ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
(повне найменування вищого навчального закладу)
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
(повне найменування інституту, назва факультету (відділення))
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до кваліфікаційної роботи
магістра
(рівень вищої освіти)

на тему: «Розробка системи реального часу для моніторингу глобальних новин з використанням асинхронних повідомлень через Telegram/Discord-бота»

Виконав: студент 6 курсу, групи 6ПР
спеціальності
121 «Інженерія програмного забезпечення» »
(шифр і назва напряму підготовки, спеціальності)

Яловик Олександр Вікторович
(прізвище та ініціали)

Керівник к.т.н., доцент Огнєва О.Є.
(прізвище та ініціали)

Рецензент к.т.н., доц.Корніловська Н.В.
(прізвище та ініціали)

Хмельницький – 2025р.

Херсонський національний технічний університет

(повне найменування вищого навчального закладу)

Інститут, факультет, відділення Факультет інформаційних технологій та дизайну
Кафедра, циклова комісія Кафедра програмних засобів і технологій
Освітньо-кваліфікаційний рівень другий (магістерський)
Напрям підготовки ОПП – Програмне забезпечення систем
(шифр і назва)
Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗіТ
к.т.н., доц. Огнєва О.Є.
“____” _____ 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Яловик Олександр Вікторович
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка системи реального часу для моніторингу глобальних новин з використанням асинхронних повідомлень через Telegram/Discord-бота
керівник роботи к.т.н. доцент Огнєва О.Є.,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджена наказом вищого навчального закладу “15” вересня 2025 року № 416-с
2. Строк подання студентом роботи 01.09.2025
3. Вихідні дані до роботи

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження методів, моделей та підходів до моніторингу глобальних новин у режимі реального часу; 2. Дослідження інформаційних технологій та програмних засобів для асинхронної обробки новин; 3. Проєктування системи реального часу для моніторингу глобальних новин; 4. Розробка програмної системи та опис її функціонування.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Комп’ютерна презентація

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Відбір та вивчення літературних джерел	01.09.2025 – 27.09.2025	
2	Аналіз стану вирішення завдання на сучасному етапі	27.09.2025 – 11.10.2025	
3	Побудова концептуальної моделі	11.10.2025 – 23.10.2025	Концептуальна схема
4	Розробка моделі	23.10.2025 – 31.10.2025	Функціональна схема
5	Побудова алгоритму функціонування додатку	31.10.2025 – 09.11.2025	Схема варіантів використання
6	Написання вихідного коду програми	09.11.2025 – 16.11.2025	
7	Налагодження програмного коду	16.11.2025– 22.11.2025	
8	Оформлення пояснювальної записки	23.11.2025 – 06.12.2025	

Студент _____ **(О.В.Яловик)**_____
(підпис) (прізвище та ініціали)

Керівник роботи _____ **(О.Є.Огнєва)**
(підпис) (прізвище та ініціа

АНОТАЦІЯ

Кваліфікаційна робота магістра містить такі структурні частини: вступ, три розділи, висновок та список використаних джерел.

Перший розділ «Дослідження методів, моделей та підходів до моніторингу глобальних новин у режимі реального часу» складається з чотирьох частин: «Поняття та особливості систем реального часу», «Глобальні інформаційні потоки та джерела новин», «Подійно-орієнтований підхід та асинхронні повідомлення в системах моніторингу» та «Аналіз існуючих систем моніторингу новин та обґрунтування розробки власного рішення».

У даному розділі проведено аналіз предметної області дослідження, розглянуто основні підходи до обробки великих інформаційних потоків, а також наведено приклади сучасних інформаційних систем для моніторингу новин.

Другий розділ «Аналіз інформаційних технологій та програмних засобів для реалізації системи моніторингу глобальних новин» містить такі підрозділи: «Аналіз технологій збору новинної інформації», «Аналіз технологій асинхронної обробки даних та обміну повідомленнями» та «Аналіз каналів доставки новинної інформації користувачам».

У цьому розділі розглянуто сучасні інформаційні технології, методи та програмні засоби, що застосовуються для побудови систем реального часу, а також проаналізовано можливості використання Telegram- та Discord-ботів для доставки новин користувачам.

Третій розділ «Проєктування та розробка системи реального часу для моніторингу глобальних новин» складається з таких підрозділів: «Загальна архітектура системи моніторингу глобальних новин», «Опис функціональних модулів системи», «Реалізація програмної системи моніторингу глобальних новин» та «Алгоритм функціонування системи».

