: 09/07/2026

УДК
004.9; 53.08;
681.004.02

ISSN 2786-7226 (print)
ISSN 2786-7234 (online)

DOI: https://doi.org/10.53920/ITS-2026-1-4

СПЕЦИФІКА ПРОЕКТУВАННЯ ТА РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПАРСИНГУ ДАНИХ

У дослідженні запропоновано та реалізовано підхід щодо опрацювання зображень графіків відключень на основі OCR із попередньою обробкою зображень і постобробкою результатів розпізнавання (виправлення типових помилок, відновлення часу, перевірка допустимих діапазонів), що підвищує придатність результатів до подальшого збереження і використання у сповіщеннях. Здійснено проєктування модульної архітектури системи з інтеграцією компонентів через централізовану базу даних PostgreSQL, а також реалізовано конвеєр опрацювання даних, що забезпечує уніфікацію дат і часових інтервалів, контроль цілісності даних та запобігання дублюванню записів при повторних запусках і регулярному моніторингу. Окрему увагу приділено забезпеченню надійності та цілісності транзакцій. Для цього в класі реалізовано механізм контекстних менеджерів, що гарантує автоматичний відкат будь-яких змін у разі виникнення помилок під час виконання операцій. Кожен метод класу передбачає попередню перевірку активності підключення, що запобігає збоям у роботі асинхронних парсерів та бота при тимчасових втратах зв'язку з сервером БД. Використання такої архітектури дозволяє зосередити всю логіку маніпулювання даними в одному модулі, спрощуючи тестування та подальший розвиток системи. Практичне значення роботи полягає у створенні інформаційної системи парсингу даних, який зменшує обсяг ручної перевірки джерел, забезпечує актуалізацію даних у режимі моніторингу та надає користувачам зручний доступ до результатів і спо-

віщень. Перспективи подальшого розвитку інформаційної системи включають розширення переліку джерел, підвищення точності OCR за рахунок адаптивної сегментації та додаткових правил нормалізації, удосконалення стійкості до змін форматів публікацій, а також розширення метрик якості й моніторингу продуктивності в реальних умовах експлуатації.

Ключові слова: *інформаційна система, веб-скрапінг, парсинг, OCR, Telegram-бот, PostgreSQL, моніторинг даних, персоналізовані сповіщення.*

Olena HRYBIUK, Mykola PRYLUTSKYI

SPECIFICS OF THE DESIGN AND DEVELOPMENT OF AN INFORMATION SYSTEM FOR DATA PARSING

This study proposes and implements an approach to processing images of power cut schedules based on OCR, involving pre-processing of images and post-processing of recognition results (correction of typical errors, time restoration, verification of valid ranges), which improves the suitability of the results for subsequent storage and use in notifications. A modular system architecture has been designed, with components integrated via a centralised PostgreSQL database; a data processing pipeline has also been implemented, ensuring the standardisation of dates and time intervals, data integrity checks, and the prevention of duplicate records during repeated runs and regular monitoring. Particular attention has been paid to ensuring the reliability and integrity of transactions. To this end, the class implements a context manager mechanism, which guarantees the automatic rollback of any changes should errors occur during the execution of operations. Each method of the class performs a preliminary check on the connection status, which prevents disruptions to the operation of asynchronous parsers and the bot in the event of temporary loss of connection to the database server. Using this architecture allows all data manipulation logic to be centralised in a single module, simplifying testing and the further development of the system. The practical significance of this work lies in the creation of a data-parsing information system that reduces the need for manual verification of sources, ensures data is kept up to date during monitoring, and provides users with convenient access to results and alerts. Prospects for the further development of the information system include expanding the list of sources, improving OCR accuracy through adaptive segmentation and additional normalisation rules, enhancing resilience to changes in publication formats, and expanding quality metrics and performance monitoring under real-world operating conditions.

Keywords: *information system, web scraping, parsing, OCR, Telegram bot, PostgreSQL, data monitoring, personalised alerts.*

Постановка проблеми. Веб-скрапінг розглядається у контексті сукупності методів і процесів автоматизованого отримання інформації з веб-ресурсів, причому включається доступ до ресурсу, отримання контенту, його аналіз та перетворення у структурований вигляд. У задачах веб-скрапінгу парсинг є ключовим етапом після отримання контенту. Залежно від джерела використовують структурний парсинг, шаблонні підходи, а також методи, орієнтовані на неструктуровані дані [1]. Для забезпечення підтримуваності парсерів важливо обирати селектори, які найменше залежать від нестабільних частин верстки.

