НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ
Кафедра ЦИФРОВИХ ТЕХНОЛОГІЇ В ЕНЕРГЕТИЦІ

‘‘До захисту допущено’’
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА
‘‘___’’_____2024 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
‘‘Цифрові технології в енергетиці’’
зі спеціальності 122 *‘‘Комп’ютерні науки’’*
на тему: « Автоматична генерація звіту динаміки показників цін на агропродукцію»

Виконав:
студент IV курсу, групи ТМ-01
Колонтирський Назар Мирославович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник: *ст. викладач каф. цифрових технологій в енергетиці*
ст. викладач, Ольга БЕСПАЛА _____
(посада, науковий ступінь, прізвище, ім’я, по батькові) (підпис)

Рецензент:
_____ _____
(посада, науковий ступінь, прізвище, ім’я, по батькові) (підпис)

Нормоконтролер: *доцент, Андрій ДОНЕЦЬ* _____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших афторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА

«__»_____2024 р.

ЗАВДАННЯ
на дипломну роботу студенту

Колонтирському Назару Мирославовичу
(прізвище, ім’я, по батькові)

1. Тема роботи **Автоматична генерація звіту динаміки показників цін на агропродукцію**

керівник роботи ***Беспала Ольга Миколаївна, ст. викладач***
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 202__ р. №_____

2. Термін подання роботи студентом **07.06.2024 року**

3. Вихідні дані до роботи мови програмування Python, JavaScript, фреймворки Flask та Charts.js, СКБД MySQL, середовища розробки PyCharm, Visual studio code.

4. Перелік питань, які потрібно розробити
1) провести аналіз серед наявних аналогів
2) веб-застосунок для автоматичної генерації звіту
3) аргументувати вибрані інструменти, технології та алгоритми для реалізації програмного забезпечення

4) побудувати архітектури програмного забезпечення та баз даних
5) створити прикладний програмний веб-інтерфейс
6) представити процес застосування веб-інтерфейсу

5. Орієнтований перелік ілюстративного матеріалу діаграми, що демонструють роботу алгоритмів веб-інтерфейсу, архітектуру системи, бази даних та сховища файлів, взаємодію користувачів із системою, класи програмного забезпечення; приклади роботи з програмним продуктом.

6. Дата видачі завдання «19» вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Затвердження теми роботи		
2.	Вивчення та аналіз задачі	17-20.04.24	
3.	Розробка архітектури та загальної структури системи	21-25.04.24	
4.	Розробка частин окремих підсистем	25.04-02.05.24	
5.	Програмна реалізація системи	03-07.05.24	
6.	Оформлення пояснювальної записки	08-12.05.24	
7.	Захист програмного продукту	13-17.05.23	
8.	Передзахист	03-06.06.23	
9.	Подання готової роботи на кафедру	07.06.2024	
10.	Захист	17-21.06.2024	

Студент _____ **Назар КОЛОНТИРСЬКИЙ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ **Ольга БЕСПАЛА**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота виконана на 64 сторінках, містить 23 ілюстрації, 1 таблицю, 1 додаток, 25 джерел в переліку посилань.

Мета роботи – створення програмного забезпечення для автоматичної генерації звіту динаміки показників цін на агропродукцію.

Методи та засоби: Мова програмування Python, мова програмування JavaScript, бібліотека парсингу BeautifulSoup, бібліотека роботи з масивами даних Pandas, допоміжні фреймворки flask та ChartsJS.

Результат – Веб-застосунок для автоматичного збору даних та генерації звітів про динаміку цін на агропродукцію.

Ключові слова: ПАРСЕР, БД, ВІЗУАЛІЗАЦІЯ.

ANNOTATION

The thesis is completed on 64 pages, contains 23 illustrations, 1 table, 1 appendix, 25 sources in the list of references.

The purpose of the work is to create software for the automatic generation of a report on the dynamics of prices for agricultural products.

Methods and tools: Python programming language, JavaScript programming language, BeautifulSoup parsing library, Pandas data array library, supporting frameworks flask and ChartsJS.

The result is a web application for automatic data collection and generation of reports on price dynamics of agricultural products.

Keywords: PARSER, DB, VISUALIZATION.

ЗМІСТ

ВСТУП

В сучасних умовах ринку зернової продукції, інформація про динаміку цін на агропродукцію відіграє вирішальну роль у прийнятті рішень трейдерами. Зернові трейдери змушені постійно досліджувати багато джерел для пошуку актуальної інформації про ціни, попит, пропозицію та інші важливі показники. Цей процес займає значну кількість часу та зусиль, що суттєво знижує ефективність роботи трейдерів. Оптимізація цього процесу за допомогою спеціалізованої програми дозволить значно прискорити їх роботу, забезпечуючи своєчасний доступ до необхідної інформації. Моя програма, розроблена для автоматизації збору та обробки даних, дозволяє не тільки збирати інформацію з різних джерел, але й зберігати її в базі даних для довготривалого використання. Це забезпечує можливість аналізу динаміки цін та інших показників на тривалий період часу, що неможливо за допомогою багатьох джерел, з яких ці дані отримуються. Завдяки цьому, трейдери можуть більш ефективно планувати свої дії, приймаючи обґрунтовані рішення на основі комплексного аналізу даних.

Метою цієї роботи є розробка програми, яка забезпечує автоматизований збір інформації з різних джерел та її виведення в зручному для аналізу вигляді. Програма дозволяє користувачу відображати дані в табличному форматі з можливістю тонко налаштовувати вибір необхідних даних, а також у вигляді візуалізацій, які він може налаштовувати відповідно до своїх потреб. Таким чином, трейдери зможуть швидко і ефективно отримувати необхідну інформацію, що значно підвищує їх продуктивність та ефективність.

На сьогоднішній день існує багато окремих компаній, які публікують інформацію про ціни на зернову продукцію. Ці компанії надають цінні дані, однак проблема полягає у великій кількості різних джерел, які необхідно досліджувати для

отримання повної картини ринку. Кожне джерело може мати власний формат подання даних, що ускладнює їх аналіз та порівняння.

Незважаючи на існування багатьох джерел інформації, жодне з них не забезпечує комплексний збір та аналіз даних з різних джерел в одному місці. Багато існуючих рішень мають обмежену функціональність та не дозволяють гнучко налаштовувати відображення та аналіз даних. Також, більшість з них не надають можливість довготривалого збереження даних та порівняння динаміки на тривалий період часу.

Таким чином, розроблена програма буде актуальним та необхідним інструментом для сучасних трейдерів, забезпечуючи їх комплексною інформацією та інструментами для її аналізу.

1 ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ЗВІТУ ДИНАМІКИ ПОКАЗНИКІВ ЦІН НА АГРОПРОДУКЦІЮ

Автоматизація процесів збору, аналізу та візуалізації даних є критично важливою для сучасного аграрного сектору, який швидко розвивається. У даній дипломній роботі розглядається проблема автоматичної генерації звіту динаміки показників цін на агропродукцію. Це завдання є особливо актуальним у контексті зростаючих обсягів інформації та потреби в оперативному прийнятті рішень на основі точних та актуальних даних.

Відповідно до мети роботи, задача полягає у створенні ефективної системи, яка забезпечить автоматичний збір даних про ціни на зернові культури з різних джерел, їх подальше зберігання, обробку та візуалізацію для кінцевих користувачів.

Для досягнення цієї мети необхідно розробити програмний продукт, що забезпечить користувачів необхідним функціоналом та дозволятиме користувачам взаємодіяти з системою через зручний веб-інтерфейс.

У першому розділі даної роботи буде розглянуто актуальність теми та проблеми аналізу даних в аграрному секторі, визначено потенційних користувачів системи та проаналізовано переваги розробленої платформи у порівнянні з існуючими рішеннями.

1.1 Актуальність та проблеми аналізу даних про ціни на агропродукцію

Актуальність автоматичної генерації звіту динаміки показників цін на агропродукцію обумовлена декількома ключовими факторами, які визначають необхідність розробки та впровадження таких систем у аграрному секторі.

По-перше, ринок агропродукції є дуже динамічним і чутливим до різноманітних зовнішніх факторів, таких як зміни кліматичних умов, політичні рішення, економічні кризи, бойові дії, та інші. Ці фактори можуть суттєво впливати на ціни агропродукції, роблячи їх нестабільними і непередбачуваними. У таких умовах для аграрних підприємств критично важливо мати доступ до актуальних і точних даних про ціни для прийняття своєчасних і обґрунтованих рішень щодо виробництва, зберігання та продажу продукції.

По-друге, обсяг даних, що генерується в аграрному секторі, постійно зростає. Зростання кількості інформації ускладнює процес її обробки та аналізу, що вимагає використання сучасних інструментів для автоматизації цих процесів. Ручне збирання та аналіз даних займає багато часу і ресурсів, а також може бути пов'язане з помилками, що знижує точність отриманих результатів. Автоматизовані системи дозволяють значно зменшити ці ризики, забезпечуючи швидку і точну обробку великих обсягів інформації.

По-третє, використання сучасних технологій для автоматизації аналізу даних дозволяє аграрним підприємствам підвищити свою конкурентоспроможність. Завдяки таким системам підприємства можуть оперативно реагувати на зміни ринкових умов, оптимізувати свої виробничі та збутові процеси, а також знижувати витрати, пов'язані з прийняттям неправильних рішень через недостатню або неточну інформацію.

Основними проблемами аналізу даних в аграрному секторі є різноманітність джерел даних, великий обсяг даних, відсутність стандартів даних та необхідність швидкого аналізу. Дані про ціни на агропродукцію можуть надходити з різних джерел, таких як урядові портали, комерційні веб-сайти, біржі, новинні ресурси тощо. Це ускладнює їх збір та інтеграцію в єдину систему. Різні джерела можуть надавати дані в різних форматах, що ускладнює їх обробку та аналіз. Це вимагає розробки спеціальних алгоритмів для нормалізації та уніфікації даних. Щоденний моніторинг цін на агропродукцію генерує великі обсяги даних, що вимагає використання потужних обчислювальних ресурсів для їх обробки. Зміни на ринку можуть

відбуватися дуже швидко, тому аграрним підприємствам потрібні інструменти, які дозволяють оперативно аналізувати дані і приймати обґрунтовані рішення.

Розроблена система для автоматичної генерації звіту динаміки показників цін на агропродукцію покликана вирішити вищезазначені проблеми. Вона забезпечує автоматизований збір даних з різних джерел, їх нормалізацію та зберігання у базі даних, а також надає користувачам зручні інструменти для аналізу і візуалізації інформації. Це дозволяє аграрним підприємствам отримувати актуальні і точні дані про ціни на агропродукцію, що є ключовим фактором для ефективного управління їх діяльністю.

1.2 Потенційні користувачі

Розроблена система автоматичної генерації звіту динаміки показників цін на агропродукцію має широкий спектр потенційних користувачів, кожен з яких може отримати значні переваги від її використання. Враховуючи різноманітність потреб та завдань, які стоять перед учасниками аграрного ринку, система забезпечує інструменти для задоволення специфічних вимог кожної групи користувачів.

Перш за все, система буде корисною для аграрних підприємств, які займаються вирощуванням, переробкою та збутом зернових культур. Для них доступ до актуальної та точної інформації про ціни на ринку є критично важливим. Завдяки системі, підприємства зможуть оперативно отримувати дані про поточні ціни на різних ринках, аналізувати тенденції та динаміку змін цін, що дозволить приймати обґрунтовані рішення щодо продажу продукції, планування виробництва та управління запасами.

Другою важливою групою користувачів є аналітики та дослідники, які займаються вивченням ринкових тенденцій та підготовкою звітів для різних зацікавлених сторін. Система надає інструменти для глибокого аналізу даних,

побудови графіків та діаграм, що допомагає виявляти закономірності та прогнозувати майбутні зміни на ринку. Аналітики зможуть використовувати отримані дані для підготовки детальних звітів, наукових досліджень та консультацій.

Третьою групою користувачів є трейдери та брокери, які працюють на ринку агропродукції. Для них важливо мати доступ до оперативної інформації про ціни та умови торгівлі. Система дозволяє отримувати актуальні дані про ціни на різних ринках, порівнювати їх та обирати найкращі умови для укладання угод. Це дозволяє трейдерам приймати виважені рішення, мінімізувати ризики та збільшувати прибутковість своєї діяльності.

Також система може бути корисною для державних установ та організацій, які займаються регулюванням аграрного ринку та розробкою аграрної політики. Вони можуть використовувати зібрані дані для моніторингу стану ринку, оцінки ефективності державних програм та прийняття рішень щодо підтримки аграрного сектора. Доступ до надійних та актуальних даних дозволяє більш ефективно планувати та реалізовувати заходи, спрямовані на розвиток аграрного сектору та забезпечення продовольчої безпеки.

Нарешті, система може бути корисною для освітніх та наукових установ, які займаються підготовкою фахівців для аграрного сектору та проведенням досліджень. Використання реальних даних у навчальному процесі дозволяє студентам отримувати практичні навички роботи з інформацією, аналізу ринкових даних та прийняття управлінських рішень. Це сприяє підвищенню якості підготовки кадрів та розвитку наукових досліджень у сфері аграрної економіки.

Таким чином, розроблена система має широкий спектр потенційних користувачів, які можуть отримати значні переваги від її використання. Це аграрні підприємства, аналітики, трейдери, державні установи, а також освітні та наукові установи. Система забезпечує інструменти для автоматичного збору, аналізу та візуалізації даних, що дозволяє кожній з цих груп користувачів ефективніше виконувати свої завдання та досягати поставлених цілей.

1.3 Огляд та аналіз існуючих систем

Наразі існує безліч джерел інформації про ціни на зернову продукцію, в більшості своїй це веб-сайти різних типів та функціоналів, зустрічаються також і додатки для смартфонів. Втім ці джерела не надають вичерпну інформацію по темі і для формування висновків спеціалістам цієї галузі доводиться вивчати та порівнювати інформацію з багатьох джерел. Існують також інструменти та програми для агрегації даних з багатьох джерел та формування візуалізацій, але ці рішення також мають свої недоліки.

Розглянемо декілька прикладів:

- AgriCensus – це глобальна новинна служба, яка спеціалізується на наданні інформації про ринки зернових та інших сільськогосподарських продуктів. Вони надають щоденні оновлення цін на різні види зерна, включаючи пшеницю, кукурудзу, ячмінь та сою. Крім цінових даних, AgriCensus також пропонує аналітичні звіти та ринкові огляди, які допомагають користувачам краще розуміти тенденції на ринку. Крім того, платформа надає інструменти для візуалізації даних, що дозволяє користувачам швидко аналізувати та інтерпретувати інформацію [1].

Переваги: Постійне оновлення даних забезпечує актуальність інформації, широкий спектр аналітичних звітів і досліджень, інтерактивні інструменти та налаштовувані сповіщення роблять сервіс зручним для користувачів різних рівнів.