У даному розділі описано архітектуру розроблюваної системи, обґрунтовано вибір алгоритмів та технологій, а також наведено логіку взаємодії основних модулів системи.

Четвертий розділ «Проєктування та реалізація системи моніторингу глоабильних новин у режимі реального часу» складається з п’яти підрозділів: «Загальна архітектура та модульна структура системи», «Організація зберігання даних та керування станом системи», «Реалізація модуля моніторингу новин (бекграунд-сервіс)», «Аналіз роботи та тестування системи моніторингу новин» та «Можливі напрями вдосконалення системи моніторингу новин».

У цьому розділі наведено програмну реалізацію розробленої системи, описано роботу кожного модуля, проведено тестування та проаналізовано результати її функціонування.

ANNOTATION

The master's qualification thesis consists of the following structural components: an introduction, four chapters, a conclusion, and a list of references.

The first chapter, “Research of Methods, Models, and Approaches to Real-Time Global News Monitoring,” consists of four sections: “The Concept and Features of Real-Time Systems,” “Global Information Flows and News Sources,” “Event-Driven Approach and Asynchronous Messaging in Monitoring Systems,” and “Analysis of Existing News Monitoring Systems and Justification for Developing a Custom Solution.”

This chapter analyzes the research domain, examines the main approaches to processing large-scale information flows, and presents examples of modern information systems for news monitoring.

The second chapter, “Analysis of Information Technologies and Software Tools for Implementing a Global News Monitoring System,” includes the following sections: “Analysis of Technologies for Collecting News Information,” “Analysis of Technologies for Asynchronous Data Processing and Message Exchange,” and “Analysis of Channels for Delivering News Information to Users.”

This chapter reviews modern information technologies, methods, and software solutions used in the development of real-time systems, as well as analyzes the possibilities of using Telegram and Discord bots for delivering news to users.

The third chapter, “Design and Development of a Real-Time System for Global News Monitoring,” includes the following sections: “General Architecture of the Global News Monitoring System,” “Description of the System’s Functional Modules,” “Implementation of the Global News Monitoring Software System,” and “Algorithm of the System’s Operation.”

This chapter describes the architecture of the developed system, justifies the choice of algorithms and technologies, and presents the interaction logic of the main system modules.

The fourth chapter, “Design and Implementation of a Real-Time Global News Monitoring System,” consists of five sections: “General Architecture and Modular Structure of the System,” “Organization of Data Storage and System State Management,” “Implementation of the News Monitoring Module (Background Service),” “Analysis and Testing of the News Monitoring System,” and “Possible Directions for Improving the News Monitoring System.”

This chapter presents the software implementation of the developed system, describes the functionality of each module, provides testing results, and analyzes the performance of the system.

РЕФЕРАТ

Кваліфікаційна робота магістра: 101 сторінок, 19 рисунків, 2 таблиці, 38 джерела.

Мета роботи – розроблення системи реального часу для моніторингу глобальних новин із використанням асинхронних повідомлень через Telegram- та Discord-бота. Для реалізації поставленої мети необхідно проаналізувати особливості побудови систем реального часу, методи асинхронної обробки даних та підходи до збору і фільтрації новинної інформації.

Об’єкт дослідження – процеси моніторингу та розповсюдження глобальних новин в інформаційних системах реального часу.

Предмет дослідження – методи, моделі та алгоритми асинхронної обробки повідомлень у системах моніторингу новин із використанням месенджерів Telegram і Discord.

Результат роботи: розроблена система забезпечує автоматизований збір, обробку, фільтрацію та доставку новинної інформації у режимі реального часу користувачам через Telegram- та Discord-ботів. Запропонована система може бути використана для оперативного моніторингу подій у сферах економіки, технологій, фінансів, криптовалют та суспільно-політичних процесів.

Новизна роботи: удосконалено підхід до моніторингу глобальних новин у режимі реального часу на основі асинхронної архітектури з використанням месенджерів як каналів доставки інформації. Розроблено модель взаємодії модулів системи, що забезпечує масштабованість та мінімальну затримку передачі повідомлень.

Реалізовано програмне рішення, яке може бути адаптоване до підключення додаткових інформаційних джерел та каналів поширення новин.