Веб-дані передаються у рамках клієнт-серверної взаємодії. Клієнт формує запит, сервер повертає відповідь зі статус-кодом, заголовками і тілом. На практиці для автоматизованого доступу мають значення: коректна обробка редиректів, лімітів частоти звернень та помилок доступу, а також стабільне відтворення параметрів запиту, що впливають на контент [2]. Ці аспекти визначають надійність накопичення даних і впливають на добір методу. Для коректного вилучення та подальшої нормалізації важливо враховувати кодування і локалізовані формати дат, часу та числових значень. Робота зі структурованими форматами також спрощує контроль якості: перевірку типів, обов'язкових полів, діапазонів значень, а також узгодження даних між різними джерелами [3].

Аналіз останніх досліджень і публікацій. Типовий OCR-процес включає підготовку зображення, розпізнавання тексту та постопрацювання даних. Ефективність OCR доцільно оцінювати не лише «якістю тексту», а і якістю витягнутих сутностей, наприклад точністю визначення дат/інтервалів часу та стабільністю результатів для різних якостей зображень [4]. Для задачі збору графіків та оголошень доцільним є комбінований підхід: використовувати API там, де це можливо; застосовувати статичний парсинг для простих HTML-джерел; переходити до динамічного збору лише за необхідності; використовувати OCR для інформації у вигляді зображень; а також організовувати моніторинг як процес із відстеженням новизни даних [5].

Антибот-захист є поширеною перешкодою для автоматизованого збору. Практики подолання повинні бути «м'якими»: раціональна частота звернень, використання кешування, вибір альтернативних джерел, а також організація моніторингу без зайвих повторних запитів [6]. У таких випадках можливими стратегіями є використання браузерної автоматизації або аналіз мережевих запитів для отримання даних зі структурованих відповідей. Вибір стратегії визначається компромісом між стабільністю, швидкодією та складністю супроводу. Оскільки формати джерел змінюються, важливо проектувати збір даних як підтримуваний процес: забезпечувати логування, перевірку коректності результатів, відновлення після

збоїв і контроль дублікатів. Для масштабування застосовують розділення задач збору та обробки, фонове виконання й планування запусків, що дозволяє підтримувати стабільність системи при зростанні кількості джерел і користувачів [7]. У рамках дослідження проаналізовано предметну область [8] і визначено ключові проблеми автоматизованого збору даних із гетерогенних інтернет-джерел [9], зокрема різні формати подання [10], нестабільність верстки [11], наявність неструктурованих даних і потребу в нормалізації для подальшого пошуку і сповіщень [12]**.**

Мета статті полягає в удосконаленні опрацювання даних шляхом формалізації опису технологій та розробленні надійної інформаційної системи парсингу даних, яка зменшує частку ручної роботи, забезпечує актуальність і якість даних після нормалізації та надає користувачеві доступ до релевантної інформації у зручному форматі з налаштовуваними сповіщеннями.

Виклад основного матеріалу дослідження. Інформаційна система парсингу даних базується на модульній архітектурі, що забезпечує необхідну гнучкість, масштабованість та зручність у підтримці програмного комплексу. Проектування системи передбачає розподіл на п'ять функціональних компонентів, взаємодія між якими реалізується через централізовану реляційну базу даних PostgreSQL. Таке рішення дозволяє забезпечити цілісність інформаційних потоків та незалежність роботи окремих модулів. До основних архітектурних компонентів системи належать (рис. 1): модуль текстового парсингу, що здійснює аналіз HTML-сторінок та вилучення графіків відключень електроенергії, представлених у текстовому форматі; модуль OCR-парсингу, який використовує технології комп'ютерного зору для обробки зображень із графіками, отриманих із Telegram-каналів; модуль збору оголошень, що виконує скрапінг веб-сторінок маркетплейсу для акумуляції даних про товари за заданими категоріями; модуль Telegram-бота, який виступає інтерактивним інтерфейсом, що забезпечує реєстрацію користувачів, налаштування фільтрів та візуалізацію даних; модуль взаємодії з БД, що інкапсулює логіку CRUD-операцій та гарантує стабільне підключення всіх модулів до спільного сховища.