Недоліки: Високі витрати на підписку можуть бути недоступними для малих підприємств або індивідуальних користувачів;

- Tableau - це один з найпопулярніших інструментів для бізнес-аналітики та візуалізації даних. Він дозволяє підключатися до різноманітних джерел даних, включаючи SQL бази даних, електронні таблиці, хмарні сервіси та API. Tableau надає можливість створення інтерактивних дашбордів та графіків. Крім того, сервіс має потужні інструменти для аналітики, такі як фільтри, обчислення та прогнозування.

Однією з переваг Tableau є можливість спільного доступу до дашбордів та звітів через веб-інтерфейс або мобільний додаток [2].

Переваги: Простота у використанні, потужні інструменти для аналізу даних, широкий спектр підтримуваних джерел даних.

Недоліки: Висока вартість ліцензії, проблематичність роботи при відсутності готової бази даних, або API у джерелах;

- Power BI – це інструмент від Microsoft для бізнес-аналітики, який дозволяє імпортувати, обробляти та візуалізувати дані з різних джерел. Він підтримує підключення до Excel, SQL баз даних, веб-служб та API. Power BI пропонує велику кількість вбудованих типів візуалізацій, а також можливість створення користувацьких візуалізацій. Тісна інтеграція з іншими продуктами Microsoft, такими як Office 365 та Azure, робить Power BI особливо привабливим для користувачів екосистеми Microsoft [3].

Переваги: Інтуїтивно зрозумілий інтерфейс, гнучкі можливості візуалізації, відносно низька вартість у порівнянні з іншими рішеннями.

Недоліки: Обмежена функціональність у безкоштовній версії, необхідність створення візуалізацій для власних потреб, що може бути занадто складно, або незручно для людини незнайомої з програмою, проблематичність роботи при відсутності готової бази даних, або API у джерелах.

Існуючі системи мають ряд переваг наприклад гарний дизайн, або інтеграція з існуючими сервісами. Втім мають готові рішення і недоліки, для вирішення яких і розроблявся мій програмний продукт.

У порівнянні з веб-сайтами, які дозволяють переглядати інформацію про ціни на агропродукцію, розроблений застосунок значно знижує трудомісткість та час на отримання інформації, адже відсутній етап ручного збору даних. Автоматизація також мінімізує ризик помилок, пов'язаних з людським фактором.

Друга перевага системи полягає в її здатності інтегрувати дані з різних джерел у єдину базу даних, незалежно від присутності чи відсутності на ресурсі API

функціоналу. Це забезпечує користувачам можливість аналізувати інформацію з різних джерел у зручному форматі. Система також підтримує функції нормалізації та уніфікації даних, що дозволяє забезпечити їх сумісність та коректність.

Система може бути легко адаптована для збору даних з нових джерел та аналізу додаткових параметрів. Наприклад, до системи можуть бути додані нові парсери для збору даних з нових веб-сайтів.

Таким чином, розроблена система автоматичної генерації звіту динаміки показників цін на агропродукцію має ряд суттєвих переваг над існуючими аналогами. Вона забезпечує високу ступінь автоматизації, інтеграцію даних з різних джерел, зручний веб-інтерфейс, гнучкість та масштабованість, а також надійність та доступність для користувачів усіх рівнів.

1.4 Задачі, які повинна вирішувати система

Відповідно до проблем, що існують на ринку, та недоліків існуючих систем, які були розглянуті у попередніх пунктах задача полягає у створенні ефективної системи, яка забезпечить закриття ніші на ринку, та дозволить користувачам усіх рівнів зручно та ефективно нею користуватись, не маючи спеціальних навичок.

Для досягнення цієї мети розроблювана система повинна:

- автоматично збирати дані з багатьох актуальних ресурсів;
- перевіряти зібрані дані на коректність;
- приводити дані з різних джерел до спільного формату по таким параметрам як формат дати, валюта операцій, одиниці виміру, спільний формат позначення текстових даних, таких як скорочення, назви областей, назви культур та інші;
- записувати дані у базу даних, для довготривалого зберігання;
- оптимізувати інтерфейс для представлення даних у зручному форматі;

- надавати функціонал для сортування та фільтрування даних по усім параметрам, наприклад джерело даних, діапазон ціни, діапазон дати, культура та інші;

- надавати функціонал для зручного та швидкого пошуку, як серед великого обєму табличних даних, так і серед великої кількості елементів фільтрації, наприклад коли різних компаній можуть бути сотні, а користувача цікавить якась конкретна;

- генерувати візуалізації даних по ключовим параметрам, таким як графіки динаміки цін (середня, максимальна, мінімальна) за певний період, графіки динаміки цін по найбільшим компаніям, кількість транзакцій у різних компаній, графіки ціни та кількості транзакцій по дням тижня, тощо;

- надавати функціонал для експорту згенерованих візуалізацій, для того щоб користувач міг швидко роздрукувати, або завантажити потрібний йому графік, або діаграму;

- представляти увесь функціонал простим і зручним інтерфейсом, аби користувачі будь якого рівня комп’ютерної грамотності могли легко освоїти систему.

При виконанні цих умов, програма буде мати місце на ринку інформації про ціни на агропродукцію, яке наразі ніким не зайняте. Адже для досягнення подібного ефекту користувачу довелося б, або довго збирати інформацію вручну, або мати спеціальні навички для роботи з вузькоспеціалізованими середовщами.

2 МЕТОДИ ТА АЛГОРИТМИ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ЗВІТУ ДИНАМІКИ ПОКАЗНИКІВ ЦІН НА АГРОПРОДУКЦІЮ

Цей розділ охоплює всі аспекти розробки системи, починаючи від збору та обробки даних, роботи з базою даних, автоматизації завдань, а також візуалізації та представлення даних користувачам.

Спочатку буде розглянуто процес збору та обробки даних, що включає парсинг даних із веб-джерел, фільтрацію та очищення даних, а також округлення цін та конвертацію валют для забезпечення точності та консистентності даних.

У частині про роботу з базою даних буде описано процес підключення до бази даних, вставки та оновлення даних, що забезпечує збереження та актуалізацію інформації. Окрема увага приділяється автоматизації завдань, зокрема використанню планувальника завдань для ефективної автоматизації операцій.

Завершується розділ описом методів візуалізації та представлення даних, що включає створення графіків та діаграм, а також обробку HTTP-запитів та форматування відповідей у форматі JSON. Ці елементи забезпечують зручність використання системи та надають користувачам можливість легко аналізувати та інтерпретувати отримані дані.

2.1 Збір даних

Збір даних є першим та ключовим етапом в процесі автоматичної генерації звіту динаміки показників цін на агропродукцію. Враховуючи специфіку ринку та необхідність отримання актуальних і точних даних, було прийнято рішення використовувати дані з декількох надійних інтернет-джерел, які регулярно публікують інформацію про ціни на зернові культури.

Для збору даних ми обрали кілька основних джерел, зокрема сайти agrotender.com, agritel.com та nibulon.com. Ці веб-ресурси надають актуальну інформацію про ціни на різні види зернових культур, що дозволяє отримувати комплексний огляд ринку [4-6].

Парсинг даних із веб-джерел є критичним етапом у процесі збору інформації про ціни на агропродукцію. Цей метод дозволяє автоматично витягувати необхідні дані з веб-сторінок, що забезпечує оперативність і точність отримання інформації. У системі використовується бібліотека BeautifulSoup для Python, яка є потужним інструментом для парсингу HTML та XML документів.

Процес парсингу починається з надсилання HTTP-запиту до цільового веб-сайту за допомогою бібліотеки requests. Після отримання відповіді метод аналізує HTML-код сторінки, виділяючи ті елементи, які містять необхідні дані. Для цього використовуються методи бібліотеки BeautifulSoup, які дозволяють знайти та витягти текстові значення з конкретних тегів та атрибутів HTML-документу.

Алгоритм роботи парсерів:

- ініціалізація та налаштування: Парсери ініціалізуються з необхідними налаштуваннями, такими як цільові веб-сайти, заголовки HTTP-запитів, параметри підключення до бази даних тощо. Цей етап також включає налаштування розкладу для автоматичного запуску парсерів;

- надсилання HTTP-запитів: Парсери надсилають HTTP-запити до цільових веб-сайтів, використовуючи налаштовані заголовки та параметри запитів. Бібліотека requests дозволяє гнучко керувати запитами, включаючи обробку редиректів, кукіз та інші аспекти HTTP-з'єднань;

- отримання та обробка відповідей: Отримані відповіді у вигляді HTML-документів передаються до модуля парсингу для подальшої обробки. У випадку помилок або невдач при отриманні відповіді, парсери можуть повторити запит або зареєструвати помилку для подальшого аналізу;

- парсинг HTML-документів: Модуль парсингу аналізує HTML-документи, витягує необхідну інформацію та перетворює її у структурований вигляд. Використання бібліотеки BeautifulSoup дозволяє легко знаходити та обробляти потрібні елементи на сторінці, такі як таблиці, заголовки, посилання та інші HTML-елементи.

Один із важливих аспектів парсингу – це обробка різних структур веб-сторінок, оскільки різні сайти можуть мати різні формати представлення даних. Для цього ми створили окремі парсери для кожного джерела даних, що дозволяє враховувати специфіку кожного сайту та забезпечувати коректне витягування інформації.

Також важливою є періодичність виконання парсингу, оскільки дані про ціни на агропродукцію можуть швидко змінюватися. Тому ми налаштували планувальник завдань для регулярного запуску парсерів, що дозволяє підтримувати актуальність зібраної інформації.

Таким чином, збір даних із веб-джерел за допомогою BeautifulSoup є ефективним методом автоматичного збору інформації про ціни на агропродукцію, що забезпечує точність, швидкість та актуальність даних, необхідних для подальшого аналізу та візуалізації.

2.2 Обробка даних

Зібрані дані часто представлені в різних форматах та можуть містити розбіжності, що ускладнює їх подальший аналіз. Тому важливим етапом після збору є обробка та уніфікація даних. Цей процес включає фільтрацію та очищення даних, що допомагає видалити помилки та дублікати, а також конвертацію цін у єдину валюту для забезпечення порівнянності.

Уніфікація даних також передбачає їх нормалізацію, що дозволяє привести різні формати до єдиного стандарту. Це включає, наприклад, перетворення дат у єдиний

формат, приведення назв культур та компаній до узгодженого вигляду. Таким чином, уніфікація даних дозволяє забезпечити консистентність і точність інформації, що є критично важливим для подальшого аналізу та візуалізації.

Загалом, обробка даних з різних інтернет-джерел, їх уніфікація та нормалізація створюють надійну базу для коректного аналізу, формування візуалізацій та роботи інших функцій системи.

2.2.1 Очищення даних

Після парсингу даних із веб-джерел, отримана інформація потребує ретельної фільтрації та очищення. Цей процес є критично важливим для забезпечення якості та точності даних, що використовуються для подальшого аналізу та генерації звітів. Фільтрація та очищення даних включають кілька ключових етапів, спрямованих на видалення помилок, дублікати та нормалізацію даних.

Основним завданням фільтрації є видалення зайвої або нерелевантної інформації, яка може бути присутня у вихідних даних. Наприклад, парсинг веб-сторінок може витягувати текстові дані, що не мають значення для нашого аналізу, такі як рекламні блоки, коментарі або інші небажані елементи. Фільтрація дозволяє зосередитися на ключових даних, таких як ціни, назви культур, компанії та дати.

Алгоритм очистки даних включає такі кроки:

- видалення дублікатів: Дублікати можуть виникати внаслідок кількох причин, таких як повторний парсинг тих самих даних або наявність однакових записів на різних веб-ресурсах. Видалення дублікатів є необхідним для забезпечення унікальності кожного запису в базі даних;

- обробка відсутніх значень: У вихідних даних можуть бути відсутні деякі значення або поля, що потребують заповнення. Це може включати додавання значень за замовчуванням або видалення записів з критично важливими відсутніми даними;

- нормалізація форматів: Різні джерела можуть використовувати різні формати для представлення одних і тих самих даних. Наприклад, дати можуть бути

представлені в різних форматах, що ускладнює їх порівняння та аналіз. Нормалізація форматів включає перетворення дат, числових значень та інших даних до єдиного стандарту;

- видалення зайвих символів та форматування: Дані, зібрані з веб-сторінок, можуть містити зайві символи, пробіли або HTML-теги, які потрібно видалити. Форматування даних дозволяє привести їх до чистого та читабельного вигляду.

Крім того, важливим аспектом очищення даних є валідація, яка включає перевірку коректності даних. Це може включати перевірку на відповідність очікуваним діапазонам значень (наприклад, ціни не можуть бути від'ємними) та логічні перевірки (наприклад, дата продажу не може бути майбутньою).

Таким чином, процес фільтрації та очищення даних є важливим етапом у забезпеченні якості інформації, що використовується у системі автоматичної генерації звіту динаміки показників цін на агропродукцію. Ці кроки гарантують, що дані є точними, консистентними та готовими до подальшого аналізу та візуалізації.

2.2.2 Приведення даних до спільного вигляду

Після процесів парсингу, фільтрації та очищення даних, важливим етапом є приведення даних до спільного вигляду. Цей етап забезпечує консистентність та порівнянність даних, що є критично важливим для подальшого аналізу та візуалізації.

Округлення цін є необхідним для стандартизації даних, оскільки різні джерела можуть надавати ціни з різною точністю. Наприклад, одні джерела можуть надавати ціни з точністю до цента, тоді як інші – до гривні або долара. Для забезпечення єдиної точності даних всі ціни округлюються до найближчого цілого числа або до певного знака після коми, в залежності від вимог до точності аналізу.

Процес округлення виконується за допомогою відповідних математичних методів, які забезпечують коректне округлення чисел. В Python для цього використовуються вбудовані функції, такі як round(), які дозволяють швидко і точно

округлити числові значення до потрібного формату. Округлення цін забезпечує легкість в подальшій роботі з даними та покращує їх сприйняття користувачами.

Конвертація валют є ще одним важливим етапом, оскільки дані з різних джерел можуть бути представлені у різних валютах. Наприклад, одні джерела можуть надавати ціни у гривнях (UAH), тоді як інші – у доларах США (USD) або євро (EUR). Для забезпечення порівнянності даних необхідно привести всі ціни до єдиної валюти.

Алгоритм конвертації включає множення ціни у вихідній валюті на відповідний курс обміну для отримання ціни в цільовій валюті. Наприклад, для конвертації ціни з USD в UAH використовується поточний курс долара до гривні. Після конвертації ціни всі дані зберігаються в єдиній валюті, що спрощує їх подальший аналіз та візуалізацію.

Ще одним важливим етапом є приведення дат до спільного формату, оскільки в різних джерелах дати можуть відрізнятися форматами, що ускладнило б подальшу роботу.

Таким чином, приведення даних до спільного вигляду є важливим етапом обробки даних, що забезпечує їх консистентність та порівнянність. Цей процес дозволяє підвищити якість даних та забезпечують зручність їх використання для аналізу динаміки цін на агропродукцію.

