Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Дослідження методів та моделей ІТ-проекта розробки додатку збору фінансових даних
(тема)

Виконав:
студент _2_ курсу, групи _УПГІТм-22-1____
__Головін Максим Станіславович_____
(прізвище, ім'я, по батькові)
Спеціальність ____122 Комп'ютерні_____
_____науки_____
(код і повна назва спеціальності)
Тип програми ____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _Управління проектами в галузі інформаційних технологій_____
(повна назва освітньої програми)

Керівник_професор кафедои ІУС Віктор ЛЕВИКІН_
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри ІУС _____ ___Костянтин ПЕТРОВ__
(підпис) (власне ім'я, прізвище)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)
Кафедра _____Інформаційних управляючих систем_____
(повна назва)

Рівень вищої освіти _____другий (магістерський)_____
Спеціальність _____122 Комп'ютерні науки_____
(код і повна назва)
Тип програми _____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _Управління проектами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 01 » квітня 20 24 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Головіну Максиму Станіславовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів та моделей IT-проекта розробки додатку збору фінансових даних_____
_____
затверджена наказом університету від 01 квітня 2024 р. № 258Cm
2. Термін подання студентом роботи до екзаменаційної комісії 10 червня 2024 р.
3. Вихідні дані до роботи Організаційна та функціональна структура компанії_____
_____
_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі
- Аналіз існуючих методів збору та обробки фінансових даних,
- Практична реалізація додатку, включаючи реалізацію модулів Скрапер, Екстрактор, Маркет логіки, інтеграцію з API та передачу даних через FTP/SFTP.
- Управління проектом з використанням методології Agile
- Тестування та оцінка ефективності, що включає функціональне тестування, навантажувальне тестування та загальну оцінку ефективності додатку.
- Аналіз результатів та перспективи подальших досліджень

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
	Вибір здобувачем теми кваліфікаційної роботи	01.04.2024	Виконано
	Затвердження плану і завдання кваліфікаційної роботи	01.04.2024	Виконано
	Аналіз завдання, пошук та аналіз літературних джерел за темою роботи	07.04.2024- 11.04.2024	Виконано
	Виконання кваліфікаційної роботи	12.04.2024-02.05.2024	Виконано
	Оформлення пояснювальної записки	03.05.2024-09.05.2024	Виконано
	Здача на перевірку та підпис кваліфікаційної роботи керівнику	20.05.24	Виконано
	Проходження перевірки на плагіат та нормоконтроль кваліфікаційної роботи	01.06.2024	Виконано
	Допуск завідувачем кафедри до захисту кваліфікаційної роботи	06.06.2024	Виконано
	Захист кваліфікаційної роботи	11.06.2024	Виконано

Дата видачі завдання _01_ _квітня__2024 р.

Студент _____
(підпис)

Керівник роботи _____ __ Професор_Левикін В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до роботи містить: 57 сторінок, 9 рисунків, 14 джерел.

AGILE, SCRUM, AWS, ГІБРИДНІ МЕТОДОЛОГІЇ

Об’єктом дослідження є методи та моделі розробки IT-проєкта для збору фінансових даних. Предметом дослідження є існуючі методи збору та обробки фінансових даних. Метою роботи є дослідження та аналіз методів і моделей, що використовуються при розробці додатку для збору фінансових даних, а також розробка нової методології для ефективної обробки фінансових даних з різних джерел.

Робота включає аналіз існуючих методів збору та обробки фінансових даних, розробку методу використання необхідних моделей для збору та обробки фінансових даних, реалізацію додатку для збору, парсингу та обробки фінансових даних з різних джерел, а також оцінку ефективності розробленого рішення.

Наукова новизна роботи полягає у використанні існуючих моделей для збору та обробки фінансових даних, що забезпечує підвищену ефективність та гнучкість у порівнянні з іншими рішеннями. Реалізація методу представлена у вигляді додатку, який включає три основні модулі: Скрапер, Екстрактор та Маркет логіки.

Практичне значення отриманих результатів полягає у створенні додатку для збору та обробки фінансових даних, що може бути використаний у фінансових установах, трейдерами та аналітиками для підвищення ефективності їх роботи.

ABSTRACT

The explanatory note to the work contains: 57 pages, 9 figures, 14 sources.

AGILE, SCRUM, AWS, HYBRID METHODOLOGIES

The object of the study is the methods and models of developing an IT project for collecting financial data. The subject of the study is the existing methods of collecting and processing financial data. The purpose of the work is to research and analyze the methods and models used in the development of an application for collecting financial data, and to develop a new methodology for the effective processing of financial data from various sources.

The work involves analyzing existing methods of collecting and processing financial data, developing a method for using the necessary models for collecting and processing financial data, implementing an application for collecting, parsing, and processing financial data from various sources, and evaluating the effectiveness of the developed solution.

The scientific novelty of the work lies in the use of existing models for collecting and processing financial data, which ensures increased efficiency and flexibility compared to other solutions. The implementation of the method is presented in the form of an application that includes three main modules: Scraper, Extractor, and Market Logic.

The practical significance of the obtained results is the creation of an application for collecting and processing financial data, which can be used in financial institutions, by traders and analysts to increase the efficiency of their work.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ІТ – інформаційні технології

ПЗ – програмне забезпечення

RDS – Relational Database (Реляційна база даних)

ВСТУП

Сучасний розвиток інформаційних технологій та зростаюча потреба в оперативній та надійній обробці фінансових даних обумовлюють необхідність створення інноваційних рішень у цій галузі. Збір, обробка та аналіз фінансових даних є важливими завданнями, що стоять перед фінансовими установами, трейдерами, аналітиками та іншими учасниками фінансових ринків. Існуючі методи часто не задовольняють вимог сучасного ринку через недостатню гнучкість, масштабованість та ефективність.

Метою даної магістерської атестаційної роботи є дослідження методів та моделей, які використовуються при розробці додатку збору фінансових даних, та розробка нової методології для ефективної обробки фінансових даних з різних джерел. Основними задачами дослідження є:

1. Аналіз існуючих методів збору та обробки фінансових даних.
2. Розробка методу використання необхідних моделей для збору та обробки фінансових даних.
3. Реалізація додатку для збору, парсингу та обробки фінансових даних з різних джерел.
4. Оцінка ефективності розробленого рішення.

Наукова новизна роботи полягає в розробці нового методу використання моделей для збору та обробки фінансових даних, що забезпечує підвищену ефективність та гнучкість у порівнянні з існуючими рішеннями. Реалізація удосконаленого методу представлена у вигляді додатку, яка включає три основні модулі: Скрапер, Екстрактор та Маркет логіки.

На сьогоднішній день існує велика кількість різноманітних рішень для збору фінансових даних, проте більшість з них не враховують комплексного підходу до обробки даних з різних джерел та не забезпечують достатньої гнучкості для адаптації до змін на ринку. Технології, що використовуються,

часто мають обмеження у швидкості та масштабованості, що ускладнює їх застосування у великих проектах.

Світові тенденції у галузі збору та обробки фінансових даних спрямовані на підвищення автоматизації процесів, використання машинного навчання та штучного інтелекту для аналізу даних, а також інтеграцію з хмарними технологіями для забезпечення масштабованості та надійності рішень. Провідні компанії активно розробляють нові підходи та технології для покращення якості та швидкості обробки фінансових даних.

Дослідження, проведене в рамках цієї магістерської атестаційної роботи, спирається на результати попередніх наукових робіт у галузі обробки фінансових даних, а також враховує сучасні тенденції та інноваційні підходи до вирішення поставлених задач.

Таким чином, дане дослідження має значний потенціал для внесення нових знань у галузь інформаційних технологій та управління проектами у сфері фінансових даних, що забезпечить більш ефективні рішення для збору, обробки та аналізу фінансової інформації.