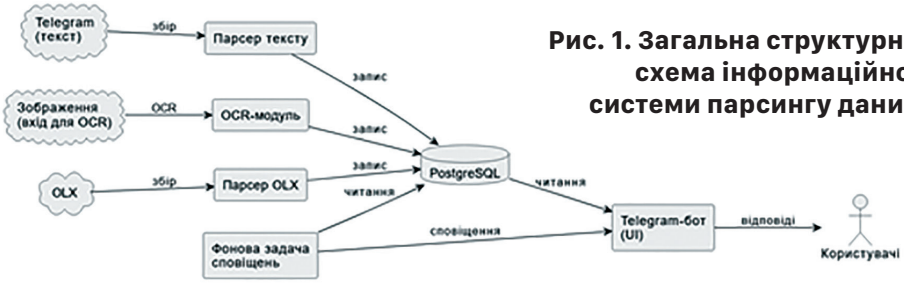

Рис. 1. Загальна структурна схема інформаційної системи парсингу даних

Потоки даних у системі розділені на три ключові вектори, що працюють паралельно в асинхронному режимі (рис. 2). **Потік збору та акумуляції:** парсери незалежно один від одного звертаються до зовнішніх джерел (веб-сайти, месенджери), структурують отриману інформацію та оновлюють записи у базі даних. **Потік обробки запитів:** бот ініціює запити до БД на основі команд користувача, забезпечуючи миттєвий доступ до актуальних графіків або оголошень. **Потік персоналізованих оповіщень:** фонові задачі періодично порівнюють нові записи у базі з налаштуваннями профілів користувачів і автоматично розсилають сповіщення у разі збігу критеріїв. Асинхронна архітектура системи дозволяє одночасно обробляти декілька джерел даних та запитів від великої кількості користувачів без блокування основних обчислювальних процесів. На етапі ініціалізації при запуску кожен модуль ініціалізуватиме підключення до бази даних, завантажуватиме необхідні конфігурації та виконуватиме початкову перевірку джерел даних. На етапі збору даних парсери періодично (кожну хвилину або за іншим розкладом) перевірятимуть свої джерела даних, а при виявленні нових даних вони виконуватимуть парсинг, обробку та збереження результатів у базі даних. На етапі зберігання всі зібрані дані зберігатимуться в централізованій базі даних PostgreSQL з використанням нормалізованої схеми, що забезпечить цілісність даних та ефективність запитів. На етапі обробки запитів користувачів, коли користувач взаємодіятиме з Telegram-ботом, бот виконуватиме запити до бази даних для отримання актуальної інформації та формуватиме відповіді з урахуванням налаштувань користувача. На етапі автоматичних оповіщень фонові задачі бота регулярно перевірятимуть базу даних на наявність нових даних, які відповідають критеріям користувачів (регіон, черга, категорія, ціна тощо), та відправлятимуть персоналізовані оповіщення. На етапі відображення даних користувачі отримуватимуть інформацію у зручному форматі через інтерфейс Telegram-бота з можливістю навігації через меню та фільтрації даних.

Розроблена архітектура забезпечує надійність системи, оскільки збій одного компонента не впливатиме на роботу інших, а централізоване зберігання даних гарантуватиме їхню доступність для всіх модулів. Для забезпечення ефективного зберігання та обробки даних, зібраних парсерами, проаналізовано та уточнено вимоги до структури бази даних (рис. 3). Система має зберігати інформацію про відключення електроенергії, оголошення з маркетплейсу OLX, налаштування користувачів та довідкову інформацію.