2.3 Запис в базу даних

Цей метод є важливим етапом у створенні системи автоматичної генерації звіту динаміки показників цін на агропродукцію. Він забезпечує надійне зберігання, організацію та доступ до даних, що є необхідними для подальшого аналізу та візуалізації. Основними аспектами роботи з базою даних є підключення до бази даних, вставка та оновлення даних, а також забезпечення цілісності та ефективного доступу до інформації.

2.3.1 Підключення до бази даних

У системі використовується СУБД MySQL, яка є однією з найпопулярніших систем управління базами даних завдяки своїй надійності, продуктивності та простоті використання. Для підключення до MySQL з використанням Python була застосована бібліотека SQLAlchemy.

Алгоритм підключення до бази даних включає:

- визначення параметрів підключення: Це включає в себе встановлення таких параметрів, як ім'я користувача, пароль, адреса сервера та назва бази даних. Ці параметри зазвичай зберігаються у файлі конфігурації для забезпечення безпеки та зручності управління;

- створення двигуна підключення: SQLAlchemy використовує концепцію "двигуна" для управління підключенням до бази даних. Двигун створюється з використанням методу create_engine, куди передаються параметри підключення. Двигун відповідає за відкриття і закриття з'єднань, виконання SQL-запитів та управління транзакціями;

- створення сесії: Для виконання операцій з базою даних SQLAlchemy використовує сесію, яка представляє собою обгортку над двигуном та забезпечує контекст для виконання запитів. Сесія створюється за допомогою класу sessionmaker, який зв'язується з двигуном та дозволяє виконувати запити до бази даних;

- підключення та перевірка: Після створення двигуна та сесії здійснюється підключення до бази даних та перевірка його успішності. Це включає виконання простого тестового запиту для перевірки доступності бази даних та коректності налаштувань підключення;

- підключення до бази даних також передбачає обробку помилок, які можуть виникнути під час встановлення з'єднання. Це включає обробку таких помилок, як неправильні параметри підключення, відсутність доступу до сервера або бази даних, та інші технічні проблеми. Обробка помилок забезпечує стабільність роботи системи та надає можливість вчасно реагувати на проблеми, що виникають.

2.3.2 Перевірка та запис даних

Цей етап забезпечує збереження нових даних, отриманих в результаті парсингу та обробки, а також оновлення вже існуючих записів у базі даних.

Алгоритм перевірки та запису даних у БД включає такі кроки:

- підготовка даних: Перед вставкою, зібрані дані проходять етап підготовки, що включає їх очистку, нормалізацію та перетворення у формат, сумісний з базою даних. Це забезпечує цілісність та відповідність даних вимогам структури бази;

- перевірка унікальності: Перед вставкою нових записів важливо перевірити їх унікальність, щоб уникнути дублювання даних. Це може бути реалізовано через порівняння нових даних з уже існуючими записами у базі, використовуючи унікальні ключі або інші атрибути;

- створення записів: Для вставки нових даних у базу даних використовуються відповідні SQL-запити. У випадку SQLAlchemy, створення нових записів реалізується через об'єктно-реляційне відображення (ORM), де кожен запис представляється як об'єкт класу, відповідного таблиці бази даних;

- вставка даних: Нові записи додаються до сесії SQLAlchemy за допомогою методу add() або add_all() для множинних записів. Після цього сесія фіксується методом commit(), що зберігає зміни в базі даних.

2.4 Автоматизація збору даних

Автоматизація завдань є важливим методом в роботі системи автоматичної генерації звіту динаміки показників цін на агропродукцію. Він забезпечує своєчасне виконання регулярних завдань, таких як збір та обробка даних, оновлення бази даних, без необхідності втручання користувачів. Завдяки автоматизації, система може

функціонувати безперебійно та ефективно, навіть при великому обсязі даних та складності операцій.

Планування завдань полягає у визначенні розкладу виконання певних дій у системі. Це може бути досягнуто за допомогою різних інструментів та бібліотек, які дозволяють налаштовувати та управляти виконанням завдань. Одним з таких інструментів є бібліотека schedule у Python, яка надає зручні методи для визначення розкладу запуску функцій. Використовуючи schedule, можна налаштувати запуск парсера даних та оновлення бази даних у визначений час, наприклад, щодня о 9:00 ранку.

Ключовим завданням, що автоматизується у нашій системі, є регулярний збір даних з веб-джерел. Це завдання включає запуск парсера у визначений час для отримання актуальної інформації про ціни на агропродукцію. Після збору дані проходять етапи фільтрації, очищення, округлення та конвертації, що також автоматизовані. Отримані та оброблені дані автоматично додаються до бази даних.

Таким чином, автоматизація завдань є критичним компонентом системи автоматичної генерації звіту динаміки показників цін на агропродукцію. Вона забезпечує своєчасне виконання необхідних операцій, підвищуючи ефективність роботи системи та мінімізуючи необхідність втручання користувачів, забезпечуючи безперебійний та регулярний збір даних.

2.5 Візуалізація даних

Метод для створення графіків та діаграм є невід'ємною частиною процесу візуалізації даних в системі автоматичної генерації звіту динаміки показників цін на агропродукцію. Графіки та діаграми допомагають користувачам зрозуміти складні

набори даних, надаючи їм можливість візуально оцінювати тенденції, порівнювати показники та виявляти аномалії.

Для створення графіків та діаграм у нашій системі ми використовуємо бібліотеку Chart.js, яка є потужним інструментом для створення інтерактивних та настроюваних графічних елементів. Chart.js підтримує різноманітні типи графіків, включаючи лінійні графіки, гістограми, кругові діаграми, графіки розсіювання та інші, що дозволяє обирати найбільш підходящий тип для конкретних даних.

Алгоритм візуалізації даних включає кілька етапів:

- підготовка даних: Дані, що будуть відображені на графіках, повинні бути структуровані у відповідному форматі. Це включає організацію даних у вигляді масивів або об'єктів, що містять значення осей, мітки та інші необхідні параметри;

- налаштування параметрів графіка: Для кожного графіка визначаються параметри, такі як тип графіка, кольори, заголовки, підписи осей, легенда та інші опції. Ці параметри можуть бути налаштовані відповідно до вимог користувачів та специфіки даних;

- інтеграція з веб-інтерфейсом: Графіки та діаграми інтегруються у веб-інтерфейс системи, що дозволяє користувачам взаємодіяти з ними в режимі реального часу. Це включає відображення графіків на веб-сторінках, додавання функцій для фільтрації та сортування даних, а також налаштування інтерактивних елементів, таких як підказки та анімації.

Створення графіків та діаграм значно покращує сприйняття інформації користувачами. Лінійні графіки дозволяють візуально оцінювати зміни показників у часі, гістограми показують розподіл значень, а кругові діаграми допомагають аналізувати складові частини загального обсягу. Інтерактивні елементи, такі як підказки та анімації, роблять графіки більш інформативними та зручними у використанні.

2.6 Обробка HTTP-запитів та форматування відповідей у JSON

Обробка HTTP-запитів та форматування відповідей у форматі JSON є необхідними аспектами для забезпечення взаємодії між клієнтською та серверною частинами системи автоматичної генерації звіту динаміки показників цін на агропродукцію. Використання HTTP-запитів дозволяє клієнтським додаткам запитувати необхідну інформацію з сервера та отримувати її у зручному форматі для подальшого використання.

Серверна частина додатка, побудована на основі Flask, забезпечує обробку різних типів HTTP-запитів, таких як GET та POST. GET-запити використовуються для отримання даних, наприклад, для запиту актуальних цін на агропродукцію, тоді як POST-запити дозволяють відправляти дані на сервер для обробки, наприклад, для запуску парсингу нових даних.

Форматування відповідей у форматі JSON забезпечує стандартизований спосіб передачі даних між сервером та клієнтом. JSON є легким для читання та написання форматом, який широко використовується у веб-розробці завдяки своїй сумісності з більшістю мов програмування та простоті використання. JSON дозволяє структурувати дані у вигляді ключ-значення, що забезпечує зручний доступ до необхідної інформації.

Алгоритм обробки HTTP-запитів у Flask включає такі етапи:

- налаштування маршрутів: Маршрути визначають, які функції обробляють певні запити. Наприклад, маршрут /get_data обробляє запити на отримання даних, а маршрут /run_parser запускає парсер для збору нових даних;

- обробка запитів: Коли сервер отримує запит, відповідна функція обробляє його, виконує необхідні дії (наприклад, отримання даних з бази) та готує відповідь;

- форматування відповіді: Підготовлені дані формуються у форматі JSON за допомогою методу jsonify(), що забезпечує правильне перетворення даних у формат, зрозумілий для клієнтської частини;

- відправка відповіді: Сервер відправляє сформовану відповідь клієнту, що дозволяє користувачам отримувати доступ до необхідної інформації у зручному форматі.

Завдяки використанню HTTP-запитів та форматування відповідей у JSON, система забезпечує ефективний обмін даними між клієнтом та сервером. Це дозволяє клієнтським додаткам динамічно отримувати актуальну інформацію та взаємодіяти з серверною частиною для виконання різноманітних операцій, забезпечуючи інтерактивність та зручність використання системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ЗВІТУ ДИНАМІКИ ПОКАЗНИКІВ ЦІН НА АГРОПРОДУКЦІЮ

Буде розглянуто різні аспекти архітектури системи, потоки даних та точки взаємодії користувачів з системою. Буде розглянуто, як компоненти системи взаємодіють між собою та які інструменти використовуються для розробки і підтримки її роботи.

Розділ починається з опису загальної архітектури системи, де детально пояснюється структура парсерів, веб-інтерфейсу та бази даних. Це включає в себе обґрунтування вибору архітектурних рішень та пояснення, як ці компоненти взаємодіють між собою для забезпечення безперебійного збору, обробки та візуалізації даних.

Далі розглядаються потоки даних у системі, де пояснюється, як дані переміщуються від моменту їх отримання з веб-джерел до збереження у базі даних та відображення у веб-інтерфейсі. Це включає обробку даних на різних етапах та забезпечення їх цілісності та актуальності.

Також у розділі описуються точки взаємодії з системою, тобто способи, якими користувачі можуть взаємодіяти з системою для отримання даних, запуску парсерів, генерації звітів та інших дій. Це охоплює як інтерфейсні елементи, так і програмні інтерфейси.

Окремий підрозділ присвячений мовам програмування, що використовуються у системі, зокрема Python, JavaScript, HTML та CSS. Кожна мова обговорюється у контексті її ролі у системі та переваг її використання.

На завершення розділу розглядаються інструменти, які використовуються для розробки та підтримки системи, такі як Visual Studio Code, PyCharm та MySQL. Описуються їх функції та переваги, а також як вони сприяють ефективній розробці та управлінню проєктом.

Цей розділ надає повне уявлення про програмну реалізацію системи, пояснює вибір технологій та інструментів, а також описує, як вони інтегруються для створення ефективного та надійного рішення для автоматичної генерації звітів про динаміку цін на агропродукцію.

3.1 Архітектура системи

Архітектура системи автоматичної генерації звіту динаміки показників цін на агропродукцію складається з серверної частини, клієнтської частини та бази даних. Ці частини включають в себе програмні модулі, кожен з яких виконує своє завдання відповідно до сучасного шаблону проектування Model-View-Controller (MVC).

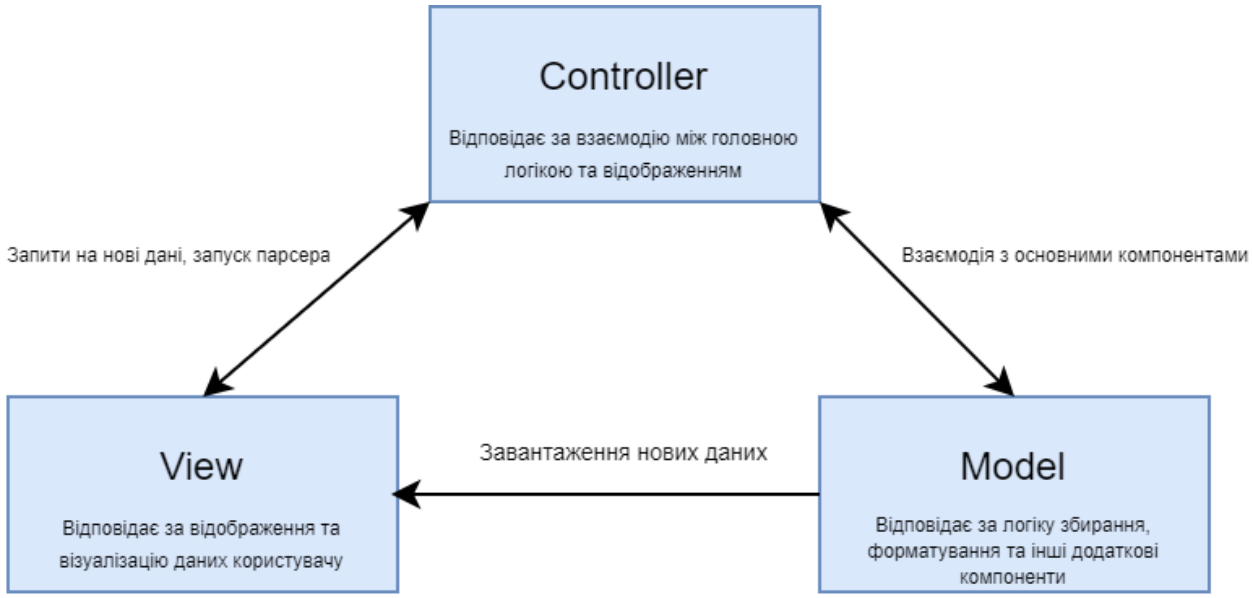


Рисунок 3.1 – Діаграма архітектури MVC

Серверна частина системи виконує функції збору, обробки та запису даних, вона складається з таких компонентів:

- DataProcessor (ProcessData.py) відповідає за обробку даних, включаючи видалення рядків, округлення цін, конвертацію валют, видалення дублікатів та оновлення філій.

- DatabaseHandler (database.py) забезпечує запис та отримання даних з бази даних, а також з'єднання з базою даних та отримання даних за множинними параметрами.

- Parser (parser.py) здійснює парсинг веб-сайтів для збору даних про ціни на агропродукцію.

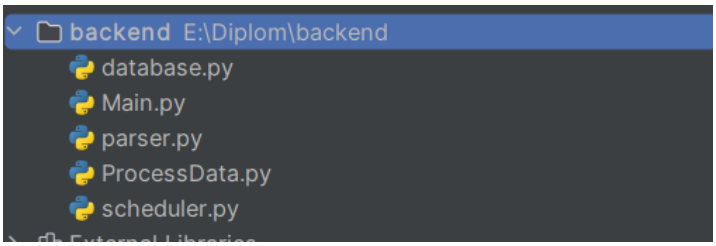


Рисунок 3.2 – Компоненти серверної частини системи

Клієнтська частина системи забезпечує взаємодію з користувачем та відображення даних. Вона включає такі компоненти:

- HTML Template (index.html) містить основні елементи інтерфейсу користувача, такі як форми фільтрів, таблиці та модальні вікна.