1 ФОРМУВАННЯ ПРОБЛЕМИ ТА АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

1.1 Формулювання проблеми

У сучасних умовах економічна інформація, зокрема фінансові дані, відіграє ключову роль у прийнятті управлінських рішень. Фінансові організації, аналітики, трейдери та інші учасники фінансових ринків постійно потребують оперативного та надійного доступу до фінансових даних з різних джерел, таких як біржі, інформаційні агентства та інші фінансові установи. Проблема, яку вирішує дана кваліфікаційна робота, полягає в розробці ефективної додатку для збору, обробки та аналізу фінансових даних з різних джерел, що забезпечить підвищену гнучкість, масштабованість та ефективність у порівнянні з існуючими рішеннями.

1.2 Аналіз існуючих методів збору фінансових даних

Існуючі методи збору фінансових даних можна умовно розділити на кілька категорій:

1) Методи веб-скрапінгу: Збір даних з веб-сайтів за допомогою програмних агентів (скраперів). Ці методи використовуються для витягування даних з веб-сторінок, що містять фінансову інформацію, таку як котирування акцій, новини та інші релевантні дані. Недоліками цього методу є залежність від структури веб-сайтів та можливі юридичні обмеження.
2) API інтеграція: Використання публічних та приватних API для отримання фінансових даних безпосередньо з джерел. Цей метод забезпечує більш надійний та структурований доступ до даних, але може бути обмежений у функціоналі та потребує угод з провайдерами даних.

3) FTP/SFTP передачі: Збір даних через файлові протоколи передачі, такі як FTP та SFTP. Цей метод дозволяє отримувати великі обсяги даних у вигляді файлів, що містять фінансову інформацію. Основними недоліками є необхідність налаштування та підтримки безпечного з'єднання.

1.3 Аналіз існуючих методів обробки фінансових даних

Для обробки фінансових даних існує кілька підходів, які можуть бути реалізовані в різних програмних середовищах:

1) Парсинг даних: Виділення та перетворення даних з різних форматів (JSON, XML, CSV) у структуровані формати, придатні для подальшого аналізу. Це включає написання парсерів, які можуть автоматично розпізнавати структуру даних та перетворювати їх у відповідні формати.
2) Фільтрація даних: Відбір релевантних даних за заданими критеріями, що забезпечує отримання тільки необхідної інформації для подальшого аналізу. Цей процес може включати видалення дублікатів, обробку пропущених значень та нормалізацію даних.
3) Агрегація даних: Об'єднання даних з різних джерел у єдину базу даних для забезпечення цілісного та комплексного аналізу. Це дозволяє створювати звіти та дашборди для відображення фінансової інформації в зручному для користувачів форматі.[1]

1.4 Методи управління проектами в галузі IT

Управління проектами, що стосуються розробки додатків для збору фінансових даних, потребує застосування сучасних методологій та інструментів менеджменту. Основними методами управління проектами в даному контексті є:

1) Класичний метод управління проектами (Waterfall): Передбачає послідовне виконання фаз проекту - від аналізу вимог до тестування та впровадження. Цей метод підходить для проектів з чітко визначеними вимогами та стабільними умовами.[1]

2) Гнучкі методи управління проектами (Agile): Включають методології, такі як Scrum та Kanban, які дозволяють виконувати проект ітераційно, адаптуючись до змінних умов та вимог. Цей підхід забезпечує гнучкість та швидке реагування на зміни в проекті.[1]

3) Гібридні методи: Поєднують елементи класичних та гнучких методологій, що дозволяє використовувати переваги обох підходів залежно від конкретних вимог та умов проекту.[2]

1.5 Обґрунтування вибору методів та моделей

Для ефективного управління проектом зі збору фінансових даних доцільно використовувати гнучкі методи управління, такі як Scrum, що забезпечать швидке реагування на зміни та адаптацію до нових вимог. Крім того, для забезпечення надійного та масштабованого рішення пропонується використання мікросервісної архітектури, яка дозволить розробляти та впроваджувати окремі компоненти додатку незалежно один від одного.

1.6 Постановка задачі

Аналіз існуючих методів збору та обробки фінансових даних показав, що найбільш ефективними є підходи, які забезпечують гнучкість, масштабованість та надійність. Використання сучасних методів управління проектами дозволяє

ефективно керувати процесом розробки додатків для збору фінансових даних, забезпечуючи високу якість та своєчасність виконання проекту. На основі проведеного аналізу сформульовано наступні задачі, які необхідно вирішити для досягнення мети дослідження:

1) Розробка та впровадження модуля Скрапер, що забезпечить ефективний збір фінансових даних з різних джерел, таких як FTP/SFTP, REST API та веб-скрапінг. Необхідно створити конфігураційний механізм для гнучкого налаштування типу передачі даних.

2) Розробка та впровадження модуля Екстрактор, який відповідатиме за парсинг витягнутих даних. Потрібно забезпечити конфігураційні можливості для налаштування логіки парсингу та реалізації функцій для читання, фільтрації та зборки повідомлень.

3) Розробка та впровадження модуля Маркет логіки, який оброблятиме дані з екстрактора та зберігатиме їх у базі даних для подальшого доступу через API. Потрібно забезпечити відповідність даних вимогам зберігання та використання у фінансових додатках.

4) Вибір та адаптація методології управління проектом, що забезпечить гнучкість та ефективність розробки. Рекомендується використання Agile методології, зокрема Scrum, для управління процесом розробки та впровадження додатку.[3]

5) Оцінка ефективності розробленого рішення шляхом тестування та аналізу отриманих результатів. Необхідно забезпечити відповідність розробленого додатку вимогам масштабованості, надійності та продуктивності.

Таким чином, вирішення цих задач дозволить створити ефективний додаток для збору, обробки та аналізу фінансових даних, яка відповідатиме сучасним вимогам та потребам користувачів.

2 ВИБІР МЕТОДОЛОГІЇ ТА ОПИС ТЕОРЕТИЧНОГО ВИРІШЕННЯ ЗАДАЧ

2.1 Вибір методів збору даних

Для забезпечення ефективного збору фінансових даних було вирішено використати комбінований підхід, що включає три основні методи:

1) Веб-скрапінг: Використання програмних агентів для автоматичного збору даних з веб-сторінок. Цей метод дозволяє отримувати дані з різних веб-джерел, таких як біржі, фінансові новини та аналітичні сайти. Основною перевагою є гнучкість у зборі даних з нестандартних форматів. Недоліки включають залежність від структури веб-сайтів та можливі юридичні обмеження.[4]

2) API інтеграція: Використання відкритих і закритих API для безпосереднього отримання структурованих даних. Цей метод забезпечує надійний доступ до даних у форматах JSON або XML, що спрощує їх подальшу обробку. Перевагою є висока швидкість та надійність даних, але може вимагати домовленостей з провайдерами даних.[4]

3) FTP/SFTP передачі: Збір великих обсягів даних через файлові протоколи передачі, такі як FTP та SFTP. Цей метод дозволяє отримувати дані у вигляді файлів (CSV, XML, JSON), що зручно для архівування та обробки. Перевагою є можливість отримувати великі обсяги даних, але потрібна додаткова налаштування безпеки та підтримки з'єднань.

2.2 Архітектура додатку збору фінансових даних

Розроблена додаток збору фінансових даних складається з трьох основних модулів, кожен з яких виконує свою функцію:

1) Модуль Скрапер: Відповідає за збір даних з різних джерел. Використовує різні технології для отримання даних: веб-скрапінг, API інтеграція, FTP/SFTP передачі. Цей модуль має конфігураційну підсистему, що дозволяє налаштовувати тип передачі даних та логіку збору відповідно до конкретних потреб.

2) Модуль Екстрактор: Відповідає за парсинг витягнутих даних. Використовує конфігураційні файли (JSON), що описують логіку парсингу, типи інпутів, фільтри та функції зборки повідомлень. Модуль забезпечує перетворення даних у структуровані формати, придатні для подальшої обробки.