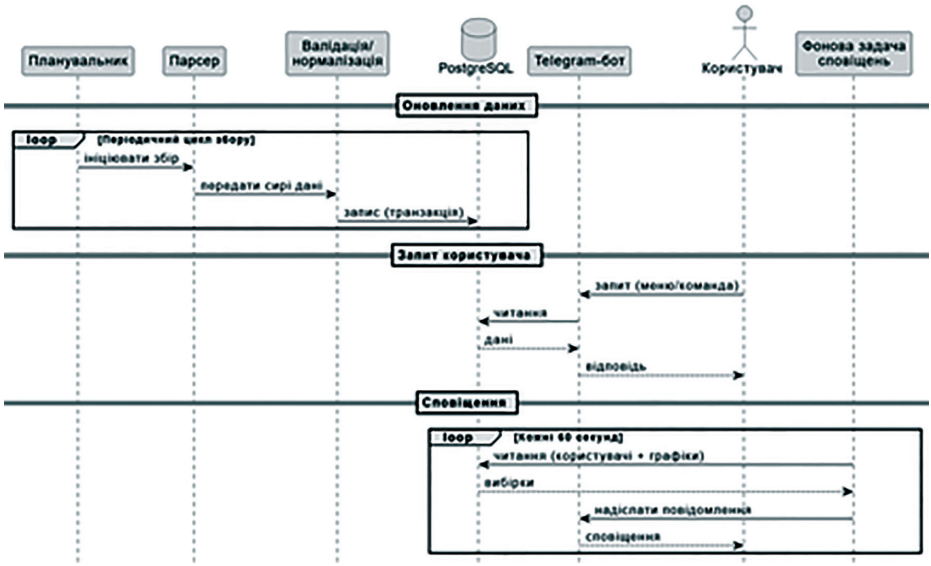

Рис. 2. Послідовність обробки оновлень інформаційної системи парсингу даних

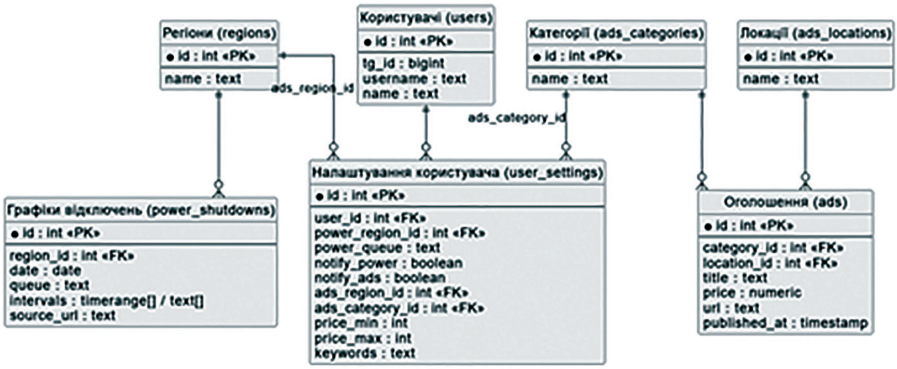

Рис. 3. ER-діаграма БД інформаційної системи парсингу даних

Перша вимога стосується зберігання графіків відключення електроенергії, де система має зберігати інформацію про регіони, дати відключення, часові інтервали відключення для різних черг, джерела інформації та іден-

тифікатори постів з Telegram каналів. Друга вимога полягає у зберіганні оголошень, де необхідно зберігати дані про оголошення з OLX, включаючи заголовок, опис, ціну, локацію, зображення, посилання та категорію, а також відстежувати час створення запису для визначення нових оголошень. Третя вимога стосується управління користувачами, де система має зберігати налаштування користувачів Telegram-бота, включаючи вибір регіону та черги для оповіщень про відключення світла, налаштування фільтрів для оголошень (категорія, регіон, ціновий діапазон), мову інтерфейсу та прапорці керування оповіщеннями. Четверта вимога полягає у зберіганні довідкової інформації, де необхідно зберігати список регіонів та категорій для парсингу, що дозволить гнучко налаштовувати систему без зміни коду. П'ята вимога стосується оптимізації запитів, де база даних має забезпечувати швидкий пошук даних за різними критеріями, тому необхідно передбачити створення індексів для часто використовуваних полів.

Логічна схема бази даних реалізована у вигляді окремого SQL-сценарію, що дозволяє забезпечити портативність системи та легкість її ініціалізації на нових серверах. Архітектура сховища базується на п'яти основних таблицях, взаємопов'язаних через систему зовнішніх ключів (foreign keys), що гарантує цілісність даних на рівні СУБД. Кожна сутність використовує автоматичну генерацію унікальних ідентифікаторів через послідовності (sequences). **Таблиця regions** виступає базовим довідником системи та зберігає унікальні назви регіонів України та URL-адреси офіційних джерел інформації, що дозволяє іншим модулям системи коректно ідентифікувати локації. **Таблиця power_shutdowns** призначена для акумуляції графіків відключень. Окрім зв’язку з регіоном, вона містить масиви часових інтервалів (shutdown_intervals), дату та номер черги. Унікальність записів контролюється комбінацією ідентифікатора поста, черги та дати, що запобігає дублюванню інформації при повторному скануванні джерел. **Таблиця categories** зберігає параметри пошуку для модуля парсингу OLX. Вона включає назву категорії, текст пошукового запиту, цінові обмеження та масив ключових слів, що дозволяє гнучко змінювати пріоритети збору даних без втручання в програмний код. **Таблиця ads** містить детальну інформацію про зібрані оголошення. Для кожного запису зберігається унікальний item_id джерела, посилання на зображення, опис, ціна та локація. Таблиця дозволяє відстежувати час появи нових об'єктів для оперативного інформування користувачів. **Таблиця users** акумулює складні налаштування користувачів Telegram-бота. Вона зберігає ідентифікатори регіонів та черг для енергооповіщень, параметри фільтрації оголошень (категорія, ціна, ключові слова), мову інтерфейсу та прапорці активності сповіщень. Для досягнення високої продуктивності системи та забезпечення оптимальної

структури збереження інформації було впроваджено використання складних типів даних та багаторівневу стратегію індексування. Застосування розширених можливостей СУБД PostgreSQL дозволяє не лише забезпечити нормалізацію бази даних, а й значно спростити логіку обробки об'єктів безпосередньо на рівні сховища.

Ключовим рішенням для структурованого зберігання графіків відключень стало впровадження користувацького складеного типу даних (composite type) time_range. Для забезпечення швидкого доступу до даних та мінімізації часу виконання пошукових операцій розроблено систему індексів для ключових полів. Окрему увагу приділено реалізації механізмів нечіткого пошуку локацій. Через специфіку користувацького контенту на маркетплейсах (можливі друкарські помилки, різні варіанти написання назв населених пунктів) стандартні методи порівняння рядків є недостатніми. Для вирішення цієї проблеми в системі активовано розширення pg_trgm та створено спеціалізований GIN-індекс для поля location, що дозволяє виконувати пошук на основі триграмного аналізу тексту, забезпечуючи високу релевантність результатів навіть при неточному співпадінні запиту користувача з даними в базі. Для забезпечення високого рівня інкапсуляції логіки взаємодії з реляційним сховищем та абстрагування від низькорівневих SQL-запитів у системі реалізовано спеціалізований клас Database. Даний компонент виступає програмним інтерфейсом для всіх інших модулів системи, приховуючи деталі реалізації запитів та забезпечуючи уніфікований доступ до даних.