- graphics (graphics.js) відповідає за візуалізацію даних, включаючи генерацію графіків та діаграм на основі даних.

- контролер здійснює взаємодію між моделлю та поданням, забезпечуючи бізнес-логіку системи. Він складається з наступних компонентів:

- main (Main.py) відповідає за обробку запитів від клієнтської частини, запускає парсер та виконує основні операції з даними.

- scheduler (scheduler.py) автоматизує процес збору даних, запускаючи парсер у визначений час.

- connector (connector.js) відповідає за завантаження даних з сервера та їх оновлення на клієнтській частині.

- main (main.js) ініціалізує події на стороні клієнта, забезпечуючи взаємодію користувача з інтерфейсом.

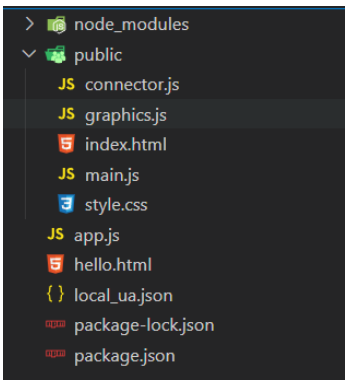


Рисунок 3.3 – Компоненти клієнтської частини системи

Повну картину архітектури MVC продемонстровано на рисунку 3.4.

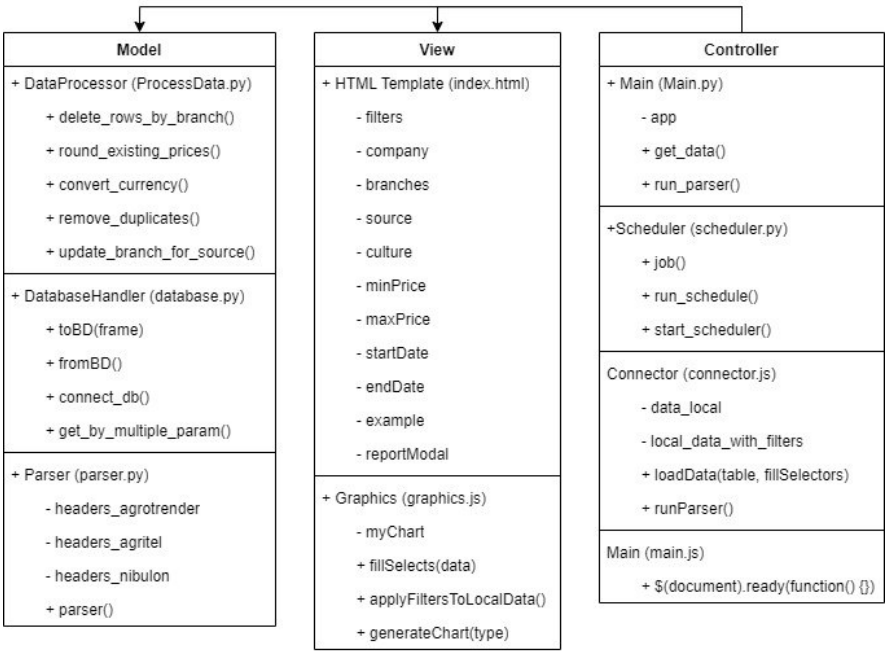


Рисунок 3.4 – Повна архітектура MVC

На діаграмі послідовності, яку представлено на рисунку 3.5 зображено типову послідовність взаємодії різних частин системи між собою та з користувачем.

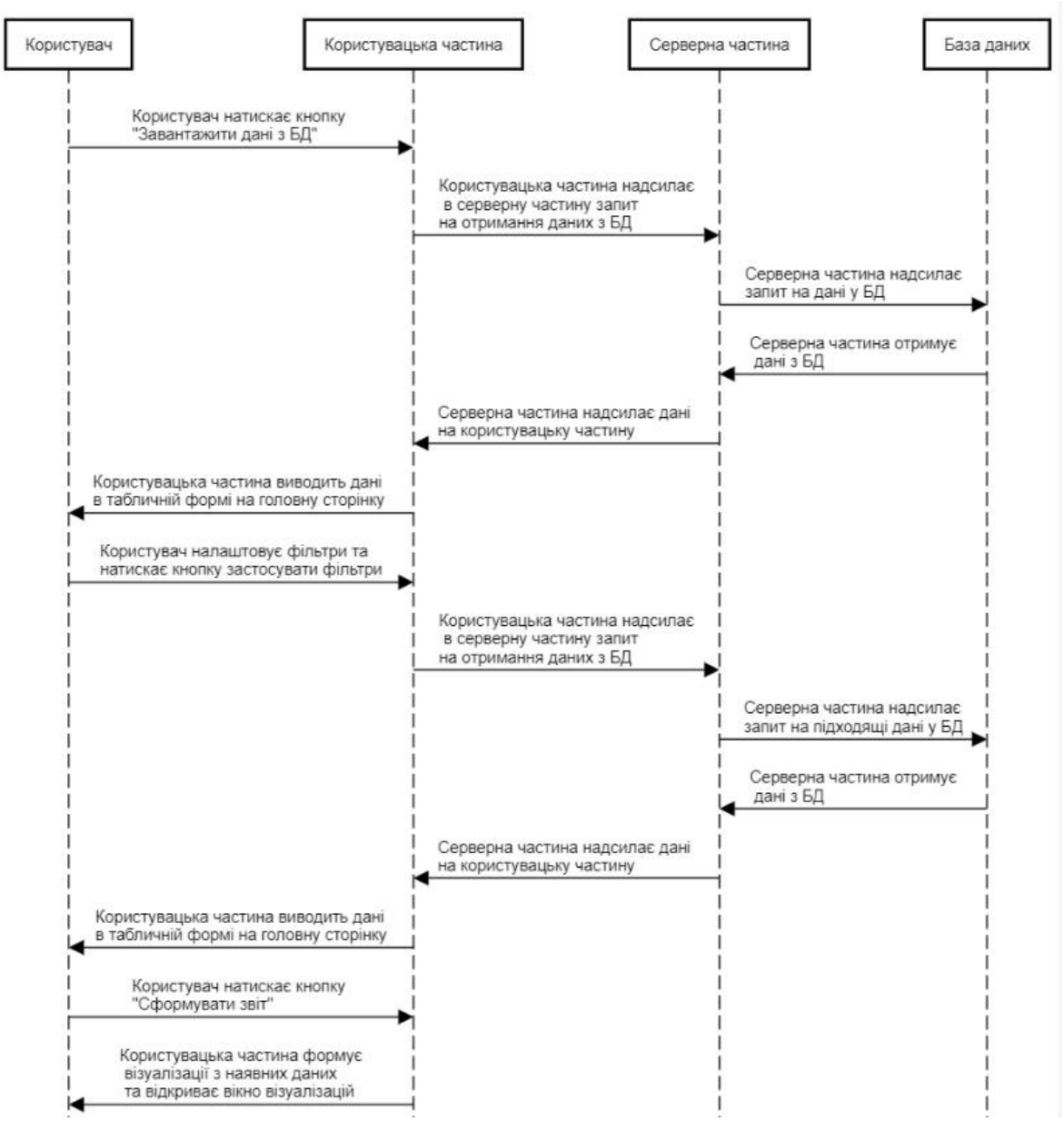


Рисунок 3.5 – Діаграма послідовності

Як продемонстровано на діаграмі, усі частини системи взаємодіють між собою для забезпечення швидкого доступу до інформації та забезпечення користувача швидкою роботою.

3.2 Структура системи

Структура системи автоматичної генерації звіту динаміки показників цін на агропродукцію складається з кількох ключових компонентів, які взаємодіють між собою для забезпечення ефективної роботи та надійності. Компоненти найвищого рівня включають серверну частину, користувацьку частину та базу даних. Ці частини містять програмні модулі, кожен з яких виконує власне завдання для забезпечення стабільного та ефективного виконання поставленої задачі відповідно до шаблону проектування Model-View-Controller (MVC).

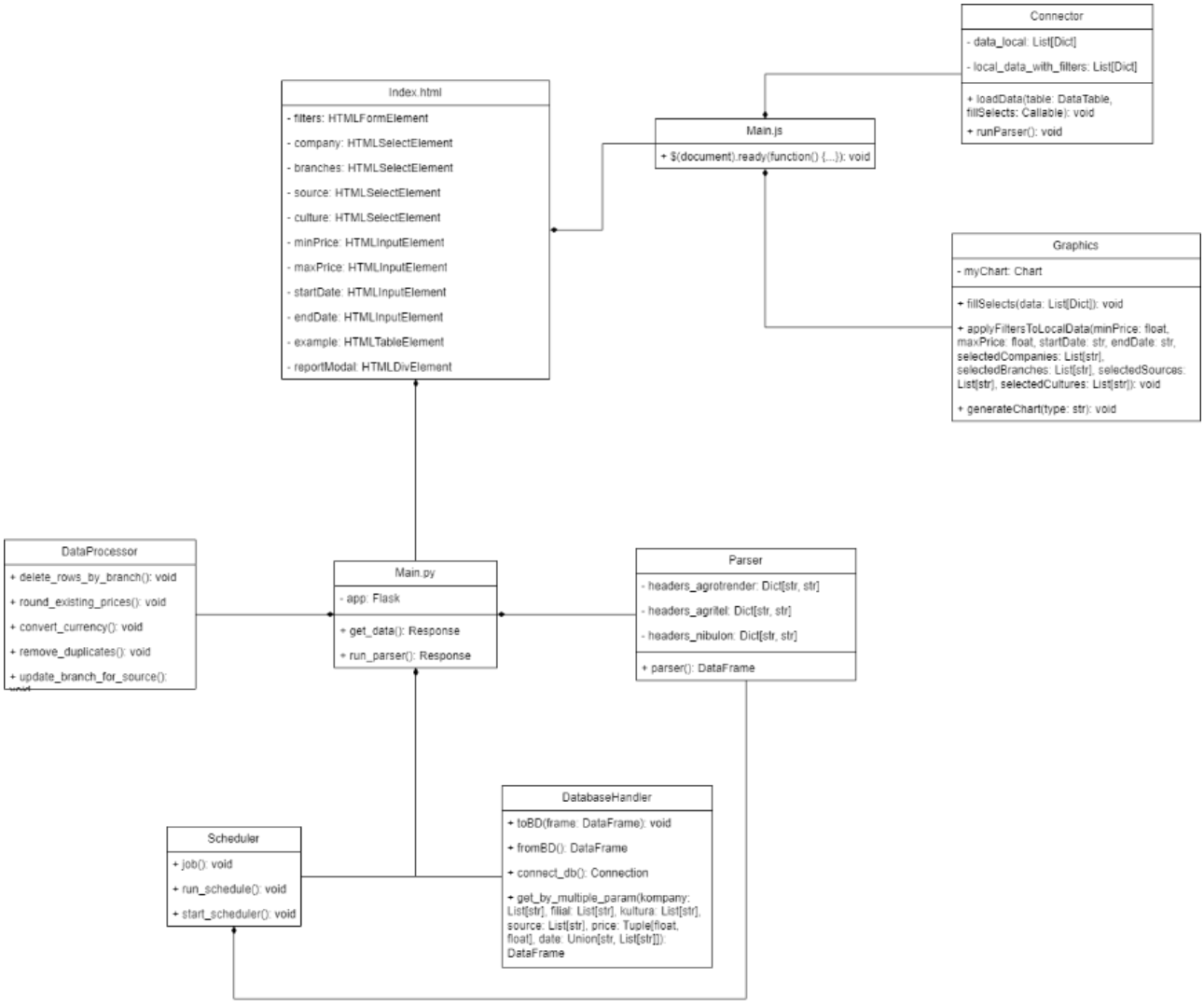


Рисунок 3.6 – Діаграма класів системи

Як продемонстровано на діаграмі, усі частини системи взаємодіють між собою для забезпечення швидкого доступу до інформації та забезпечення користувача швидкою роботою.

3.2.1 Структура серверної частини системи

Серверна частина системи містить такі модулі:

- модуль збору даних: Цей модуль відповідає за надсилання HTTP-запитів до цільових веб-сайтів та отримання HTML-коду сторінок. Для цього використовується бібліотека requests, яка дозволяє легко виконувати GET-запити та отримувати відповіді від серверів. Модуль збору даних також обробляє можливі помилки при з'єднанні та повторні запити у разі невдачі;

- модуль парсингу: Після отримання HTML-коду, модуль парсингу витягує необхідну інформацію з веб-сторінок. Для цього використовується бібліотека BeautifulSoup, яка дозволяє ефективно аналізувати та обробляти HTML-документи. Парсери знаходять відповідні елементи на сторінці (наприклад, таблиці цін, назви культур, компанії, регіони) та витягують їх значення;

- модуль обробки даних: Витягнуті дані проходять подальшу обробку, включаючи очищення, нормалізацію та валідацію. Очищення даних включає видалення зайвих пробілів, спеціальних символів та форматування значень. Нормалізація даних забезпечує приведення значень до єдиного формату, що полегшує їх подальшу обробку. Валідація даних перевіряє їх відповідність очікуваним критеріям, таким як правильний формат дат або коректні числові значення;

- модуль зберігання даних: Оброблені дані зберігаються у базі даних. Для цього використовується бібліотека SQLAlchemy, яка забезпечує зручний інтерфейс для роботи з базами даних у Python. Парсери додають нові записи до таблиць або оновлюють існуючі записи, якщо такі вже є у базі даних. Це дозволяє підтримувати актуальність та цілісність зібраної інформації;

- модуль планування завдань: Для регулярного виконання парсинг-завдань використовується бібліотека schedule, яка дозволяє налаштовувати розклад запуску парсерів. Модуль планування завдань забезпечує автоматичний запуск парсерів у визначений час (наприклад, щодня о 9:00 ранку), що гарантує своєчасне оновлення даних.