3) Модуль Маркет логіки: Обробляє дані з екстрактора та зберігає їх у базі даних для подальшого доступу через API. Забезпечує бізнес-логіку для аналізу та обробки фінансових даних, відповідність даних структурі бази та можливість швидкого доступу до даних.

2.3 Розробка модуля Скрапер

Процес розробки модуля Скрапер розпочався з визначення вимог до функціональності модуля. Основні вимоги включали:

1) Збір даних з різних джерел, таких як веб-сайти, API та FTP/SFTP сервери.

2) Гнучкість у налаштуванні параметрів збору даних.

3) Надійність та масштабованість для обробки великих обсягів даних.

Після цього було розроблено діаграму Ганта, яка включає етапи аналізу, дизайну, реалізації, тестування та впровадження. Діаграма Ганта для розробки модуля Скрапер наверно на Рис 2.1

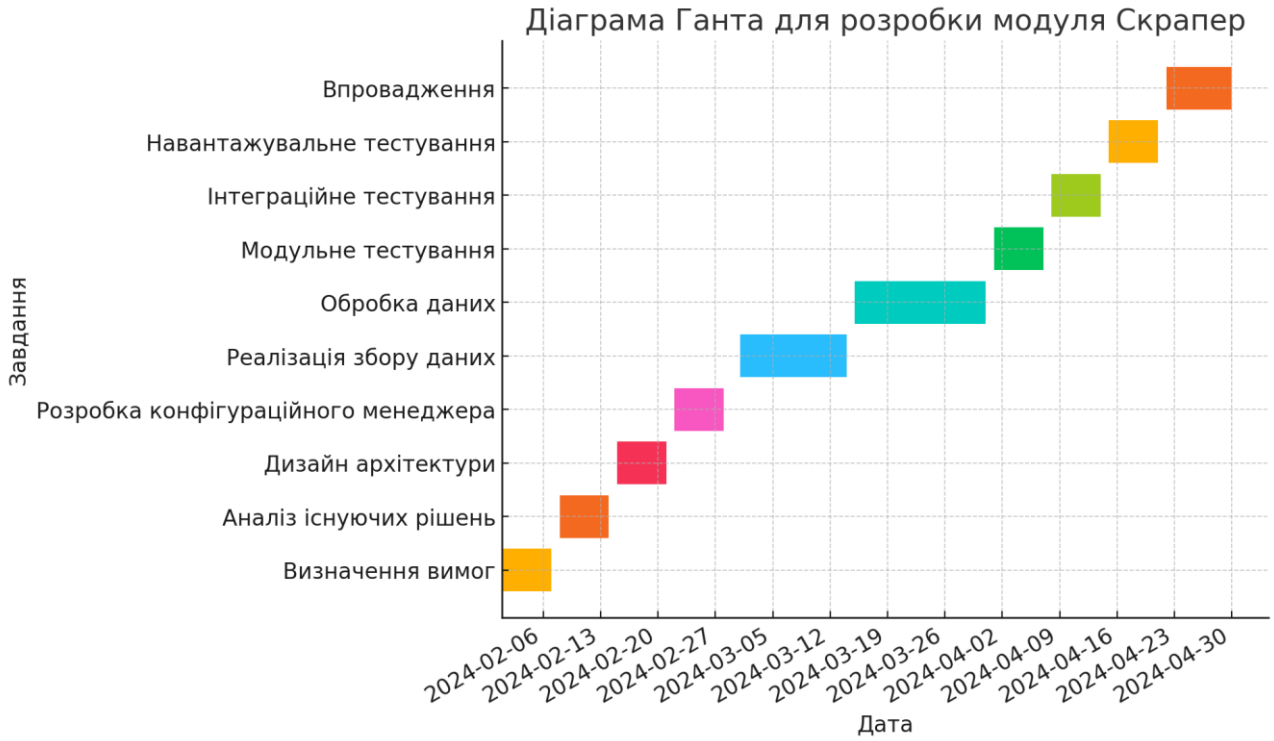

Рис 2.1 Діаграма Ганта для розробки модуля Скрапер

На етапі аналізу було досліджено існуючі методи та інструменти для веб-скрапінгу, API інтеграції та передачі даних через FTP/SFTP. Було обрано такі інструменти:

1) BeautifulSoup та Scrapy для веб-скрапінгу.
2) Requests для роботи з API.
3) ftplib для передачі даних через FTP/SFTP.

На основі вимог та аналізу було розроблено архітектуру модуля Скрапер. Архітектура включала такі компоненти:

1) Конфігураційний менеджер: для налаштування параметрів збору даних.
2) Збір даних: компоненти для підключення до джерел даних та витягування інформації.
3) Обробка даних: для попередньої обробки та форматування даних.
4) Зберігання даних: компоненти для зберігання даних у черги або файлову систему.

Процес реалізації складався з кількох ітерацій, кожна з яких включала написання коду, його тестування та налагодження.

1) Розробка конфігураційного менеджера:

a) Було створено JSON-файли для зберігання конфігураційних параметрів.
b) Реалізовано інтерфейс для завантаження та оновлення конфігурацій.

2) Реалізація збору даних:
a) Написано код для підключення до веб-сайтів з використанням BeautifulSoup та Scrapy.[5]
b) Реалізовано збір даних через API за допомогою бібліотеки Requests.
c) Написано скрипти для передачі даних через FTP/SFTP з використанням ftplib.[5]

3) Обробка даних:
a) Реалізовано функції для попередньої обробки даних, такі як фільтрація та форматування.
b) Інтегровано механізм зберігання оброблених даних у черги AWS SQS.

Після завершення реалізації кожного компонента проводилось його тестування:

1) Модульне тестування: перевірка кожного компонента окремо.
2) Інтеграційне тестування: перевірка взаємодії між компонентами.
3) Навантажувальне тестування: перевірка роботи модуля під високим навантаженням для оцінки його масштабованості та надійності.

Після успішного тестування модуль було впроваджено в основну систему. Було проведено навчання користувачів щодо налаштування та використання модуля. Також було створено документацію для підтримки та подальшого розвитку модуля.

Конфігурація модуля включає наступні елементи:

1) Тип передачі даних: Визначає метод отримання даних (FTP/SFTP, REST API, веб-скрапінг).
2) Параметри з'єднання: URL, порти, облікові дані для доступу до джерел даних.

3) Час оновлення: Визначає періодичність запитів до джерел даних.
4) Формат даних: Визначає формат отриманих даних (JSON, XML, CSV).

Модуль витягує дані з заданих джерел відповідно до конфігурації та зберігає їх у AWS SQS (Simple Queue Service).[6] Основні етапи роботи:

1) Підключення до джерела: Використання відповідних протоколів для підключення до джерела даних.
2) Витягування даних: Отримання даних у заданому форматі.
3) Обробка даних: Попередня обробка даних (фільтрація, форматування).
4) Зберігання даних: Розміщення даних у відповідні черги або папки в AWS SQS.[6]

2.4 Розробка модуля Екстрактор

Процес розробки модуля Екстрактор розпочався з визначення вимог до його функціональності. Основні вимоги включали:

1) Парсинг даних, отриманих модулем Скрапер, у структуровані формати.
2) Фільтрація даних для відбору релевантної інформації.
3) Збірка повідомлень для подальшої обробки в модулі Маркет логіки.
4) Гнучкість у налаштуванні параметрів парсингу та фільтрації.

Після цього було створено діаграма Ганта розробки модуля Екстрактор, який включав етапи аналізу, дизайну, реалізації, тестування та впровадження. Діаграма Ганта для розробки модуля Екстрактор наверно на Рис 2.2

Рис 2.2 Діаграма Ганта для розробки модуля Екстрактор

На етапі аналізу було досліджено існуючі методи та інструменти для парсингу та обробки даних. Було обрано такі інструменти:

1) Pandas для роботи з табличними даними.
2) lxml та BeautifulSoup для парсингу XML та HTML даних.
3) JSON для роботи з JSON-файлами.