Функціональний склад класу Database структурований за предметними областями та охоплює наступні групи методів: **управління з'єднанням:** методи connect та close для ініціалізації та коректного завершення сесій зв'язку з базою даних. **Обробка даних про енергопостачання:** методи додавання графіків (add_power_shutdown), перевірки їх існування за унікальними ознаками (ідентифікатор поста, черга, дата) та вибірки за різними критеріями (регіон, номер черги, актуальна дата). Окремий метод get_latest_power_shutdown_post_id дозволяє системі синхронізувати процес збору даних, уникаючи повторної обробки вже парсених повідомлень. **Моніторинг ринку оголошень:** комплекс методів для реєстрації нових товарних позицій (add_ad), вилучення актуальних оголошень за заданий період та реалізації нечіткого пошуку локацій (find_ads_locations_by_similarity), що базується на триграмному аналізі тексту. **Адміністрування користувачів та налаштувань:** методи для реєстрації профілів (create_or_update_user), отримання даних користувача за Telegram ID та гнучкого оновлення персональних параметрів: мови інтерфейсу, часових інтервалів сповіщень,

цінових діапазонів та статусів активності оповіщень. **Робота з довідниками:** методи для отримання повних переліків регіонів та категорій (get_all_regions, get_all_categories), що забезпечує динамічне формування меню в інтерфейсі бота. Для забезпечення високого ступеня уніфікації логіки функціонування різних модулів збору даних та дотримання принципів чистого коду (DRY - Don't Repeat Yourself), у системі спроектовано абстрактний базовий клас BasePowerOffParser. Даний компонент виступає фундаментом для побудови спеціалізованих парсерів, визначаючи спільний інтерфейс та базовий функціонал, необхідний для обробки інформації про енергопостачання незалежно від формату джерела (текстового чи графічного). Програмна реалізація базового класу базується на використанні вбудованого модуля abc (Abstract Base Classes) мови програмування Python. Це дозволяє визначити жорсткий контракт для всіх похідних класів: будь-який парсер, що наслідує BasePowerOffParser, зобов'язаний реалізувати ключові абстрактні методи, інакше інтерпретатор не дозволить створити екземпляр такого класу. Такий підхід гарантує архітектурну узгодженість системи та дозволяє уникнути хаотичного дублювання процедур підключення до бази даних або валідації регіонів у кожному окремому модулі.

Конструктор класу (__init__) виконує критично важливу роль у життєвому циклі парсера, ініціалізуючи основні робочі атрибути (рис. 4): **db (екземпляр класу Database):** централізований об'єкт для взаємодії з реляційним сховищем, що дозволяє парсеру зберігати результати своєї роботи безпосередньо в PostgreSQL; **region_id:** числовий ідентифікатор регіону, який парсер обслуговує. Це дозволяє системі чітко розмежовувати дані, наприклад, між Київською та Дніпропетровською областями ще на етапі збору; **last_post:** ідентифікатор останнього успішно обробленого повідомлення. Використання цього атрибута дозволяє реалізувати механізм інкрементального парсингу, коли система автоматично ігнорує застарілі дані та починає роботу лише з новими публікаціями у Telegram-каналах чи на веб-сайтах.

Крім ініціалізації атрибутів, конструктор класу бере на себе відповідальність за перевірку стану з'єднання з базою даних та автоматичну конфігурацію параметрів підключення. Таке централізоване управління ресурсами дозволяє значно спростити код похідних класів, зосередивши увагу виключно на специфіці витягування даних (регулярні вирази для тексту або алгоритми OCR для зображень). Впровадження BasePowerOffParser створює надійну абстракцію, яка робить систему стійкою до розширення: для додавання нового джерела графіків відключень достатньо створити невеликий клас-нащадок, не змінюючи при цьому ядро системи.