Алгоритм взаємодії модулів серверної частини системи при автоматичному зборі та збереженні інформації:

- ініціалізація та налаштування: Модулі ініціалізуються з необхідними налаштуваннями, такими як цільові веб-сайти, заголовки HTTP-запитів, параметри підключення до бази даних тощо. Цей етап також включає налаштування розкладу для автоматичного запуску парсерів;

- надсилання HTTP-запитів: Модуль збору даних надсилає HTTP-запити до цільових веб-сайтів, використовуючи налаштовані заголовки та параметри запитів. Бібліотека requests дозволяє гнучко керувати запитами, включаючи обробку редиректів, кукіз та інші аспекти HTTP-з'єднань;

- отримання та обробка відповідей: Отримані відповіді у вигляді HTML-документів передаються до модуля парсингу для подальшої обробки. У випадку помилок або невдач при отриманні відповіді, парсери можуть повторити запит або зареєструвати помилку для подальшого аналізу;

- парсинг HTML-документів: Модуль парсингу аналізує HTML-документи, витягує необхідну інформацію та перетворює її у структурований вигляд. Використання бібліотеки BeautifulSoup дозволяє легко знаходити та обробляти потрібні елементи на сторінці, такі як таблиці, заголовки, посилання та інші HTML-елементи;

- очищення та нормалізація даних: Витягнуті дані проходять через процес очищення, в модулі обробки даних, який включає видалення зайвих пробілів, спеціальних символів та інших небажаних елементів. Нормалізація даних забезпечує приведення значень до єдиного формату, що полегшує їх подальшу обробку та аналіз;

- валідація даних: В модулі обробки даних оброблені дані перевіряються на відповідність очікуваним критеріям. Це включає перевірку коректності форматів дат, числових значень, наявності необхідних полів тощо. Валідація допомагає виявити та виправити помилки на ранніх етапах обробки;

- збереження даних у базі: Модуль запису даних в базу даних записує дані, що пройшли всі етапи обробки, у базу даних. Використання SQLAlchemy забезпечує зручний функціонал для додавання нових записів або оновлення існуючих. Модуль обробки даних також може виконувати додаткові перевірки перед збереженням даних, наприклад, перевірку унікальності записів;

- регулярне виконання парсинг-завдань: Модуль планування завдань забезпечує автоматичний збір даних відповідно до налаштованого розкладу. Це гарантує регулярне оновлення даних без необхідності ручного втручання.

3.2.2 Структура користувацької частини системи

Модулі користувацької частини системи:

- модуль розмітки сторінки: Основу веб-інтерфейсу складають HTML-сторінки, які визначають структуру та розташування елементів на веб-сторінці. HTML забезпечує розмітку контенту, включаючи заголовки, таблиці, форми, кнопки та інші елементи інтерфейсу. Правильна організація HTML-документів забезпечує зрозумілу ієрархію та логіку веб-сторінок;

- модуль стилю сторінки: Для покращення зовнішнього вигляду та підвищення зручності користувацького інтерфейсу використовується CSS. Стилі CSS відповідають за оформлення елементів веб-сторінки, визначаючи кольори, шрифти, відступи, межі та інші візуальні аспекти. Використання адаптивного дизайну забезпечує коректне відображення інтерфейсу на різних пристроях, включаючи комп'ютери, планшети та смартфони;

- модуль формування візуалізацій: Для візуалізації даних використовується бібліотека Chart.js, яка дозволяє створювати різноманітні типи графіків та діаграм.

Chart.js забезпечує інтерактивність графіків, дозволяючи користувачам взаємодіяти з даними в режимі реального часу. Це включає можливість зміни параметрів відображення, фільтрації даних та деталізації інформації;

- модуль взаємодії з серверною частиною: Веб-інтерфейс взаємодіє з серверною частиною системи за допомогою AJAX-запитів, що дозволяє динамічно отримувати та відправляти дані. Серверна частина, побудована на Flask, відповідає за обробку цих запитів, виконання необхідних операцій та повернення результатів у форматі JSON. Це забезпечує швидку та ефективну комунікацію між клієнтською та серверною частинами системи;

Взаємодія модулів при роботі клієнтської частини:

- завантаження сторінки: Користувач відкриває веб-сторінку, модулі, що відповідають за розмітку та стиль формують веб-сторінку. Сторінка завантажується у веб-браузері, забезпечуючи базову структуру та оформлення інтерфейсу;

- ініціалізація інтерфейсу: Після завантаження сторінки, модулі взаємодії з сервером та формування візуалізацій формують динамічні елементи інтерфейсу, такі як графіки, таблиці та форми. Це включає налаштування параметрів відображення, підключення обробників подій та взаємодію з сервером;

- отримання даних з сервера: При взаємодії користувача з інтерфейсом (наприклад, при виборі фільтрів або запуску парсера), модуль взаємодії з сервером надсилає AJAX-запити до серверної частини системи. Сервер обробляє запити, виконує необхідні операції та повертає результати у форматі JSON;

- оновлення інтерфейсу: Отримані дані використовуються модулями формування візуалізацій для оновлення відображення інтерфейсу. Наприклад, нові дані можуть бути відображені на графіках, в таблицях або інших елементах інтерфейсу. JavaScript динамічно оновлює контент без перезавантаження сторінки, забезпечуючи плавну та інтерактивну взаємодію;

- обробка дій користувача: Веб-інтерфейс реагує на дії користувача, такі як натискання кнопок, вибір значень у формах, зміна фільтрів тощо. модулі виконують

відповідні дії, наприклад, надсилають нові запити до сервера, оновлюють відображення даних або змінюють параметри графіків.

Таким чином, структура веб-інтерфейсу забезпечує зручну та інтуїтивно зрозумілу взаємодію користувачів з системою автоматичної генерації звіту динаміки показників цін на агропродукцію. Вона поєднує статичні HTML та CSS з динамічним JavaScript, створюючи інтерактивний інтерфейс, який легко адаптується до потреб користувачів та забезпечує ефективну роботу з даними.

3.3 Структура бази даних

Структура бази даних у системі автоматичної генерації звіту динаміки показників цін на агропродукцію побудована на основі СУБД MySQL і включає одну основну таблицю для зберігання та управління даними. Основною таблицею є grain_prices, яка зберігає інформацію про ціни на агропродукцію.

Таблиця grain_prices має такі поля:

- ID: Поле типу int, що є унікальним ідентифікатором кожного запису. Це ключове поле забезпечує унікальність кожного рядка в таблиці та використовується для швидкої ідентифікації записів;

- Ціна: Поле типу text, що містить значення ціни на агропродукцію. Це поле включає числові значення, які можуть бути як цілими, так і дробовими;

- Валюта: Поле типу text, що вказує валюту, в якій зазначена ціна (наприклад, гривня або долар США). Це поле важливе для правильного тлумачення цінових значень;

- Компанія: Поле типу text, що містить назву компанії, яка надає цінову інформацію. Це дозволяє відстежувати дані за джерелами та аналізувати їх відповідно до постачальника інформації;

- Філія: Поле типу text, що містить інформацію про регіон або філію компанії, де була зібрана цінова інформація. Це допомагає в регіональному аналізі даних;

- Period: Поле типу text, що зберігає дату, на яку відноситься цінова інформація. Це дозволяє проводити аналіз динаміки цін у часі;

- Source: Поле типу text, що вказує джерело даних, з якого була отримана інформація (наприклад, веб-сайт або інший ресурс). Це важливо для прозорості та перевірки даних;

- Культура: Поле типу text, що містить назву культури, наприклад, кукурудза, пшениця або соя. Це поле дозволяє класифікувати дані за типами продукції.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ID	INT	☐	☑	☐	☐	☐	☐	☐	☐	'1'
Ціна	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Валюта	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Компанія	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Філія	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Period	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Source	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
Культура	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Рисунок 3.7 – інформація про поля таблиці grain_prices

ID	Ціна	Валюта	Компанія	Філія	Period	Source	Культура
1	7000	uah/t	АГРОФОНД-2022	Чорноморський МП	2024-05-06	Агротрендер	Кукурудза
2	5451	uah/t	АГРОФОНД-2022	Ізмаїльський МП	2024-05-06	Агротрендер	Кукурудза
4	6912	uah/t	AGS GRAIN	Ренійський МП	2024-05-06	Агротрендер	Кукурудза
5	7440	uah/t	SGI Trade AG	Ренійський МП	2024-05-06	Агротрендер	Кукурудза
6	6900	uah/t	SGI Trade AG	Чорноморський МП	2024-05-06	Агротрендер	Кукурудза
7	6518	uah/t	AGS GRAIN	Південний МП	2024-05-06	Агротрендер	Кукурудза
8	7700	uah/t	ГРАНОВА УКРАЇНА	Ізмаїльський МП	2024-05-06	Агротрендер	Кукурудза
9	7300	uah/t	ГРАНОВА УКРАЇНА	Чорноморський МП	2024-05-06	Агротрендер	Кукурудза

Рисунок 3.8 – Вигляд таблиці grain_prices

Ця таблиця є центральним сховищем для всіх зібраних даних про ціни на агропродукцію, забезпечуючи структуру, яка дозволяє легко додавати, оновлювати та запитувати інформацію.

Для забезпечення цілісності та узгодженості даних у базі використовуються індекси, що прискорюють пошук та сортування записів, а також зв'язки між таблицями. Наприклад, поле ID є основним ключем, що гарантує унікальність кожного запису.

Таким чином, структура бази даних в системі автоматичної генерації звіту динаміки показників цін на агропродукцію забезпечує ефективне зберігання, організацію та управління великими обсягами даних, підтримуючи високу продуктивність та надійність роботи системи.

3.4 Потоки даних у системі

Потоки даних у системі автоматичної генерації звіту динаміки показників цін на агропродукцію забезпечують ефективну обробку та передачу інформації від джерел збору до кінцевого користувача.

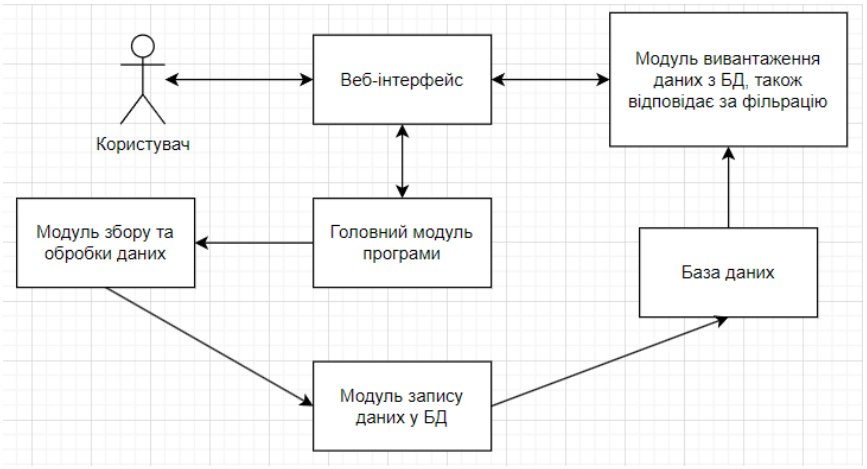


Рисунок 3.9 - блок-схема інформаційних потоків

Розглянемо детальніше, як дані переміщаються у програмі, починаючи від моменту їх збору і до моменту відображення у веб-інтерфейсі.

1. Збір даних: Процес починається з автоматичного збору даних за допомогою парсерів. Парсери регулярно запускаються відповідно до розкладу, визначеного у модулі планування завдань. Використовуючи бібліотеку requests, парсери надсилають HTTP-запити до цільових веб-сайтів, що містять інформацію про ціни на агропродукцію. Отримані HTML-документи обробляються за допомогою бібліотеки BeautifulSoup, яка витягує необхідні дані.

2. Обробка та нормалізація даних: Отримані дані проходять через кілька етапів обробки. Спочатку дані очищаються від зайвих символів, пробілів та інших небажаних елементів. Потім проводиться нормалізація даних, щоб привести їх до єдиного формату, що включає перетворення валют, округлення числових значень та форматування дат. Після цього дані валідуються для забезпечення їх відповідності очікуваним критеріям.

3. Збереження даних у базі: Оброблені дані зберігаються у базі даних MySQL. Використовуючи бібліотеку SQLAlchemy, дані додаються до таблиці grain_prices, яка містить інформацію про ціни, культури, компанії, філії, дати та джерела. Цей процес включає додавання нових записів або оновлення існуючих, якщо такі вже є у базі даних.

4. Взаємодія з сервером: Коли користувач взаємодіє з веб-інтерфейсом, наприклад, вибираючи фільтри або запитуючи оновлення даних, браузер надсилає AJAX-запити до серверної частини додатка, реалізованої за допомогою Flask. Сервер отримує запити та виконує відповідні операції з базою даних, отримуючи необхідні дані.

5. Форматування та передача даних: Отримані з бази даних результати форматуються у формат JSON за допомогою функції jsonify(). Це забезпечує стандартизований спосіб передачі даних від сервера до клієнта. JSON-формат легко інтерпретується клієнтськими скриптами, що дозволяє швидко та ефективно оновлювати вміст веб-сторінки без її перезавантаження.

6. Відображення даних у веб-інтерфейсі: Форматовані у JSON дані передаються клієнту, де вони обробляються JavaScript-скриптами для оновлення відображення. Дані можуть бути відображені у вигляді таблиць, графіків або діаграм за допомогою бібліотеки Chart.js. Інтерактивні елементи інтерфейсу дозволяють користувачам взаємодіяти з даними, застосовувати фільтри та отримувати детальну інформацію за обраними параметрами.

Таким чином, потоки даних у системі забезпечують ефективну обробку та передачу інформації від моменту її збору парсерами до відображення у веб-інтерфейсі. Цей процес включає очищення, нормалізацію, збереження, запити до бази даних, форматування та відображення даних, що забезпечує актуальність та точність наданої інформації користувачам.

3.5 Точки взаємодії з системою

У системі автоматичної генерації звіту динаміки показників цін на агропродукцію реалізовано дві основні точки взаємодії (маршрути) для комунікації між клієнтською та серверною частинами додатка. Ці маршрути обробляють HTTP-запити та забезпечують відповідні функціональні можливості. Маршрути та їх опис представлені у таблиці 3.1

Таблиця 3.1 - Таблиця точок взаємодії (маршрутів)

Маршрут	Метод	Опис
/get_data	GET	Використовується для отримання даних з бази даних. Повертає всю інформацію у форматі JSON, включаючи ціни на агропродукцію, назви компаній, філій, дати та джерела. Цей маршрут викликається, коли користувач завантажує дані для перегляду або аналізу.

Кінець таблиці 3.1

/run_parser	POST	Запускає процес парсингу для збору нових даних з веб-джерел. Цей маршрут викликається, коли користувач хоче оновити дані в базі даних, запустивши парсери для отримання актуальної інформації. Після успішного виконання парсингу нові дані додаються до бази даних.

3.6 Використані мови програмування, фреймворки та бібліотеки

Для розробки системи автоматичної генерації звіту динаміки показників цін на агропродукцію було використано кілька мов програмування, кожна з яких виконувала свою специфічну роль у різних компонентах системи. Це забезпечило гнучкість та ефективність розробки, дозволяючи створити інтегроване рішення, що відповідає усім вимогам проекту.

Основною мовою програмування для серверної частини та парсерів була обрана Python. Ця мова забезпечує легкість написання коду, широкий набір бібліотек для веб-розробки та обробки даних, а також можливість швидкої інтеграції з базами даних.

Для реалізації клієнтської частини та створення динамічного веб-інтерфейсу використовувалася мова JavaScript. Вона дозволила створити інтерактивні елементи інтерфейсу, які реагують на дії користувачів, забезпечуючи зручність та інтуїтивність взаємодії.

Статична структура веб-інтерфейсу була створена за допомогою HTML, що забезпечує базову розмітку та організацію контенту на веб-сторінках. Для покращення зовнішнього вигляду та забезпечення адаптивного дизайну використовувалася мова стилів CSS.