Ключові слова: система реального часу, моніторинг новин, асинхронна обробка, Telegram-бот, Discord-бот, інформаційні потоки.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Скорочення, термін, позначення	Пояснення
API	Application Programming Interface
API Rate Limit	Обмеження кількості запитів до програмного інтерфейсу
Async	Асинхронна обробка даних
Bot	Програмний агент для автоматичної взаємодії з користувачем
CPU	Central Processing Unit
DB	База даних
Discord	Мережевий месенджер для обміну повідомленнями
HTTP	HyperText Transfer Protocol
HTTPS	Захищений протокол передачі даних
JSON	JavaScript Object Notation
Log	Журнал подій системи
MQ	Message Queue
OS	Operating System
Polling	Періодичне опитування джерела даних
Queue	Черга повідомлень
Real-Time System (RTS)	Система реального часу
REST	Representational State Transfer
RSS	Really Simple Syndication
Scraping	Автоматизований збір даних з веб-сторінок

Telegram	Месенджер для обміну повідомленнями
Webhook	Механізм взаємодії між системами за подіями
X (Twitter)	Соціальна мережа для обміну короткими повідомленнями

ЗМІСТ

ВСТУП

На сьогоднішній день актуальність оперативного отримання та аналізу глобальних новин стрімко зростає у зв’язку з активним розвитком інформаційних технологій, цифрових платформ та збільшенням обсягів інформаційних потоків. Сучасне суспільство постійно стикається з необхідністю швидкого реагування на події у сферах політики, економіки, фінансів, технологій та безпеки. Використання систем реального часу для моніторингу новин дозволяє забезпечити своєчасне інформування користувачів та прийняття рішень на основі актуальних даних.

Значний вплив на процес розповсюдження інформації мають соціальні мережі та цифрові платформи, які стали одним із основних джерел новин у реальному часі. Особливу популярність у сучасних інформаційних системах набуває використання асинхронних механізмів обробки даних, що дозволяє забезпечити масштабованість, стабільність та мінімальні затримки при передачі повідомлень. Важливу роль у цьому процесі відіграють месенджери, зокрема Telegram та Discord, які використовуються як ефективні засоби оперативної доставки новинної інформації кінцевим користувачам.

Актуальність теми. Актуальність теми дослідження зумовлена необхідністю створення систем, здатних здійснювати моніторинг глобальних новин у режимі реального часу та забезпечувати їх автоматизовану доставку з використанням сучасних інформаційних технологій. Розроблення спеціалізованої системи моніторингу новин із використанням асинхронних повідомлень через Telegram- та Discord-ботів дозволить підвищити ефективність обробки інформаційних потоків та розширити можливості оперативного інформування користувачів.

Актуальність теми дослідження обумовлена наступними чинниками:

- Зростання обсягів глобальних інформаційних потоків;

- Необхідність оперативного отримання актуальних новин у режимі реального часу;
- Широке використання месенджерів як основних каналів комунікації;
- Обмежена кількість універсальних програмних рішень для моніторингу новин із використанням асинхронних архітектур;
- Потреба у масштабованих та надійних системах обробки подій у реальному часі.

Мета і задачі дослідження. Метою дослідження є розробка системи реального часу для моніторингу глобальних новин з використанням асинхронних повідомлень через Telegram- та Discord-бота.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

- Вивчити особливості побудови систем реального часу;
- Дослідити методи та алгоритми збору, обробки і фільтрації новинної інформації;
- Проаналізувати існуючі системи та сервіси моніторингу новин;
- Дослідити технології асинхронної обробки повідомлень;
- Спроєктувати архітектуру системи моніторингу глобальних новин;
- Реалізувати механізм збору та обробки новин з інформаційних джерел;
- Реалізувати Telegram- та Discord-ботів для доставки повідомлень користувачам;
- Провести тестування розробленої системи та проаналізувати отримані результати.

Об’єкт дослідження – процеси моніторингу та розповсюдження глобальних новин в інформаційних системах реального часу.

Предмет дослідження – методи, моделі та алгоритми асинхронної обробки повідомлень у системах моніторингу новин із використанням Telegram- та Discord-ботів.