Рис. 4.
UML діаграма класів парсерів

Для забезпечення стабільної роботи парсерів у базовому класі реалізовано механізми захисту від збоїв та валідації даних. Це мінімізує ризик аварійної зупинки системи при помилках у джерелах або проблемах із мережею. Основні інструменти відмовостійкості включають: контроль з'єднання (перед кожною операцією з БД перевіряється активність підключення; якщо зв'язок розірвано, метод перериває виконання з логуванням попередження, не зупиняючи весь процес); транзакційну безпеку (використання блоків try-except у методі save_power_shutdown_to_db() гарантує виконання rollback при будь-якій помилці запису, що запобігає появі в базі часткових або пошкоджених графіків); управління конкурентністю (перед перевіркою на дублікати виконується попередній rollback поточної транзакції, що усуває конфлікти при одночасній роботі кількох парсерів та забезпечує актуальність результатів перевірки); валідацію типів (обов'язкові поля (region, date) проходять перевірку перед збереженням, а формат дати верифікується через datetime.strptime(), що відсікає некоректні записи на ранньому етапі); очищення ресурсів (метод close() забезпечує коректне розірвання сесій із PostgreSQL, запобігаючи витоку пам'яті та переповненню пулу з'єднань).

Модуль текстового парсингу призначений для автоматизації збору та структурування даних про обмеження енергопостачання, які оприлюднюються у форматі текстових повідомлень. Основним джерелом інформації

виступає офіційний Telegram-канал Черкаської обласної державної адміністрації, де публікуються актуальні графіки для обласного центру та регіону. Аналіз вхідних даних показав, що публікації мають сталу семантичну структуру, яка включає наступні елементи: хронологічні мітки; ідентифікатори споживачів (перелік черг, що позначаються цифровим індексом; часові інтервали (конкретні періоди відключень).

Програмна реалізація модуля базується на принципі наслідування від базового класу системи. Це дозволяє використовувати вже готову логіку взаємодії з реляційною базою даних та існуючі алгоритми пошуку публікацій. Для вилучення контенту з Telegram-каналу використовується поєднання бібліотек BeautifulSoup та requests. Цей стек технологій дозволяє здійснювати збір даних через веб-інтерфейс месенджера, що є ефективною альтернативою використанню офіційного API в задачах, де не потрібна авторизація. Процес доступу до даних базується на формуванні спеціалізованих URL-запитів. Telegram надає можливість отримати спрощену HTML-версію будь-якої публікації за шаблоном https://t.me/channel_name/post_id?embed=1. Використання параметра embed=1 є ключовим технічним рішенням, оскільки воно дозволяє отримати «чисте» вбудовування поста без зайвих скриптів та елементів навігації, що значно прискорює подальшу обробку.

Алгоритм роботи основного методу парсингу починається з ініціалізації HTTP-сесії на базі об'єкта requests.Session(), що дає змогу повторно використовувати TCP-з'єднання та зменшити мережеві накладні витрати. Після отримання HTML-вмісту сторінки він передається до бібліотеки BeautifulSoup із використанням парсера html.parser, у результаті чого формується об'єктна модель документа (DOM), зручна для подальшої навігації. На наступному етапі здійснюється локалізація основного контенту повідомлення шляхом пошуку HTML-елемента з CSS-класом tgme_widget_message_text, у якому міститься текстовий опис графіка відключень електропостачання. Завершальним кроком є екстракція тексту за допомогою методу get_text() із явним збереженням символів переходу на новий рядок, що є необхідною умовою для коректної роботи регулярних виразів на етапі подальшого семантичного аналізу.Використання такого підходу забезпечує стабільний доступ до публічної інформації каналу та дозволяє уникнути складних процедур реєстрації додатків у Telegram API, спрощуючи архітектуру модуля збору даних. Водночас недоліком даного підходу є залежність від поточної HTML-структури веб-інтерфейсу Telegram, яка може змінюватися з часом, що потребує періодичного коригування використовуваних CSS-селекторів та регулярних виразів.

Програмна генерація динамічних посилань для пошуку. Процес збору даних ініціюється методом generate_url(), який відповідає за формуван-

ня коректної URL-адреси пошукового запиту з урахуванням специфічних фільтрів та параметрів обраної категорії. Алгоритм починається з вилучення вхідних параметрів з об'єкта цільової категорії, зокрема текстового запиту (query), а також мінімальних і максимальних цінових порогів. Для забезпечення технічної стійкості посилання та коректної обробки кириличних символів, пробілів чи спеціальних знаків, пошуковий запит проходить процедуру безпечного кодування через метод urllib.parse.quote(). Це перетворює «сирий» текст у формат, придатний для передачі в HTTP-запиті, після чого він інтегрується в базову адресу ресурсу за шаблоном.