Разом ці мови програмування створили цілісну систему, що забезпечує ефективний збір, обробку, зберігання та відображення даних про ціни на

агропродукцію. Кожна з мов виконувала свою роль у відповідних компонентах системи, забезпечуючи їх функціональність та зручність використання.

3.6.1 Мова програмування Python

Python був обраний основною мовою програмування для серверної частини та парсерів у системі автоматичної генерації звіту динаміки показників цін на агропродукцію. Python забезпечує ряд переваг, таких як легкість написання коду, висока читабельність та велика кількість доступних бібліотек, що значно полегшує розробку та підтримку складних систем [7].

Однією з ключових бібліотек, що використовуються у проекті, є Flask, мікрофреймворк для веб-розробки. Flask дозволяє швидко створювати веб-додатки та API, що забезпечують обробку запитів, маршрутизацію та взаємодію з базою даних. Це дозволяє ефективно обробляти запити від клієнтської частини, надаючи дані у форматі JSON.

Парсери, які збирають дані з веб-джерел, також написані на Python. Використовуючи бібліотеки BeautifulSoup та requests, вони забезпечують надійний збір, аналіз та обробку HTML-документів. BeautifulSoup дозволяє легко витягувати необхідні дані з веб-сторінок, тоді як requests забезпечує зручний інтерфейс для надсилання HTTP-запитів та отримання відповідей.

Python також використовується для обробки та нормалізації даних перед їх збереженням у базі даних. Це включає конвертацію валют, округлення числових значень та перевірку відповідності даних визначеним критеріям. Використання бібліотеки SQLAlchemy забезпечує зручну роботу з базою даних MySQL, дозволяючи легко виконувати операції додавання, оновлення та видалення записів.

Список усіх фреймворків та бібліотек, що використовуються з мовою Python:

- Requests - Бібліотека для відправки HTTP-запитів, використовується для отримання даних з веб-сайтів [8];

- BeautifulSoup - Бібліотека для парсингу HTML та XML документів, використовується для витягування потрібних даних з отриманих веб-сторінок [9];

- Pandas - Потужна бібліотека для аналізу та обробки даних, використовується для обробки, аналізу та маніпуляції табличними даними [10];

- SQLAlchemy - ORM (Object-Relational Mapping) бібліотека для роботи з базами даних, використовується для взаємодії з базою даних MySQL [11];

- Flask - Мікрофреймворк для створення веб-додатків, використовується для створення веб-сервера та API для доступу до даних [12];

- Subprocess - Бібліотека для запуску нових процесів та взаємодії з ними, використовується для створення бекапів бази даних [13];

- Schedule - Бібліотека для роботи з плануванням завдань, використовується для автоматизації збору та збереження даних [14].

3.6.2 Мова програмування JavaScript

JavaScript відіграє ключову роль у реалізації клієнтської частини системи, забезпечуючи створення динамічного та інтерактивного веб-інтерфейсу. Використання JavaScript дозволяє реалізувати функціональні можливості, які забезпечують зручну та інтуїтивно зрозумілу взаємодію користувачів з системою [15].

JavaScript забезпечує динамічне оновлення контенту на веб-сторінках без необхідності перезавантаження сторінки. Це досягається за допомогою AJAX-запитів, які дозволяють взаємодіяти з серверною частиною системи, отримуючи та відправляючи дані у форматі JSON. Такий підхід забезпечує швидке реагування на дії користувачів та покращує загальну продуктивність веб-додатка.

Для візуалізації даних у вигляді графіків та діаграм використовується бібліотека Chart.js. Вона дозволяє створювати різноманітні типи графіків, включаючи лінійні графіки, гістограми, кругові діаграми та інші. Chart.js забезпечує інтерактивність графіків, дозволяючи користувачам взаємодіяти з даними, змінювати параметри відображення та аналізувати результати.

JavaScript також використовується для реалізації обробників подій, які реагують на дії користувачів, такі як натискання кнопок, вибір значень у формах та інші інтерактивні елементи. Це забезпечує високу інтерактивність та гнучкість веб-інтерфейсу, дозволяючи користувачам ефективно взаємодіяти з системою.

Таким чином, JavaScript є невід'ємною частиною клієнтської частини системи, забезпечуючи динамічне оновлення контенту, інтерактивну візуалізацію даних та зручну взаємодію користувачів з веб-інтерфейсом.

Список усіх фреймворків та бібліотек, що використовуються з мовою JavaScript:

jQuery - Швидка, невелика та багатофункціональна бібліотека JavaScript, використовується для маніпуляції DOM та виконання AJAX запитів [16].

DataTables - Плагін для jQuery, який додає інтерактивні функції до HTML-таблиць, використовується для інтерактивного відображення даних у таблицях [17].

Chart.js - Простий у використанні плагін для створення графіків, використовується для візуалізації даних у вигляді графіків [18].

Select2 - Плагін для jQuery, який покращує роботу з селекторами, використовується для створення зручних багатофункціональних селекторів [19].

3.6.3 Мова розмітки HTML

HTML (HyperText Markup Language) є основною мовою розмітки, що використовується для створення структури веб-сторінок у системі автоматичної генерації звіту динаміки показників цін на агропродукцію. HTML забезпечує базову розмітку контенту, визначаючи структуру та компонування елементів веб-інтерфейсу [20].

Використання HTML дозволяє організувати різні типи даних у зручному та зрозумілому вигляді. Кожен елемент інтерфейсу, такий як таблиці, форми, заголовки, параграфи, кнопки та графіки, визначається за допомогою відповідних HTML-тегів.

Це дозволяє створити логічну та чітку структуру веб-сторінок, яка легко сприймається користувачами.

HTML також забезпечує інтеграцію інших мов програмування та технологій, таких як CSS для оформлення та JavaScript для додавання інтерактивності. Завдяки HTML можливе створення адаптивного дизайну, який коректно відображається на різних пристроях, включаючи комп'ютери, планшети та смартфони.

У системі автоматичної генерації звіту HTML використовується для створення сторінок, які відображають дані про ціни на агропродукцію у вигляді таблиць та графіків. Це забезпечує зручний доступ до інформації та легку навігацію по веб-інтерфейсу.

3.6.4 Мова розмітки CSS

CSS (Cascading Style Sheets) використовується для оформлення веб-сторінок, визначених за допомогою HTML, і грає важливу роль у створенні естетично привабливого та зручного інтерфейсу користувача. CSS дозволяє розділити візуальне оформлення від структури контенту, забезпечуючи гнучкість та можливість легко змінювати зовнішній вигляд веб-сторінок без впливу на їх зміст [21].

За допомогою CSS можна налаштовувати кольори, шрифти, відступи, межі, розташування елементів та багато інших аспектів оформлення. Це дозволяє створювати єдиний стиль для всього веб-додатка, що покращує користувацький досвід і робить інтерфейс більш привабливим та зручним для використання.

CSS також підтримує адаптивний дизайн, який дозволяє веб-сторінкам коректно відображатися на різних пристроях та екранах різного розміру. Це досягається за допомогою медіа-запитів, які дозволяють застосовувати різні стилі в залежності від розмірів вікна браузера або типу пристрою.

У системі автоматичної генерації звіту CSS використовується для оформлення таблиць, форм, кнопок та інших елементів інтерфейсу, а також для створення

адаптивних макетів, які забезпечують зручну навігацію та доступ до інформації незалежно від використовуваного пристрою.

Список усіх фреймворків та бібліотек, що використовуються з мовою CSS:

- Bootstrap - Фреймворк для розробки адаптивного та мобільного веб-дизайну, використовується для створення адаптивного дизайну та стилю веб-сторінки [22].

3.7 Використані середовища розробки та системи управління

Для розробки та підтримки системи автоматичної генерації звіту динаміки показників цін на агропродукцію було використано кілька інструментів, які забезпечили ефективність та зручність роботи на всіх етапах проекту. Ці інструменти допомогли створити, тестувати та підтримувати систему, забезпечуючи її надійність та продуктивність.

3.7.1 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) — це потужне середовище розробки, яке було використано для написання та редагування коду у проекті автоматичної генерації звіту динаміки показників цін на агропродукцію. Цей інструмент став незамінним завдяки своїм численним можливостям та розширенням, що забезпечують зручну та ефективну роботу з різними мовами програмування та фреймворками.

VS Code підтримує широкий спектр мов програмування, включаючи Python та JavaScript, що робить його ідеальним вибором для розробки як серверної, так і клієнтської частин системи. Редактор надає зручний інтерфейс для навігації по проекту, швидкого пошуку та заміни тексту, а також інтелектуальні підказки та автозавершення коду, що значно підвищує продуктивність роботи [23].

Однією з ключових переваг VS Code є його розширюваність. Розробники можуть встановлювати численні розширення, що додають нові можливості та

поліпшують робочий процес. Наприклад, для роботи з Python були використані розширення Python та Pylance, які забезпечують підтримку синтаксису, налагодження, запуск тестів та багато інших функцій. Для роботи з JavaScript та веб-розробкою використовувалися розширення, такі як ESLint для перевірки коду та Prettier для автоматичного форматування.

Вбудований термінал у VS Code дозволяє розробникам виконувати команди безпосередньо з редактора, що полегшує роботу з системою контролю версій Git, запуск локальних серверів, встановлення пакетів та виконання інших команд. Це дозволяє зосередитися на написанні коду, не переключаючись між різними інструментами.

Налагодження коду у VS Code також забезпечено на високому рівні. Інструмент підтримує налаштування точок зупину, моніторинг змінних, покрокове виконання коду та інші функції налагодження, що дозволяють швидко знаходити та виправляти помилки.

Таким чином, Visual Studio Code став основним інструментом для розробки веб-інтерфейсу, забезпечуючи зручне та ефективне середовище для написання, редагування та налагодження коду. Його розширюваність, підтримка численних мов програмування та інтеграція з іншими інструментами зробили його незамінним вибором для цього проекту.

3.7.2 Середовище розробки PyCharm

PyCharm — це інтегроване середовище розробки (IDE), спеціально призначене для роботи з Python, і воно було використане в проекті автоматичної генерації звіту динаміки показників цін на агропродукцію. Це середовище надає розширені можливості для написання, налагодження та тестування Python-коду, що робить його ідеальним вибором для розробки серверної частини та парсерів [24].

PyCharm пропонує інтелектуальні інструменти для написання коду, такі як автозавершення, рефакторинг та перевірка синтаксису в реальному часі. Ці функції

значно підвищують продуктивність та точність розробників, дозволяючи швидко виявляти та виправляти помилки ще на етапі написання коду.

Однією з ключових можливостей PyCharm є вбудована підтримка налагодження. Розробники можуть встановлювати точки зупину, виконувати покрокове налагодження, моніторити змінні та виконувати інші дії, що полегшують виявлення та виправлення помилок у коді. Це забезпечує глибоке розуміння того, як виконується код, і допомагає знаходити складні помилки.

PyCharm також надає потужні інструменти для тестування, що дозволяють легко створювати та виконувати юніт-тести. Вбудовані функції для роботи з фреймворками тестування, такими як unittest та pytest, дозволяють автоматизувати процес тестування та забезпечують високу якість коду. Це важливо для підтримки стабільності та надійності системи.

Для роботи з базами даних PyCharm пропонує вбудовані інструменти, що дозволяють легко підключатися до різних баз даних, виконувати SQL-запити та переглядати результати. Це забезпечує зручний спосіб керування даними та інтеграції з базою даних MySQL, використаною у проекті.

Інтеграція з системами контролю версій, такими як Git, також є важливою особливістю PyCharm. Це дозволяє розробникам ефективно відстежувати зміни в коді, управляти гілками, виконувати злиття та інші операції, що забезпечують спільну роботу над проектом.

Таким чином, PyCharm став ключовим інструментом для розробки серверної частини та парсерів у проекті, забезпечуючи розширені можливості для написання, налагодження, тестування та управління кодом. Його потужні функції та інтеграція з іншими інструментами зробили розробку більш продуктивною та ефективною.

3.7.3 СКБД MySQL

MySQL — це система управління базами даних (СУБД), що є однією з найпопулярніших і найефективніших реляційних СУБД, використовуваних у багатьох

проектах для зберігання та керування даними. У cbcntvs автоматичної генерації звіту динаміки показників цін на агропродукцію MySQL була обрана завдяки своїй надійності, продуктивності та легкості у використанні.

MySQL надає розширені можливості для створення, збереження та маніпуляції даними. Вона підтримує широкий спектр типів даних та надає багатий набір функцій для роботи з даними, включаючи потужні інструменти для виконання SQL-запитів, транзакцій та збережених процедур [25].

Однією з ключових переваг MySQL є її висока продуктивність. Вона забезпечує швидку обробку запитів та оптимізацію роботи з великими обсягами даних, що особливо важливо для проектів з великим навантаженням на базу даних. MySQL використовує ефективні алгоритми індексації та кешування, що дозволяє зменшити час обробки запитів і покращити загальну продуктивність системи.

MySQL також відома своєю надійністю та стабільністю. Вона забезпечує захист даних за допомогою функцій резервного копіювання та відновлення, а також підтримує реплікацію для створення резервних копій баз даних. Це дозволяє зберігати дані в безпеці та мінімізувати ризик їх втрати у разі аварійних ситуацій.

Для розробників MySQL надає зручні інструменти адміністрування та керування базою даних. Інтерфейси, такі як MySQL Workbench, дозволяють легко створювати та модифікувати бази даних, виконувати запити та аналізувати результати. Це забезпечує зручний спосіб роботи з базою даних та інтеграцію з іншими інструментами розробки.

Таким чином, MySQL стала ключовим інструментом для зберігання та керування даними у проекті, забезпечуючи надійні, продуктивні та зручні можливості для роботи з базою даних.

4 РОБОТА КОРИСТУВАЧА З СИСТЕМОЮ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ ЗВІТУ ДИНАМІКИ ПОКАЗНИКІВ ЦІН НА АГРОПРОДУКЦІЮ

Тут продемонстровано, як користувач може взаємодіяти з програмною системою для збирання, обробки та візуалізації даних про ціни на зернові культури. Будуть розглянуті основні функціональні можливості інтерфейсу користувача, а також детальні інструкції щодо використання різних функцій системи. Додаток забезпечує зручний і зрозумілий веб-інтерфейс, що дозволяє легко отримувати потрібну інформацію та здійснювати необхідні операції.

4.1 Системні вимоги та інсталяція системи

Для інсталяції та запуску системи збирання, обробки та візуалізації даних про ціни на зернові культури необхідно виконати декілька вимог, які охоплюють як апаратне, так і програмне забезпечення. Мінімальні апаратні вимоги включають комп'ютер з процесором, що має не менше двох ядер, та 4 ГБ оперативної пам'яті, проте рекомендовані параметри — процесор з чотирма ядрами та 8 ГБ оперативної пам'яті для забезпечення оптимальної продуктивності системи. Операційна система повинна бути Windows 10 або новішої версії, macOS 10.14 або новішої версії, або сучасна дистрибуція Linux (наприклад, Ubuntu 20.04 або новіша).