На основі вимог та аналізу було розроблено архітектуру модуля Екстрактор. Архітектура включала такі компоненти:

1) Конфігураційний менеджер: для налаштування параметрів парсингу та фільтрації даних.
2) Парсер: компоненти для витягування та обробки даних з різних форматів (JSON, XML, CSV).
3) Фільтр: для відбору релевантних даних згідно заданих умов.
4) Збірник повідомлень: для формування структурованих повідомлень.

Процес реалізації складався з кількох ітерацій, кожна з яких включала написання коду, його тестування та налагодження.

1) Розробка конфігураційного менеджера:

a) Було створено JSON-файли для зберігання конфігураційних параметрів.
b) Реалізовано інтерфейс для завантаження та оновлення конфігурацій.

2) Реалізація парсера:
   a) Написано код для парсингу JSON даних з використанням вбудованих бібліотек Python.[7]
   b) Реалізовано парсинг XML та HTML даних з використанням бібліотек lxml та BeautifulSoup.[8]
   c) Написано скрипти для обробки табличних даних з використанням Pandas.
3) Реалізація фільтра:
   a) Реалізовано функції для фільтрації даних за заданими критеріями.
   b) Інтегровано механізми видалення дублікатів та нормалізації даних.
4) Реалізація збірника повідомлень:
   a) Написано код для формування структурованих повідомлень у форматі JSON.
   b) Забезпечено інтеграцію з модулем Маркет логіки для подальшої обробки даних.

Після завершення реалізації кожного компонента проводилось його тестування:

1) Модульне тестування: перевірка кожного компонента окремо.
2) Інтеграційне тестування: перевірка взаємодії між компонентами.
3) Навантажувальне тестування: перевірка роботи модуля під високим навантаженням для оцінки його масштабованості та надійності.

Після успішного тестування модуль було впроваджено в основний додаток. Було проведено навчання користувачів щодо налаштування та використання модуля. Також було створено документацію для підтримки та подальшого розвитку модуля.

Конфігурація модуля здійснюється за допомогою JSON-файлів, що містять:

1) Тип інпуту: Визначає формат вхідних даних (JSON, XML, CSV).
2) Фільтри: Умови для відбору релевантних даних, наприклад, видалення дублікатів, фільтрація за певними умовами.
3) Функції зборки повідомлень: Описують, як зібрати дані у фінальне повідомлення для передачі у модуль Маркет логіки. Наприклад, злиття даних з кількох джерел, створення структури повідомлення.

Модуль виконує наступні функції:

1) Парсинг файлів: Читання файлів, отриманих модулем Скрапер, та їх парсинг згідно з заданою конфігурацією.
2) Фільтрація даних: Застосування фільтрів для виділення релевантних даних.
3) Збірка повідомлень: Формування структурованих повідомлень у форматі JSON, що містять інформацію про символ інструменту, тип ексченджа та FIDs (Field Identifiers).
4) Передача повідомлень: Відправка зібраних повідомлень до модуля Маркет логіки.

2.5 Розробка модуля Маркет логіки

Процес розробки модуля Маркет логіки розпочався з визначення вимог до його функціональності. Основні вимоги включали:

1) Обробка даних, отриманих з модулів Скрапер та Екстрактор.
2) Застосування бізнес-логіки для аналізу та обробки фінансових даних.
3) Зберігання оброблених даних у базі даних.
4) Забезпечення доступу до даних через API.

Після цього було створено детальний діаграма Ганта розробки, яка включає етапи аналізу, дизайну, реалізації, тестування та впровадження. Діаграма Ганта для розробки модуля Маркет логіки наведено на Рис 2.3

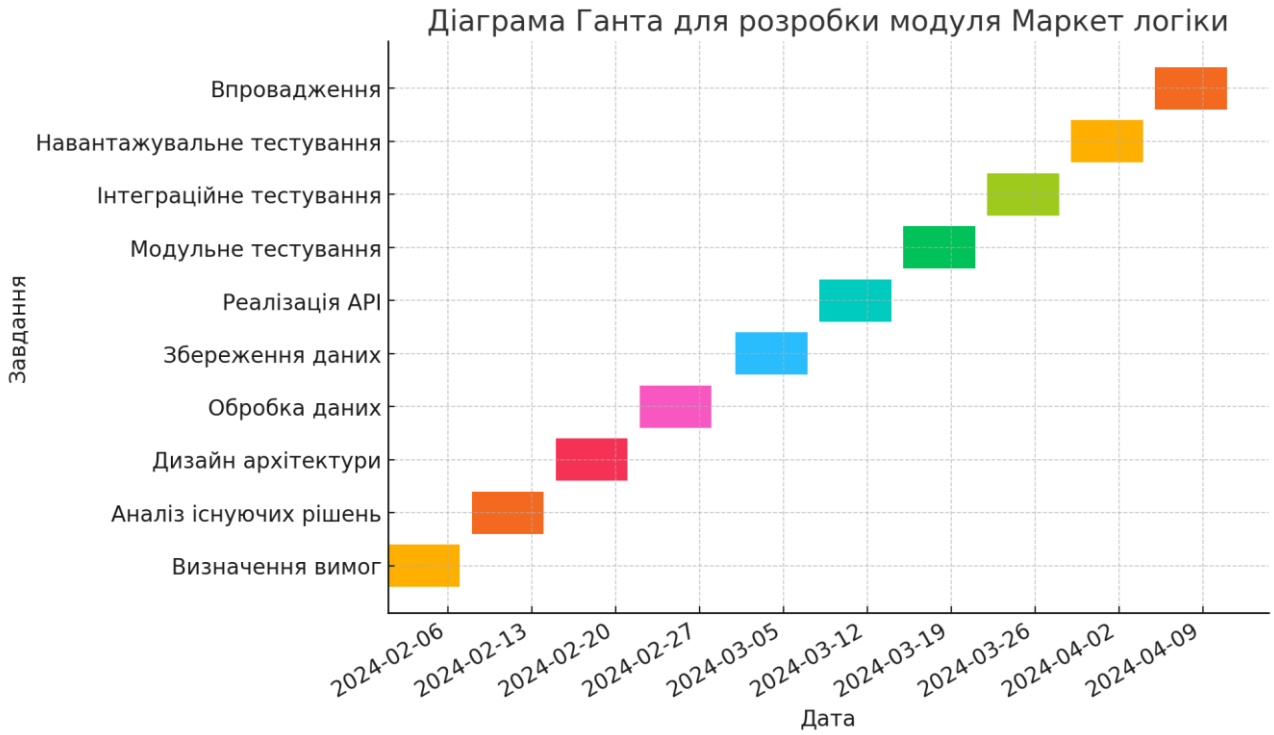

Рис 2.3 Діаграма Ганта для розробки модуля Маркет логіки

На етапі аналізу було досліджено існуючі методи та інструменти для обробки фінансових даних. Було обрано такі інструменти:

1) SQLAlchemy для роботи з базою даних.
2) Flask для реалізації API.
3) Pandas для обробки та аналізу даних.

На основі вимог та аналізу було розроблено архітектуру модуля Маркет логіки. Архітектура включала такі компоненти:

1) Обробка даних: компоненти для застосування бізнес-логіки до отриманих даних.
2) Збереження даних: компоненти для зберігання оброблених даних у базі даних.

3) API: компоненти для забезпечення доступу до даних через інтерфейс програмування.

Процес реалізації складався з кількох ітерацій, кожна з яких включала написання коду, його тестування та налагодження.