Завершальним етапом формування запиту є додавання технічних параметрів фільтрації та сортування, що дозволяють системі фокусуватися на найбільш актуальних пропозиціях. Використання ключа search[order]=created_at:desc забезпечує отримання списку оголошень, відсортованих за часом їх публікації у зворотному хронологічному порядку, що є критично важливим для оперативного моніторингу нових лотів. Одночасно з цим у рядок запиту впроваджуються фільтри цінового діапазону search[filter_float_price:from] та search[filter_float_price:to], які відсікають нерелевантні пропозиції ще на рівні серверної відповіді маркетплейсу. Сформований таким чином URL є точним інструментом вибірки, який мінімізує обсяг надлишкового трафіку та дозволяє парсеру працювати виключно з цільовим сегментом ринку. Для отримання актуального масиву даних із результатів пошуку використовується асинхронний метод get_new_posts(), який виконує роль первинного фільтра та збирача посилань. Процес розпочинається з ініціалізації HTTP GET-запиту до раніше згенерованої адреси. Архітектура модуля передбачає суворий контроль статусів відповіді: у разі отримання коду, відмінного від 200 або 201 (що зазвичай свідчить про блокування або технічні роботи на сервері), система переходить у режим тривалого очікування тривалістю 6000 секунд. Така значна пауза є вимушеним заходом для запобігання остаточному блокуванню IP-адреси парсера з боку анти-фрод систем маркетплейсу.

Після успішного завантаження контенту отриманий HTML-код передається на обробку бібліотеці BeautifulSoup із використанням стандартного парсера html.parser. Алгоритм навігації по DOM-дереву сторінки фокусується на головному контейнері listing-grid, усередині якого ідентифікуються всі дочірні вузли l-card. Кожен такий вузол представляє окрему картку оголошення та піддається точковій декомпозиції. Зокрема, система вилучає унікальний ідентифікатор item_id, пряме посилання на сторінку об'єкта ad_url, а також текстові метадані: заголовок, ціну та дату публікації.

Програмна реалізація розбору карток побудована на принципах від-

мовостійкості: будь-яка помилка при екстракції даних із конкретного оголошення фіксується в логах, але не переривається цикл обробки всього списку. Це дозволяє парсеру ігнорувати некоректно заповнені або спотворені рекламні блоки, збираючи максимум доступної інформації за один ітераційний цикл. Такий підхід забезпечує стабільну наповнюваність бази даних навіть при часткових змінах у верстці окремих елементів сайту. На тестовій збірці OCR (30 зображень) отримано 437 еталонних інтервалів, зафіксовано 4 пропуски (FN) та 2 зайвих спрацювання (FP), що підтверджує працездатність підходу та визначає напрямки покращення для складних випадків розпізнавання. Узагальненням результатів є 95% успішності автоматизованих перевірок (76/80) та високі показники якості OCR на рівні інтервалів: повнота (recall) ≈ 99.1% і точність (precision) ≈ 99.5% (загальна помилка ≈ 1.37%). За результатами бенчмарків середній час OCR-обробки одного зображення графіка становить близько 1.63 с/зображення при використанні GPU (CUDA) та близько 8.66 с/зображення при виконанні на CPU без CUDA (на наборі з 30 зображень); середній час повного циклу обробки одного поста з графіком (Telegram + download + OCR) становить близько 2.65 с/пост. На підставі аналізу резудьтатів тестування можна зробити висновок, що розроблена іфнормаційна система парсингу даних демонструє високу точність розпізнавання (повнота recall >99% на рівні інтервалів). Обсяг тестового набору (437 інтервалів) є достатнім для первинної оцінки стабільності алгоритму в умовах реальних даних. Використання GPU (CUDA) суттєво підвищує швидкодію OCR: середній час OCR-конвеєра зменшується приблизно у 5.3 раза (8.66/1.63). Помилки, що виникають під час обробки окремих користувачів або оголошень, логуються та не зупиняють виконання задачі. Запуск check_and_send_ads_notifications() здійснюється у функції main() через asyncio.create_task(), що забезпечує її паралельну роботу з основним циклом Telegram-бота.

Висновки та пропозиції. У рамках дослідження розроблено інформаційну систему для автоматизації накопичення даних про графіки відключення електроенергії та моніторингу нових оголошень на маркетплейсі з наданням результатів через Telegram-бота і підтримкою персоналізованих сповіщень. Спроєктовано архітектуру системи та реалізовано базу даних із п'яти таблиць: regions, power_shutdowns, categories, ads, users. Реалізовано модулі парсингу: текстовий парсинг графіків відключень на основі BeautifulSoup і регулярних виразів; OCR-парсинг зображень графіків ДТЕК із Telegram-каналу з використанням Telethon, OpenCV та EasyOCR; модуль збору оголошень з OLX із формуванням динамічних URL, витягуванням даних зі сторінок і фільтрацією. Також розроблено Telegram-бота на базі aiogram, що надає користувачам доступ до графіків відключень і нових

оголошень, підтримує налаштування персоналізованих оповіщень та керування обліковим записом. Фонові задачі перевіряють дані в базі та надсилають сповіщення відповідно до налаштувань користувачів.