Програмні вимоги включають встановлення наступних компонентів: Python 3.9 або новіша версія, пакетний менеджер pip для встановлення необхідних бібліотек, MySQL сервер версії 8.0 або новішої для зберігання даних. Також необхідно встановити кілька бібліотек Python, серед яких Flask для розробки веб-додатків, SQLAlchemy для роботи з базами даних, BeautifulSoup4 та requests для парсингу веб-

сторінок, pandas для обробки даних, а також бібліотеку Chart.js для візуалізації даних на стороні клієнта. Для коректного відображення веб-інтерфейсу слід використовувати сучасний веб-браузер, такий як Google Chrome (версія 90 або новіша), Mozilla Firefox (версія 88 або новіша) чи інший сумісний браузер.

Інсталяція системи починається з налаштування середовища розробки. Після встановлення Python та MySQL необхідно створити базу даних та таблиці відповідно до структури, описаної у розділі 4.2. Далі, за допомогою pip, слід встановити всі необхідні бібліотеки, виконавши команду pip install -r requirements.txt, де requirements.txt містить список всіх необхідних пакетів. Потім потрібно налаштувати файл конфігурації для підключення до бази даних, вказавши відповідні облікові дані. Після чого необхідно клонувати репозиторій, або завантажити його з сайту https://github.com: git clone https://github.com/diplom_grain_prices.git Для запуску серверної частини додатку необхідно виконати команду python main.py, яка запустить Flask-сервер на локальному хості. Веб-інтерфейс буде доступний за адресою http://localhost:5000, де користувач зможе взаємодіяти з системою, застосовувати фільтри та візуалізувати дані.

4.2 Інструкція з запуску системи

Для запуску програми необхідно в командному рядку перейти в папку, куди було клоновано, або завантажено репозиторій. Після чого необхідно запустити серверну частину додатку командою python main.py. Наступним кроком потрібно запустити веб-інтерфейс командою node app.js, як вказано на рисунку 4.1.

```
D:\Uni\Diplom\site>node app.js
Server working on port 8000
```

Рисунок 4.1 – Запуск веб інтерфейсу, та повернення порту

Успішне виконання цієї команди поверне нам порт, на якому запущено сервер. Наступним кроком потрібно відкрити браузер та ввести в адресну строку адресу http://localhost:5000, де 5000 потрібно замінити на число, яке нам повернула команда node app.js, як вказано на рисунку 4.2.

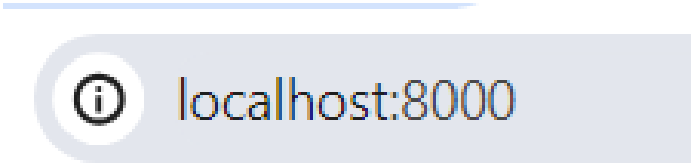


Рисунок 4.2 – Перехід за адресою

Успішне виконання цього кроку перенесе нас на стартову(вітальну) сторінку, де починається робота користувача з системою.

4.3 Методика роботи з системою автоматичної генерації звіту динаміки показників цін на агропродукцію

У цьому підрозділі буде розглянуто типову роботу користувача з програмним продуктом сисетми автоматичної генерації звітів про динаміку цін на агропродукцію. Буде продемонстровано типову роботу користувача, з усіма основними етапами, такими як завантаження нових даних з джерел, завантаження даних з БД у інтерфейс, формування візуалізацій та взаємодія з ними, а також експорт сформованих візуалізацій для друку, або збереження.

4.3.1 Огляд стартової сторінки

Робота з застосунком починається з вікна привітання, де коротко описане призначення та функціонал застосунку, як продемонстровано на рисунку 4.3.

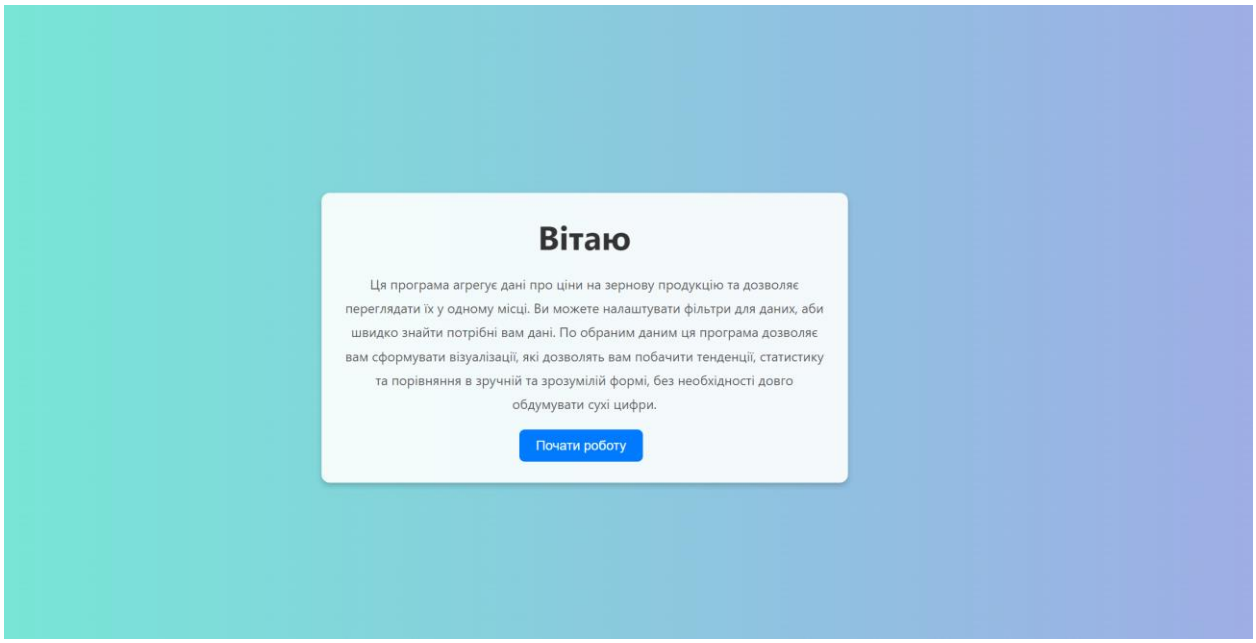


Рисунок 4.3 – Стартова сторінка та кнопка початку роботи

На стартовій сторінці потрібно натиснути кнопку «Почати роботу», після чого користувача перенесе на головну сторінку.

4.3.2 Огляд головної сторінки

Головна сторінка складається з декількох ключових елементів, а саме:

- Навігаційна панель: Навігаційна панель знаходиться зверху сторінки та містить основні кнопки для роботи з даними, а саме кнопка запуску парсерів, для отримання найновішої інформації та завантаження її в базу даних, кнопку завантаження даних з бази даних в застосунок, а також кнопку формування звітів для перегляду візуалізацій;

- Панель фільтрації даних: Панель фільтрації даних знаходиться з лівого боку сторінки та містить усі необхідні кнопки та поля вводу, щоб користувач міг швидко знайти необхідні йому дані;

- Таблиця з даними: Таблиця з даними знаходиться в центрі сторінки та забезпечує користувача відібраними по його умовам даними. Таблиця також містить

поле пошуку, кнопки переключення сторінок, сортування даних та зміни кількості рядків інформації на екрані одночасно.

Головна сторінка продемонстрована на рисунку 4.4.

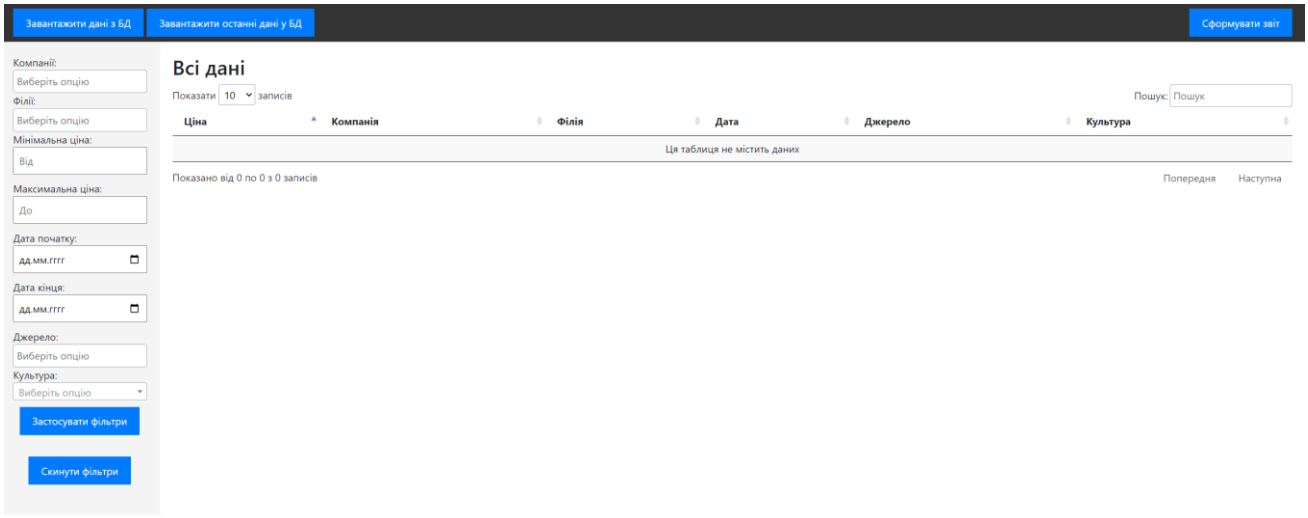


Рисунок 4.4 – Головна сторінка

4.3.3 Завантаження данних та запуск парсера

При завантаженні головної сторінки, на ній не буде жодної інформації, для завантаження інформації на сторінку необхідно натиснути відповідну кнопку в лівому верхньому кутку сторінки, кнопка продемонстрована на рисунку 4.5.

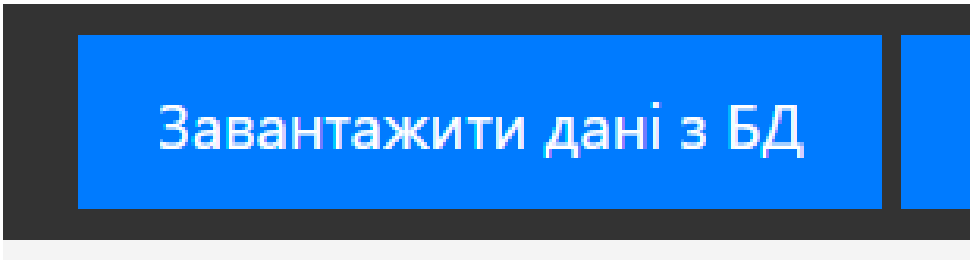
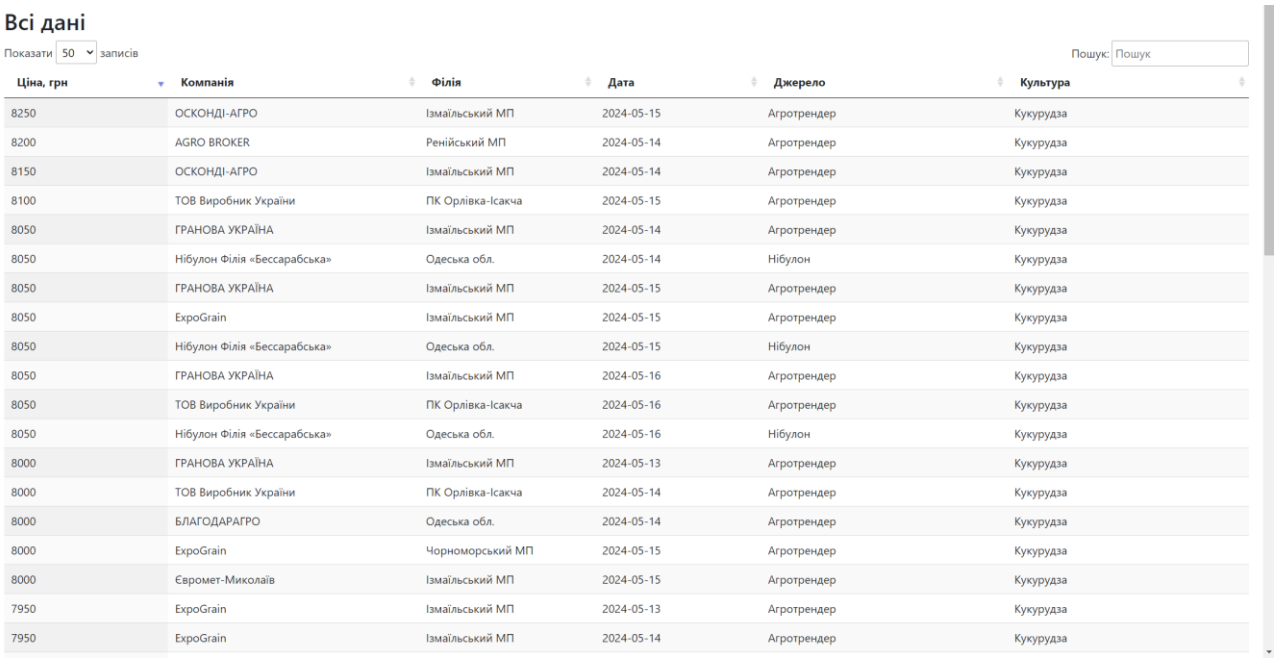


Рисунок 4.5 – Кнопка завантаження даних з БД

Після чого на сторінці зявляться табличні дані. По замовчуванню на сторінку виводяться усі дані. Це показано на рисунку 4.6.



Всі дані

Показати 50 записів

Пошук: Пошук

Ціна, грн	Компанія	Філія	Дата	Джерело	Культура
8250	ОСКОНДІ-АГРО	Ізмаїльський МП	2024-05-15	Агротрендер	Кукурудза
8200	AGRO BROKER	Ренійський МП	2024-05-14	Агротрендер	Кукурудза
8150	ОСКОНДІ-АГРО	Ізмаїльський МП	2024-05-14	Агротрендер	Кукурудза
8100	ТОВ Виробник України	ПК Орлівка-Ісакча	2024-05-15	Агротрендер	Кукурудза
8050	ГРАНОВА УКРАЇНА	Ізмаїльський МП	2024-05-14	Агротрендер	Кукурудза
8050	Нібулон Філія «Бессарабська»	Одеська обл.	2024-05-14	Нібулон	Кукурудза
8050	ГРАНОВА УКРАЇНА	Ізмаїльський МП	2024-05-15	Агротрендер	Кукурудза
8050	ExpoGrain	Ізмаїльський МП	2024-05-15	Агротрендер	Кукурудза
8050	Нібулон Філія «Бессарабська»	Одеська обл.	2024-05-15	Нібулон	Кукурудза
8050	ГРАНОВА УКРАЇНА	Ізмаїльський МП	2024-05-16	Агротрендер	Кукурудза
8050	ТОВ Виробник України	ПК Орлівка-Ісакча	2024-05-16	Агротрендер	Кукурудза
8050	Нібулон Філія «Бессарабська»	Одеська обл.	2024-05-16	Нібулон	Кукурудза
8000	ГРАНОВА УКРАЇНА	Ізмаїльський МП	2024-05-13	Агротрендер	Кукурудза
8000	ТОВ Виробник України	ПК Орлівка-Ісакча	2024-05-14	Агротрендер	Кукурудза
8000	БЛАГОДАРАГРО	Одеська обл.	2024-05-14	Агротрендер	Кукурудза
8000	ExpoGrain	Чорноморський МП	2024-05-15	Агротрендер	Кукурудза
8000	Євромет-Миколаїв	Ізмаїльський МП	2024-05-15	Агротрендер	Кукурудза
7950	ExpoGrain	Ізмаїльський МП	2024-05-13	Агротрендер	Кукурудза
7950	ExpoGrain	Ізмаїльський МП	2024-05-14	Агротрендер	Кукурудза

Рисунок 4.6 – Усі дані в табличній формі

Для отримання нових даних з джерел, в інтерфейсі присутня кнопка, при натисканні якої запуститься блок збору та обробки інформації, після чого нові дані будуть записані в БД. Кнопка продемонстрована на рисунку 4.7.