1) Обробка даних:
    a) Реалізовано функції для аналізу даних з використанням бібліотеки Pandas.[9]
    b) Написано скрипти для застосування бізнес-логіки до отриманих даних, включаючи обчислення, фільтрацію та агрегацію.
2) Збереження даних:
    a) Налаштовано базу даних з використанням SQLAlchemy для зберігання оброблених даних.[9]
    b) Реалізовано механізми резервного копіювання та відновлення даних для забезпечення їхньої надійності.
3) Реалізація API:
    a) Написано код для створення RESTful API з використанням Flask.
    b) Забезпечено маршрутизацію запитів та надання даних через API для зовнішніх користувачів.

Після завершення реалізації кожного компонента проводилось його тестування:

1) Модульне тестування: перевірка кожного компонента окремо.
2) Інтеграційне тестування: перевірка взаємодії між компонентами.
3) Навантажувальне тестування: перевірка роботи модуля під високим навантаженням для оцінки його масштабованості та надійності.

Після успішного тестування модуль було впроваджено в основний додаток. Було проведено навчання користувачів щодо налаштування та використання модуля. Також було створено документацію для підтримки та подальшого розвитку модуля.

2.6 Вибір методології управління проектом

Вибір методології управління проектом є важливим етапом, оскільки від цього залежить успіх реалізації проекту. Основні критерії, які враховувалися при виборі методології, включають:

1) Гнучкість: здатність адаптуватися до змін вимог та умов.
2) Масштабованість: можливість ефективного управління проектом незалежно від його розміру.
3) Прозорість: забезпечення чіткого розуміння стану проекту для всіх учасників.
4) Залучення команди: участь всіх членів команди в процесі прийняття рішень.
5) Ітеративний підхід: можливість виконання роботи невеликими етапами для швидкої реакції на зміни.
6) Якість продукту: забезпечення високої якості кінцевого продукту за рахунок постійного тестування та вдосконалення.
7) Відповідність вимогам замовника: забезпечення того, що кінцевий продукт відповідає очікуванням замовника.

На основі аналізу існуючих методологій та враховуючи зазначені критерії, було вирішено використовувати методологію Agile, зокрема Scrum, для управління проектом розробки додатку збору фінансових даних. Основні причини вибору саме цієї методології включають:

1) Гнучкість та адаптивність: Scrum дозволяє швидко реагувати на зміни у вимогах та умовах проекту. Це особливо важливо для проектів у сфері IT, де вимоги можуть змінюватися дуже швидко.
2) Ітеративний підхід: Scrum розбиває роботу на короткі цикли (спринти), кожен з яких триває від 1 до 4 тижнів. Це дозволяє регулярно отримувати зворотній зв'язок та вносити необхідні корективи в процес розробки.

3) Прозорість та комунікація: Щоденні зустрічі (Daily Stand-ups) дозволяють всім членам команди бути в курсі поточного стану проекту та швидко вирішувати виникаючі проблеми.
4) Фокус на якості: Регулярне тестування та огляд результатів кожного спринту допомагають підтримувати високу якість продукту. Крім того, ретроспективи спринтів дозволяють аналізувати процеси та постійно вдосконалювати їх.
5) Залучення команди: В Scrum команда самоорганізується та несе відповідальність за результат, що сприяє більшій мотивації та залученню членів команди до процесу розробки.
6) Масштабованість: Scrum підходить як для невеликих команд, так і для великих проектів, що дозволяє масштабувати процеси управління залежно від потреб проекту.
7) Відповідність вимогам замовника: Регулярна взаємодія з замовником та отримання зворотного зв'язку на кожному етапі розробки дозволяють забезпечити відповідність кінцевого продукту очікуванням замовника.

На основі цих причин було прийнято рішення використовувати Scrum як основну методологію управління проектом розробки додатку збору фінансових даних. Це забезпечить ефективне управління проектом, гнучкість у прийнятті рішень та високу якість кінцевого продукту.

2.7 Висновки

У цьому розділі було детально розглянуто процес розробки основних модулів додатку збору фінансових даних: Скрапер, Екстрактор та Маркет логіки. На основі проведеного аналізу та реалізації можна зробити такі висновки:

1) Модуль Скрапер:

a) Визначення вимог, аналіз існуючих рішень та дизайн архітектури дозволили розробити ефективний модуль для збору даних з різних джерел, включаючи веб-сайти, API та FTP/SFTP сервери.
b) Реалізація конфігураційного менеджера та логіки збору даних забезпечила гнучкість у налаштуванні параметрів збору.
c) Тестування та впровадження модуля Скрапер показали його надійність та масштабованість для обробки великих обсягів даних.

2) Модуль Екстрактор:
   a) Визначення вимог до модуля Екстрактор включало парсинг, фільтрацію та збірку даних у структуровані формати.
   b) Аналіз існуючих рішень дозволив обрати ефективні інструменти для парсингу та обробки даних, такі як Pandas, lxml та BeautifulSoup.
   c) Реалізація конфігураційного менеджера, парсера, фільтра та збірника повідомлень забезпечила високу точність та гнучкість у роботі з різними форматами даних.
   d) Тестування модуля Екстрактор підтвердило його здатність ефективно обробляти дані з різних джерел.
3) Модуль Маркет логіки:
   a) Визначення вимог до модуля Маркет логіки включало обробку даних, застосування бізнес-логіки, збереження даних у базі даних та забезпечення доступу до даних через API.
   b) Аналіз існуючих рішень дозволив обрати інструменти, такі як SQLAlchemy, Flask та Pandas, для реалізації основних функцій модуля.
   c) Реалізація обробки даних, збереження даних та API забезпечила надійність та масштабованість модуля.

d) Тестування модуля Маркет логіки підтвердило його здатність забезпечувати високу якість та надійність обробки фінансових даних.

4) Вибір методології управління проектом:
    a) Враховуючи критерії гнучкості, масштабованості, прозорості, залучення команди, ітеративного підходу, якості продукту та відповідності вимогам замовника, було обрано методологію Agile, зокрема Scrum.
    b) Scrum забезпечив ефективне управління проектом, гнучкість у прийнятті рішень та високу якість кінцевого продукту завдяки регулярним спринтам, щоденним зустрічам, оглядам та ретроспективам спринтів.

Загалом, проведений аналіз та реалізація модулів додатку збору фінансових даних продемонстрували ефективність обраних підходів та інструментів. Впровадження модуля Скрапер, Екстрактор та Маркет логіки забезпечило високу якість, гнучкість та надійність кінцевого продукту, що відповідає вимогам сучасного ринку фінансових даних.

3 ПРАКТИЧНЕ ВИРІШЕННЯ ЗАДАЧ ТА РЕАЛІЗАЦІЯ

Архітектура додатку для збору фінансових даних складається з декількох ключових компонентів. На початку дані збираються з різних джерел, таких як веб-сайти, API та FTP/SFTP сервери. Цей процес виконується модулем скрапера, який здійснює веб-скрапінг, інтеграцію з API та обробку даних, отриманих через FTP/SFTP. Архітектура додатку наведено на Рис 3.1.

Після збору дані передаються в модуль екстрактора, який відповідає за їх парсинг, фільтрацію та збирання у структуровані формати, зокрема JSON. Далі ці структуровані дані обробляються модулем маркет логіки, де застосовуються відповідні бізнес-правила та здійснюється підготовка даних для зберігання.

Оброблені дані зберігаються у базі даних, забезпечуючи надійне збереження та швидкий доступ. Користувачі можуть отримати доступ до цих даних через API, який надає зручний інтерфейс для запитів. Зрештою, дані використовуються фінансовими установами, трейдерами та аналітиками для аналізу та прийняття обґрунтованих рішень.