Реалізовано механізми персоналізованих сповіщень і доступу до даних через Telegram-бота, зокрема налаштування параметрів користувача (регіон/черга, критерії оголошень, часові параметри), фонові задачі перевірки оновлень та фільтрацію оголошень за умовами користувача. Проведено успішне тестування розроблених модулів інформаційної системи. Практичне значення роботи полягає у створенні готового інструменту, який зменшує обсяг ручної перевірки джерел, забезпечує актуалізацію даних у режимі моніторингу та надає користувачам зручний доступ до результатів і сповіщень. Перспективи подальшого розвитку системи включають розширення переліку джерел, підвищення точності OCR за рахунок адаптивної сегментації та додаткових правил нормалізації, удосконалення стійкості до змін форматів публікацій, а також розширення метрик якості й моніторингу продуктивності в реальних умовах експлуатації.

ЛІТЕРАТУРА

1. Mitchell R. Web Scraping with Python: Collecting More Data from the Modern Web. 2nd ed. Sebastopol: O'Reilly Media, 2018.

2. Rao N. V., et al. Optical Character Recognition Technique Algorithms // Journal of Theoretical and Applied Information Technology. 2016. Vol. 83, No. 2.

3. Hamad K., Kaya M. A Detailed Analysis of Optical Character Recognition Technology // International Journal of Applied Mathematics Electronics and Computers. 2016. Special Issue 1. P. 244–249.

4. Hrybiuk O. CleverCOMSRL Intelligent Expert System Knowledge Representation Model Features and Inconsistency Resolution Capabilities. In: Machado J., Trojanowska J., Soares F., Rea P., Butdee S., Gramescu B. (eds) Innovations in Mechatronics Engineering IV. icieng 2025. Lecture Notes in Mechanical Engineering. Springer, Cham. 2025. P. 57-68. https://doi.org/10.1007/978-3-031-94223-5_6

5. Ortega J. M. Mastering Python for Networking and Security. Birmingham: Packt Publishing, 2018.

6. Sweigart A. Automate the Boring Stuff with Python. San Francisco: No Starch Press, 2025.

7. Han J., Pei J., Tong H. Data Mining: Concepts and Techniques. San Francisco: Morgan Kaufmann, 2022.

8. Følstad A., Brandtzæg P. B. Chatbots and the New World of HCI // Interactions. 2017. Vol. 24, No. 4. P. 38–42.

9. Ferrara E., Varol O., Davis C., Menczer F., Flammini A. The Rise of Social Bots // Communications of the ACM. 2016. Vol. 59, No. 7. P. 96–104.

10. Pirozzi E. PostgreSQL 10 High Performance: Expert Techniques for Query Optimization, High Availability, and Efficient Database Maintenance. Birmingham: Packt Publishing Ltd, 2018.

11. Coronel C., Morris S., Rob P. Database Systems: Design, Implementation, and Management. Boston: Course Technology, Cengage Learning, 2011.

12. Rubin J., Chisnell D. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. Indianapolis: John Wiley & Sons, 2008.

REFERENCES

1. Mitchell R. Web Scraping with Python: Collecting More Data from the Modern Web. 2nd ed. Sebastopol: O'Reilly Media, 2018.

2. Rao N. V., et al. Optical Character Recognition Technique Algorithms // Journal of Theoretical and Applied Information Technology. 2016. Vol. 83, No. 2.

3. Hamad K., Kaya M. A Detailed Analysis of Optical Character Recognition Technology // International Journal of Applied Mathematics Electronics and Computers. 2016. Special Issue 1. P. 244–249.

4. Hrybiuk O. CleverCOMSRL Intelligent Expert System Knowledge Representation Model Features and Inconsistency Resolution Capabilities. In: Machado J., Trojanowska J., Soares F., Rea P., Butdee S., Gramescu B. (eds) Innovations in Mechatronics Engineering IV. icieng 2025. Lecture Notes in Mechanical Engineering. Springer, Cham. 2025. P. 57-68. https://doi.org/10.1007/978-3-031-94223-5_6.

5. Ortega J. M. Mastering Python for Networking and Security. Birmingham: Packt Publishing, 2018.

6. Sweigart A. Automate the Boring Stuff with Python. San Francisco: No Starch Press, 2025.

7. Han J., Pei J., Tong H. Data Mining: Concepts and Techniques. San Francisco: Morgan Kaufmann, 2022.

8. Følstad A., Brandtzæg P. B. Chatbots and the New World of HCI // Interactions. 2017. Vol. 24, No. 4. P. 38–42.

9. Ferrara E., Varol O., Davis C., Menczer F., Flammini A. The Rise of Social Bots // Communications of the ACM. 2016. Vol. 59, No. 7. P. 96–104.

10. Pirozzi E. PostgreSQL 10 High Performance: Expert Techniques for Query Optimization, High Availability, and Efficient Database Maintenance. Birmingham: Packt Publishing Ltd, 2018.

11. Coronel C., Morris S., Rob P. Database Systems: Design, Implementation, and Management. Boston: Course Technology, Cengage Learning, 2011.

12. Rubin J., Chisnell D. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests. Indianapolis: John Wiley & Sons, 2008.