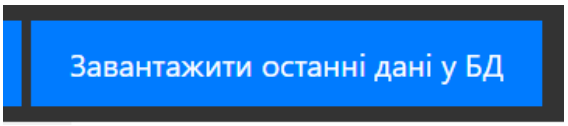


Рисунок 4.7 – Кнопка для отримання нових даних з джерел

4.3.4 Приклад фільтрування даних

В лівій частині сторінки знаходиться область фільтрів, яка містить функціонал для фільтрування даних по усім параметрам. Усі параметри та кнопки застосування та

скидання фільтрів продемонстровано на рисунку 4.8. Для фільтрів реалізований множинний вибір параметрів, з можливістю пошуку по заданому значенню у списку доступних значень.

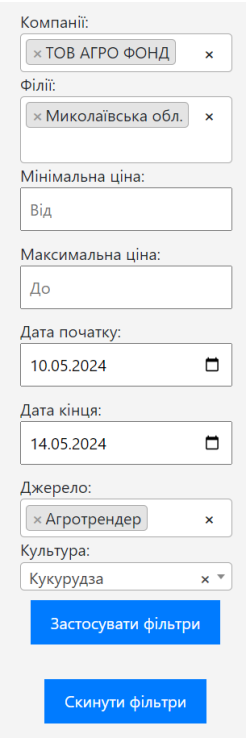


Рисунок 4.8 – Блок з фільтрами

Після налаштування усіх фільтрів необхідно натиснути кнопку “Застосувати фільтри”, що поверне на головну сторінку лише ті дані, які відповідають заданим критеріям. Це продемонстровано на рисунку 4.9.

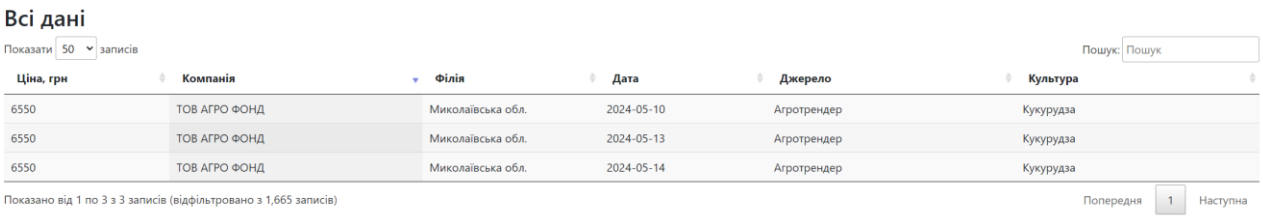


Рисунок 4.9 – Демонстрація відфільтрованих даних

4.3.5 Приклад генерації звіту

В правому верхньому кутку сторінки є кнопка, продемонстрована на рисунку 4.10.

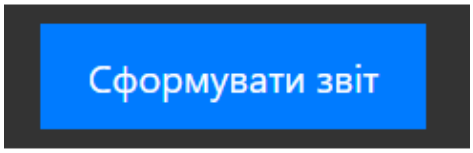


Рисунок 4.10 – Кнопка для формування звіту у формі візуалізацій

При натисканні цієї кнопки, відкриється вікно з візуалізаціями, сформованими з обраних користувачем даних, воно продемонстроване на рисунку 4.11.

Рисунок 4.11 – Вікно візуалізацій

У вікні з візуалізаціями можливе перемикання між різними візуалізаціями, всі вони є інтерактивними, тобто можливо наводити курсор та бачити точні дані, а також

можливо вимикати непотрібні дані. Ця можливість продемонстрована на рисунку 4.12.

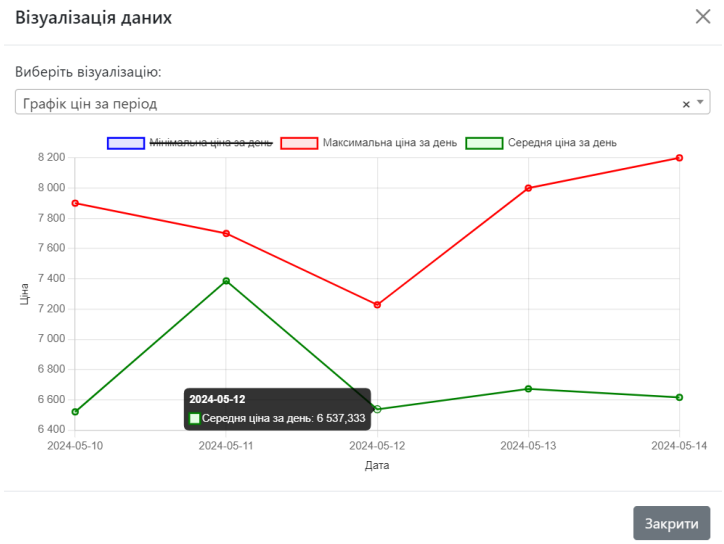


Рисунок 4.12 – Інтерактивність візуалізацій

4.3.6 Збереження та друк

В лівому нижньому кутку вікна візуалізацій є кнопка, продемонстрована на рисунку 4.13.

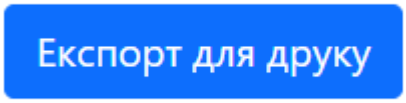


Рисунок 4.13 – Кнопка експорту для друку

При нажатті цієї кнопки обрана візуалізація відкриється в вікні для друку, на якому також можливо зберегти цю візуалізацію у форматі PDF. Це продемонстровано на рисунку 4.14.

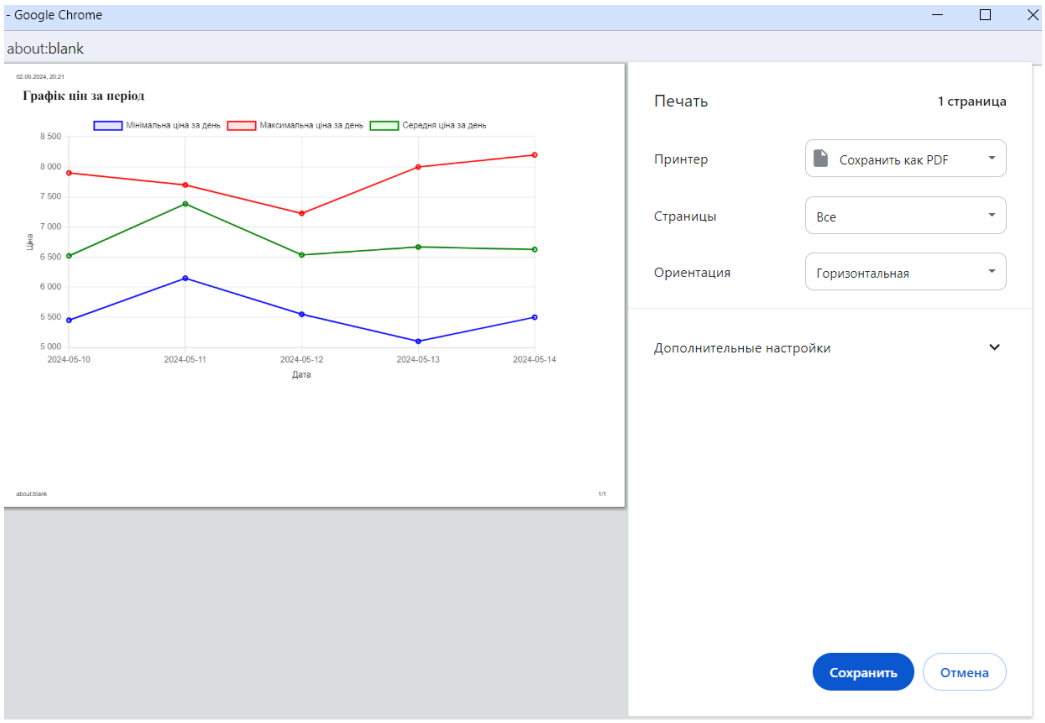


Рисунок 4.14 – Вікно експорту для друку

Як продемонстровано на рисунках, пограмний продукт відповідає всім критеріям завдання, має зручний, але не переобтяжений інтерфейс, яким буде в силі користуватись користувач будь якого рівня комп’ютерної грамотності.

ВИСНОВКИ

У сучасних умовах ринку зернової продукції інформація відіграє вирішальну роль у прийнятті рішень трейдерами. Для полегшення їхньої роботи була розроблена програма, яка забезпечує автоматизований збір та обробку даних з різних джерел, зберігає їх у базі даних та дозволяє аналізувати динаміку цін та інших показників на тривалий період часу. Завдяки цій програмі, трейдери можуть швидко та ефективно отримувати необхідну інформацію, що значно підвищує їхню продуктивність та ефективність.

Розроблена програма дозволяє відображати зібрані дані в табличному форматі з можливістю налаштовувати вибір необхідних даних. Крім того, програма надає функцію візуалізації даних, яка дозволяє користувачам налаштовувати відображення інформації відповідно до своїх потреб. Це дає змогу трейдерам приймати обґрунтовані рішення на основі комплексного аналізу даних.

Використання цієї програми дозволить досягти значного підвищення ефективності роботи трейдерів зернової продукції, забезпечуючи їм своєчасний доступ до актуальної інформації та зручні інструменти для її аналізу. Це, в свою чергу, посприяє оптимізації процесу прийняття рішень та покращенню загальної продуктивності роботи трейдерів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Agricensus. URL: https://www.agricensus.com (дата звернення: 04.06.2024).
2. Tableau. URL: https://www.tableau.com (дата звернення: 04.06.2024).
3. Power BI. URL: https://www.microsoft.com/en-us/power-platform/products/power-bi (дата звернення: 04.06.2024).
4. Agrotender. URL: https://agrotender.com.ua/ (дата звернення: 04.06.2024).
5. Agritel. URL: https://www.agritel.com/ua/ (дата звернення: 04.06.2024).
6. Nibulon. URL: https://www.nibulon.com/ (дата звернення: 04.06.2024).
7. Python. URL: https://www.python.org (дата звернення: 04.06.2024).
8. Requests. URL: https://requests.readthedocs.io/en/latest/ (дата звернення: 04.06.2024).
9. BeautifulSoup. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата звернення: 04.06.2024).
10. Pandas. URL: https://pandas.pydata.org/ (дата звернення: 04.06.2024).
11. SQLAlchemy. URL: https://www.sqlalchemy.org/ (дата звернення: 04.06.2024).
12. Flask. URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення: 04.06.2024).
13. Subprocess. URL: https://docs.python.org/3/library/subprocess.html (дата звернення: 04.06.2024).
14. Schedule. URL: https://schedule.readthedocs.io/en/stable/ (дата звернення: 04.06.2024).
15. Javascript. URL: https://www.javascript.com (дата звернення: 04.06.2024).
16. jQuery. URL: https://jquery.com/ (дата звернення: 04.06.2024).
17. DataTables. URL: https://datatables.net/ (дата звернення: 04.06.2024).
18. Chart.js. URL: https://www.chartjs.org/ (дата звернення: 04.06.2024).
19. Select2. URL: https://select2.org/ (дата звернення: 04.06.2024).
20. What is HTML? URL: https://www.hostinger.com/tutorials/what-is-html (дата звернення: 04.06.2024).

21. What is CSS? URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS (дата звернення: 04.06.2024).
22. Bootstrap. URL: https://getbootstrap.com/ (дата звернення: 04.06.2024).
23. Visual Studio Code. URL: https://code.visualstudio.com (дата звернення: 04.06.2024).
24. Pycharm. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 04.06.2024).
25. MySQL. URL: https://www.mysql.com/ (дата звернення: 04.06.2024).

ДОДАТОК А

Реалізація центрального модуля серверної частини системи

Текст програмного модуля

УКР.НТУУ”КПІ імені Ігора Сікорського”

Аркушів 4

Київ 2024

Програмні засоби:

– мова програмування – Python;

– отримання даних з веб-сайтів – requests;

– витягування потрібних даних з веб-сторінок – BeautifulSoup;

– обробка, аналіз та маніпуляція табличними даними – pandas;

– створення веб-сервера та API – Flask;

– створення бекапів бази даних – subprocess;

– обробка, аналіз та маніпуляція табличними даними – pandas;

– автоматизація – schedule.

```
from flask import Flask, jsonify
from flask_cors import CORS
from ProcessData import delete_rows_by_branch, round_existing_prices, convert_currency
from parser import parser
from database import toBD, get_by_multiple_param
from scheduler import start_scheduler

app = Flask(__name__)
CORS(app)

@app.route('/get_data')
def get_data():
    # Виконуємо очищення рядків за філіями
    delete_rows_by_branch()
    # Конвертуємо валюту цін
    convert_currency()
    # Округлюємо існуючі ціни
```

```
    round_existing_prices()
    # Отримуємо дані за множинними параметрами
    df = get_by_multiple_param()
    # Повертаємо результат у форматі JSON
    return jsonify(df.to_dict(orient='records'))

@app.route('/run_parser', methods=['POST'])
def run_parser():
    # Запускаємо парсер і отримуємо результат
    parser_result = parser()
    # Додаємо результати парсингу до бази даних
    toBD(parser_result)
    # Виконуємо очищення рядків за філіями
    delete_rows_by_branch()
    # Конвертуємо валюту цін
    convert_currency()
    # Округлюємо існуючі ціни
    round_existing_prices()
    # Повертаємо успішний результат
    return jsonify({'success': True}), 200

if __name__ == '__main__':
    # Виконуємо очищення рядків за філіями перед запуском серверу
    delete_rows_by_branch()
    # Конвертуємо валюту цін перед запуском серверу
    convert_currency()
    # Округлюємо існуючі ціни перед запуском серверу
    round_existing_prices()
```

```
# Запуск планировщика в отдельном потоке
start_scheduler()

# Запускаємо Flask додаток на порту 5000
app.run(port=5000, debug=True)
```