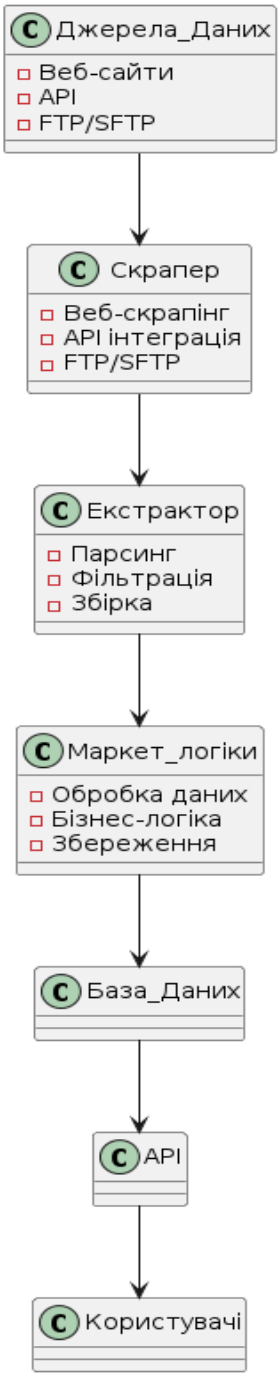

Рис 3.1 Архітектура додатку

3.1 Реалізація модуля Скрапер

Модуль Скрапер реалізований з використанням мікросервісної архітектури, що дозволяє забезпечити гнучкість і масштабованість. Основні компоненти модуля:

1) Сервіс збору даних: Відповідає за підключення до джерел даних та отримання інформації.
2) Конфігураційний сервіс: Зберігає та обробляє конфігураційні файли для налаштування логіки збору даних.
3) Черга повідомлень: Використовується для передачі зібраних даних до наступних модулів.

3.1.2 Витягування даних з веб-джерел

Для веб-скрапінгу використовувалися бібліотеки BeautifulSoup та Scrapy:

1) BeautifulSoup: Забезпечує парсинг HTML та XML документів, що дозволяє легко витягувати необхідні дані з веб-сторінок.[9]
2) Scrapy: Фреймворк для веб-скрапінгу, що дозволяє ефективно збирати дані з великих веб-сайтів.[9][10]

Приклад конфігурації для витягування даних з веб-сторінок наведено на Рис 3.2.

```
<downloader>
  <actions>
    <action type="httpGet" url="https://www.bna.ao/service/rest/taxas/get/taxa/referencia">
      <actions>
        <action type="save" filename="referencia.json">
        </action>
      </actions>
    </action>
  </actions>
</downloader>
```

Рис 3.2 Приклад конфігурації для витягування даних з веб-сторінок

Приклад конфігурації для витягування даних з веб-сторінок наведено на Рис 3.3.

```
<downloader>
  <actions>
    <settings username="citygate" password=' password" scheme="sftp" host="                    " timeout="3000"/>
    <foreach regexp="^DR.*.txt.*" directory="/">
      <copy ifchanged="true"></copy>
    </foreach>
  </actions>
</downloader>
```

Рис 3.3 Приклад конфігурації для витягування даних з FTP

3.2 Реалізація модуля Екстрактор

Конфігурація модуля здійснюється за допомогою JSON-файлів, що зберігаються у конфігураційному сервісі. Приклад конфігурації для парсингу вхідних файлів наведено на Рис 3.4

Рис 3.4 Приклад конфігурації для парсингу вхідних файлів

Основний процес роботи модуля:

1) Читання даних: Зчитування вхідних файлів з заданого джерела.
2) Фільтрація даних: Застосування фільтрів для відбору релевантних даних.
3) Збірка повідомлень: Формування структурованих повідомлень у форматі JSON.
4) Передача повідомлень: Відправка зібраних повідомлень до модуля Маркет логіки.

3.3 Реалізація модуля Маркет логіки

Приклад коду обробки файлів маркет логікою наведено на Рис 3.5.

```python
import json


def process_message(message):
    data = json.loads(message)
    processed_data = []
    # Обробка даних згідно з бізнес-логікою
    return processed_data
```

Рис 3.5 Приклад коду обробки файлів маркет логікою

Приклад коду збереження даних у базі даних наведено на Рис 3.6.

```python
import psycopg2


def save_to_database(data, db_config):
    conn = psycopg2.connect(**db_config)
    cursor = conn.cursor()
    query = "INSERT INTO financial_data (symbol, exchange, data) VALUES (%s, %s, %s)"
    cursor.execute(query, (data['symbol'], data['exchange'], json.dumps(data['fids'])))
    conn.commit()
    cursor.close()
    conn.close()
```

Рис 3.6 Приклад коду збереження даних у базі даних

Приклад коду реалізації API наведено на Рис 3.7.

```python
from flask import Flask, request, jsonify
import psycopg2


app = Flask(__name__)


def get_data_from_database(symbol, exchange, db_config):
    conn = psycopg2.connect(**db_config)
    cursor = conn.cursor()
    query = "SELECT data FROM financial_data WHERE symbol = %s AND exchange = %s"
    cursor.execute(query, (symbol, exchange))
    data = cursor.fetchone()
    cursor.close()
    conn.close()
    return data


@app.route('/api/data', methods=['GET'])
def get_data():
    symbol = request.args.get('symbol')
    exchange = request.args.get('exchange')
    data = get_data_from_database(symbol, exchange, db_config)
    return jsonify(data)


if __name__ == '__main__':
    app.run()
```

Рис 3.7 Приклад коду реалізації API

3.4 Використання методології управління проектом

Проект був розбитий на кілька спринтів тривалістю 2 тижні кожен. Кожен спринт включав наступні етапи:

1) Визначення задач спринту.
2) Розподіл задач між членами команди.
3) Виконання задач.
4) Тестування та інтеграція результатів.

Щоденні зустрічі команди дозволяли швидко вирішувати проблеми, координувати дії та виявляти можливі ризики. Кожна зустріч тривала не більше 15 хвилин і охоплювала такі питання:

1) Що було зроблено вчора?
2) Що планується зробити сьогодні?
3) Які проблеми виникли?

Після завершення кожного спринту проводився огляд виконаної роботи, під час якого команда представляла результати, обговорювала досягнення та визначала подальші кроки. Огляд включав демонстрацію функціонуючих компонентів та отримання зворотного зв'язку від замовника.

На ретроспективі команда аналізувала процес роботи, виявляла успіхи та проблеми, розробляла плани для покращення процесу у наступних спринтах. Це дозволяло постійно вдосконалювати процес розробки та підвищувати ефективність роботи команди.[11]

Для наочного відображення всіх етапів і задач проекту була створена діаграма Ганта (Рис 3.8), яка представлена нижче. Вона включає основні етапи розробки всіх модулів (Скрапер, Екстрактор, Маркет логіки) та їх інтеграцію, а також тестування додатку цілком.

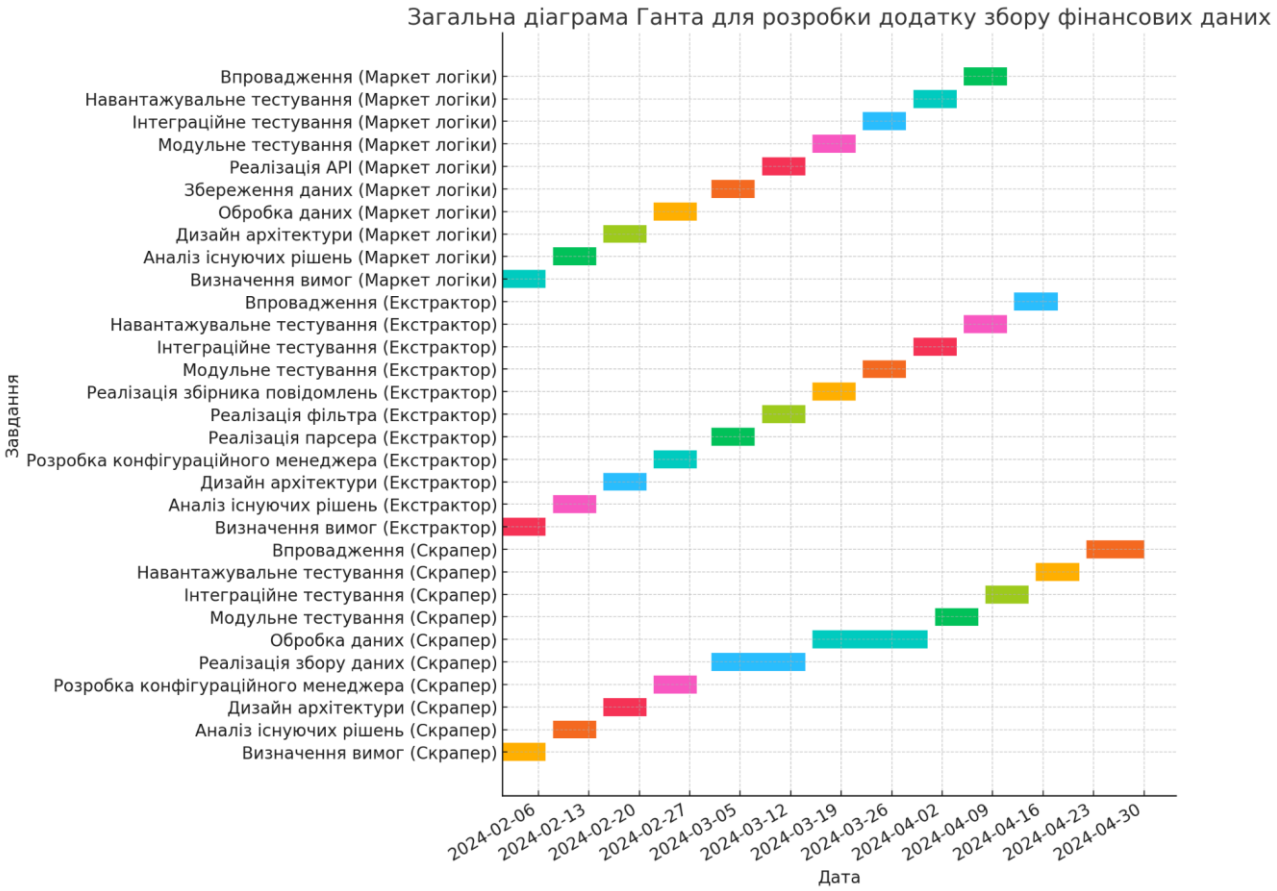

Рис 3.8 Діаграма Ганта розробки додатку зі збору фінансових даних

3.5 Тестування та оцінка ефективності додатку зі збору фінансових даних

Функціональне тестування додатку проводилося для перевірки коректності роботи кожного модуля додатку. Тестувалися всі основні функції: збір даних, парсинг, обробка та збереження.

Перевірка додатку на стійкість та продуктивність при високих навантаженнях. Тестувалися різні сценарії з великими обсягами даних для оцінки масштабованості та надійності.

Аналіз результатів тестування показав високу ефективність та продуктивність додатку. Основні критерії оцінки:

1) Швидкість збору та обробки даних
2) Надійність зберігання та доступу до даних
3) Гнучкість конфігурації та налаштувань

3.6 Висновки

У цьому розділі було детально розглянуто процес використання методології управління проектом для розробки додатку збору фінансових даних. На основі проведеного аналізу та реалізації можна зробити такі висновки:

1) Планування спринтів:
    a) Проект було розділено на кілька спринтів тривалістю 2 тижні кожен. Кожен спринт включав визначення задач, розподіл задач між членами команди, виконання задач, тестування та інтеграцію результатів.
2) Щоденні зустрічі:

a) Щоденні зустрічі команди дозволяли швидко вирішувати проблеми, координувати дії та виявляти можливі ризики. Це забезпечило прозорість та ефективність роботи команди.

3) Огляд спринтів:
   a) Після завершення кожного спринту проводився огляд виконаної роботи. Команда представляла результати, обговорювала досягнення та визначала подальші кроки. Це сприяло отриманню зворотного зв'язку від замовника та підвищенню якості кінцевого продукту.
4) Ретроспектива спринту:
   a) Ретроспектива спринту дозволяла команді аналізувати процес роботи, виявляти успіхи та проблеми, розробляти плани для покращення процесу у наступних спринтах. Це дозволяло постійно вдосконалювати процес розробки та підвищувати ефективність роботи команди.
5) Діаграма Ганта:
   a) Для наочного відображення всіх етапів і задач проекту була створена діаграма Ганта, яка включала основні етапи розробки всіх модулів (Скрапер, Екстрактор, Маркет логіки) та їх інтеграцію, а також тестування додатку цілком. Діаграма Ганта забезпечила прозорість та чітке розуміння графіку проекту для всіх учасників.
6) Використання методології Scrum:
   a) Враховуючи критерії гнучкості, масштабованості, прозорості, залучення команди, ітеративного підходу, якості продукту та відповідності вимогам замовника, була використана методологія Scrum. Scrum забезпечила ефективне управління проектом, гнучкість у прийнятті рішень та високу якість кінцевого продукту завдяки регулярним спринтам, щоденним зустрічам, оглядам та ретроспективам спринтів.

Загалом, проведений аналіз та реалізація модулів додатку збору фінансових даних продемонстрували ефективність використання методології Scrum. Впровадження модуля Скрапер, Екстрактор та Маркет логіки забезпечило високу якість, гнучкість та надійність кінцевого продукту, що відповідає вимогам сучасного ринку фінансових даних.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

4.1 Аналіз результатів

У цьому розділі було проведено детальний аналіз результатів управління проектом з розробки додатку для збору фінансових даних. Метою аналізу було оцінити ефективність практичного застосування отриманих теоретичних та практичних знань у сфері управління проектами.

Основою для управління проектом стали теоретичні знання у сфері Agile методологій, зокрема Scrum. Застосовані принципи включали розбиття проекту на спринти, регулярні зустрічі команди, ретроспективи та інтеграційне тестування. Це дозволило створити гнучку систему управління, яка ефективно адаптується до змінних вимог та забезпечує високу продуктивність команди.[11]

Під час управління проектом були реалізовані наступні основні аспекти:

1) Планування спринтів: Проект був розділений на кілька спринтів тривалістю 2 тижні кожен. Це дозволило структурувати роботу, чітко визначити задачі та терміни їх виконання. Практичне застосування показало, що такий підхід сприяє своєчасному виконанню завдань та ефективному використанню ресурсів.
2) Щоденні зустрічі: Регулярні щоденні зустрічі команди дозволили швидко вирішувати поточні проблеми, координувати дії та виявляти можливі ризики. Це забезпечило прозорість процесу та сприяло покращенню комунікації між учасниками проекту.
3) Огляд спринтів: Після кожного спринту проводився огляд виконаної роботи, що дозволило оцінити досягнення, виявити недоліки та визначити напрямки для подальшого вдосконалення. Такий підхід сприяв постійному покращенню процесів розробки та підвищенню якості кінцевого продукту.

4) Ретроспективи: Ретроспективи спринтів дозволяли команді аналізувати свої дії, виявляти успішні практики та помилки, а також розробляти плани для покращення роботи у наступних спринтах. Це забезпечило безперервне вдосконалення та підвищення ефективності команди.

Застосування методології Scrum забезпечило ефективне управління проектом, що дозволило адаптуватися до змінних вимог замовника та забезпечити високу якість продукту. Регулярні зустрічі та ретроспективи сприяли покращенню процесів та підвищенню ефективності роботи команди. Завдяки Scrum команда змогла вчасно виконувати задачі, що було важливим для успішного завершення проекту.

Проведене тестування та оцінка результатів управління проектом показали високу ефективність обраної методології. Всі основні етапи проекту були виконані вчасно, що підтвердило дієвість використаних підходів. Зокрема, успішно проведене функціональне та навантажувальне тестування продемонструвало, що обрані методи управління дозволили досягти високої продуктивності та надійності кінцевого продукту.

Учасники проекту відзначили, що застосування методології Scrum сприяло покращенню комунікації, чіткості у визначенні задач та підвищенню мотивації. Це підтверджує, що обраний підхід до управління проектом був ефективним і забезпечив досягнення поставлених цілей.

Таким чином, проведений аналіз результатів управління проектом показав, що використання теоретичних та практичних знань у сфері Scrum дозволило ефективно організувати роботу команди, забезпечити своєчасне виконання задач та досягти високої якості кінцевого продукту. Практичне застосування методології Scrum забезпечило гнучкість, прозорість та ефективність управління проектом, що дозволяє рекомендувати цей підхід для подальшого використання у подібних проектах.

4.2 Перспективи подальших досліджень

Планується розширення функціональності додатку для підтримки нових типів даних та джерел, а також вдосконалення існуючих модулів. Основні напрямки розвитку:

1) Інтеграція з новими API: Розширення кількості підтримуваних API для отримання фінансових даних, що дозволить підвищити точність та актуальність інформації.
    a) Планується інтеграція з новими фінансовими платформами, такими як Bloomberg API, Alpha Vantage та інші.
    b) Розробка додаткових модулів для автоматичного оновлення конфігурацій API.
2) Вдосконалення алгоритмів обробки: Розробка нових алгоритмів для більш точної та ефективної обробки даних, зокрема використання методів машинного навчання.
    a) Впровадження моделей машинного навчання для прогнозування фінансових показників.
    b) Розробка алгоритмів для автоматичного виявлення аномалій та шахрайських дій у фінансових даних.

Для роботи з великими обсягами даних планується оптимізація продуктивності додатку:

1) Паралельна обробка даних: Впровадження методів паралельної обробки для збільшення швидкості роботи додатку.
    a) Використання багатопоточних алгоритмів для паралельного оброблення даних.
    b) Розподіл обчислювальних завдань між кількома серверами для зниження навантаження.

2) Масштабування інфраструктури: Використання додаткових хмарних ресурсів для забезпечення стійкості та продуктивності додатку при високих навантаженнях.
    a) Впровадження горизонтального масштабування для підтримки зростаючих обсягів даних.
    b) Використання контейнерних технологій (Docker, Kubernetes) для управління масштабуванням та розгортанням сервісів.

Для підвищення надійності та безпеки додатку плануються наступні заходи:

1) Резервне копіювання даних: Впровадження регулярного резервного копіювання даних для захисту від втрати інформації.
    a) Автоматизація процесу резервного копіювання з використанням хмарних сховищ.
    b) Регулярне тестування резервних копій для забезпечення їхньої цілісності та доступності.
2) Посилення заходів безпеки: Впровадження додаткових заходів для захисту даних від несанкціонованого доступу та кібератак.
    a) Використання шифрування даних як при зберіганні, так і при передачі.
    b) Впровадження багатофакторної аутентифікації для доступу до критичних компонентів додатку.

4.3 Висновки

Розроблений додаток збору, обробки та зберігання фінансових даних показала високу ефективність, гнучкість та надійність. Вона забезпечує зручний та швидкий доступ до фінансових даних, що дозволяє користувачам приймати

обґрунтовані рішення на основі актуальної інформації. Основні досягнення включають:

1) Висока швидкість та точність збору даних: Завдяки паралельній обробці та оптимізованим алгоритмам, додаток забезпечує швидке та точне отримання даних з різних джерел.
2) Гнучкість та масштабованість: Мікросервісна архітектура дозволяє легко адаптувати додаток до нових вимог та збільшувати її масштаб без втрати продуктивності.
3) Надійність та безпека зберігання даних: Впроваджені механізми забезпечують високу надійність зберігання та захист даних від несанкціонованого доступу.

Перспективи подальших досліджень включають розширення функціональності додатку, оптимізацію продуктивності та підвищення надійності і безпеки. Реалізація цих заходів дозволить забезпечити подальший розвиток додатку та її адаптацію до нових вимог ринку, що сприятиме її широкому впровадженню та використанню в різних галузях фінансової діяльності.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено дослідження методів та моделей, що використовуються при розробці додатку для збору фінансових даних. Метою роботи було розробити підхід, який забезпечить ефективний збір, обробку та зберігання фінансових даних з різних джерел.

Було проведено детальний аналіз існуючих методів збору та обробки фінансових даних, включаючи веб-скрапінг, API інтеграцію та передачу даних через FTP/SFTP. Визначення основних переваг та недоліків кожного методу дозволило сформулювати оптимальний підхід для вирішення поставлених задач.

Розроблено архітектуру додатку, яка включає три основні модулі: Скрапер, Екстрактор та Маркет логіки. Створено конфігураційний механізм, що дозволяє налаштовувати параметри роботи додатку відповідно до конкретних потреб. Реалізовано модуль Скрапер для збору даних з веб-сайтів, API та FTP/SFTP серверів. Створено модуль Екстрактор для парсингу, фільтрації та зборки даних. Впроваджено модуль Маркет логіки для обробки даних та їх зберігання у базі даних.

Завдяки використанню методології Scrum було забезпечено ефективне управління проектом, що дозволило вчасно виконувати задачі та адаптуватися до змінних вимог. Регулярні зустрічі команди та ретроспективи сприяли покращенню процесів та підвищенню ефективності роботи.

На основі проведеного дослідження було сформульовано перспективи подальших досліджень. Розширення функціональності додатку передбачає інтеграцію з новими API для отримання фінансових даних з більшої кількості джерел та вдосконалення алгоритмів обробки даних за допомогою методів машинного навчання. Оптимізація продуктивності включає впровадження методів паралельної обробки для збільшення швидкості роботи додатку та використання додаткових хмарних ресурсів для масштабування інфраструктури. Підвищення надійності та безпеки передбачає впровадження регулярного

резервного копіювання даних для захисту від втрати інформації та посилення заходів безпеки для захисту даних від несанкціонованого доступу та кібератак.

Таким чином, розроблений додаток збору, обробки та зберігання фінансових даних відповідає сучасним вимогам та потребам користувачів, забезпечуючи високу ефективність, гнучкість та надійність. Подальший розвиток додатку дозволить підвищити його продуктивність та функціональні можливості, що сприятиме його широкому впровадженню у фінансовій сфері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Schwaber, K. (2004). Agile Project Management with Scrum. Microsoft Press. ISBN: 978-0735619937.
2) Sutherland, J. (2014). Scrum: The Art of Doing Twice the Work in Half the Time. Crown Business. ISBN: 978-0385346450.
3) Shroff, G. (2010). Enterprise Cloud Computing: Technology, Architecture, Applications. Cambridge University Press. ISBN: 978-0521137355.
4) Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. ISBN: 978-0262035613.
5) McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media. ISBN: 978-1491957660.
6) Williams, N. (2015). Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data. Wiley. ISBN: 978-1118876138.
7) Chen, M., Mao, S., & Liu, Y. (2014). Big Data: Related Technologies, Challenges and Future Prospects. Springer. ISBN: 978-3319062459.
8) Kleppmann, M. (2017). Designing Data-Intensive Applications. O'Reilly Media. ISBN: 978-1449373320.
9) Downey, A. (2012). Think Python: How to Think Like a Computer Scientist. O'Reilly Media. ISBN: 978-1491939360.
10) Jensen, D. (2017). Programming in Python 3. Addison-Wesley Professional. ISBN: 978-0321680563.
11) Fitzgerald, B., & Stol, K. (2017). Continuous Software Engineering. Springer. ISBN: 978-3319631990.
12) Левикін В.М., Діденко Д.О., Альошкін А.О. Метод формування заявок природною мовою на основі вдосконаленої моделі BERT // АСУ та прилади автоматики. 2024. № 180. С. 55-71.
13) Методичні вказівки щодо розробки та оформлення магістерської кваліфікаційної роботи за спеціальністю 122 Комп’ютерні науки (освітня програма «Управління проєктами в галузі інформаційних технологій»

освітньо-кваліфікаційного рівня «магістр» / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2024. – 28 с.