Новизна роботи: У результаті виконаного дослідження одержані наступні наукові результати:

- Досліджено особливості застосування асинхронної архітектури для систем моніторингу новин;
- Розроблено модель взаємодії модулів системи реального часу;
- Реалізовано механізм автоматизованого збору та доставки новин через месенджери;
- Адаптовано систему до розширення кількості джерел та каналів розповсюдження інформації.

Практичне значення одержаних результатів. Розроблена система дає можливість здійснювати моніторинг глобальних новин у режимі реального часу та оперативно доставляти інформацію користувачам через Telegram- і Discord-ботів. Запропоноване рішення може бути використане у сферах фінансів, технологій, криптовалют, бізнесу та суспільно-політичної аналітики.

Теоретичне значення одержаних результатів. Матеріали дослідження можуть бути використані для подальшого розвитку теорії побудови систем реального часу та асинхронних інформаційних систем, а також у навчальному процесі при вивченні сучасних інформаційних технологій.

Апробація результатів бакалаврської кваліфікаційної роботи. Результати кваліфікаційної роботи можуть бути використані для подальшого удосконалення систем моніторингу інформаційних потоків та розробки аналогічних програмних рішень.

Публікації:

1. Яловик О. В. Асинхронна система моніторингу глобальних новин у реальному часі // Матеріали студентської науково-практичної конференції «Інформаційні технології XXI століття». – Київ, 2025. – С. 52–56.

2. Яловик О. В. Застосування Telegram- і Discord-ботів у сучасних інформаційних системах // Інформаційні технології та програмна інженерія. Національного технічного університету України «КПІ ім. Ігоря Сікорського». – Київ, 2025. – С. 88–94.

3. Яловик О. В., Яловик В. М. Використання асинхронних технологій Python для побудови сервісів реального часу // Технології програмування: наукові дослідження та розробки. – Київ, 2025. – С. 133–138.

4. Яловик О. В. Автоматизація збору та фільтрації новинних потоків із застосуванням штучного інтелекту // Науковий вісник кафедри комп’ютерних наук. – Київ, 2025. – No 5 (19). – С. 102–107.

Структура: робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, додатку. Загальний обсяг роботи – 101 сторінок.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МЕТОДІВ, МОДЕЛЕЙ ТА ПІДХОДІВ ДО МОНІТОРИНГУ ГЛОБАЛЬНИХ НОВИН У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

1.1. Поняття та особливості систем реального часу

У сучасному інформаційному суспільстві все більшого значення набувають системи реального часу, які забезпечують оперативну обробку та передачу інформації з мінімальною затримкою. Активний розвиток цифрових технологій, інтернет-платформ та соціальних мереж призвів до значного зростання обсягів інформаційних потоків, що вимагає застосування ефективних підходів до їх збору, аналізу та розповсюдження. Особливо це стосується глобальних новин, які впливають на прийняття рішень у сферах економіки, фінансів, бізнесу, безпеки та суспільно-політичного життя [6, 13, 19].

Система реального часу – це інформаційна система, яка повинна реагувати на вхідні події або зміну стану зовнішнього середовища у заздалегідь визначений час. Основною відмінністю таких систем від традиційних інформаційних систем є не тільки коректність обчислень, але й дотримання часових обмежень при обробці даних. Порушення часових параметрів може призводити до втрати актуальності інформації або зниження ефективності всієї системи [13, 20, 24].

У контексті моніторингу глобальних новин системи реального часу відіграють ключову роль, оскільки новинна інформація має високу динаміку та швидко застаріває. Користувачі очікують отримання повідомлень про важливі події практично миттєво, що зумовлює необхідність побудови систем із мінімальною затримкою між моментом появи новини та її доставкою кінцевому користувачеві [19, 31].

Системи реального часу можуть бути класифіковані за різними ознаками, зокрема за жорсткістю часових обмежень. Залежно від критичності дотримання часових вимог виділяють жорсткі, умовно-жорсткі та м’які системи реального часу, що схематично представлено на рисунку 1.1. У жорстких системах критично важливо дотримуватися встановлених часових рамок, оскільки навіть незначне перевищення допустимого часу може призвести до серйозних наслідків. У м’яких системах допускається незначне порушення часових вимог, яке не призводить до відмови системи, проте може знижувати якість її функціонування [13, 20, 24]. Системи моніторингу новин, як правило, належать до м’яких систем реального часу, оскільки незначна затримка доставки повідомлень не призводить до критичних наслідків, однак істотно впливає на актуальність інформації.

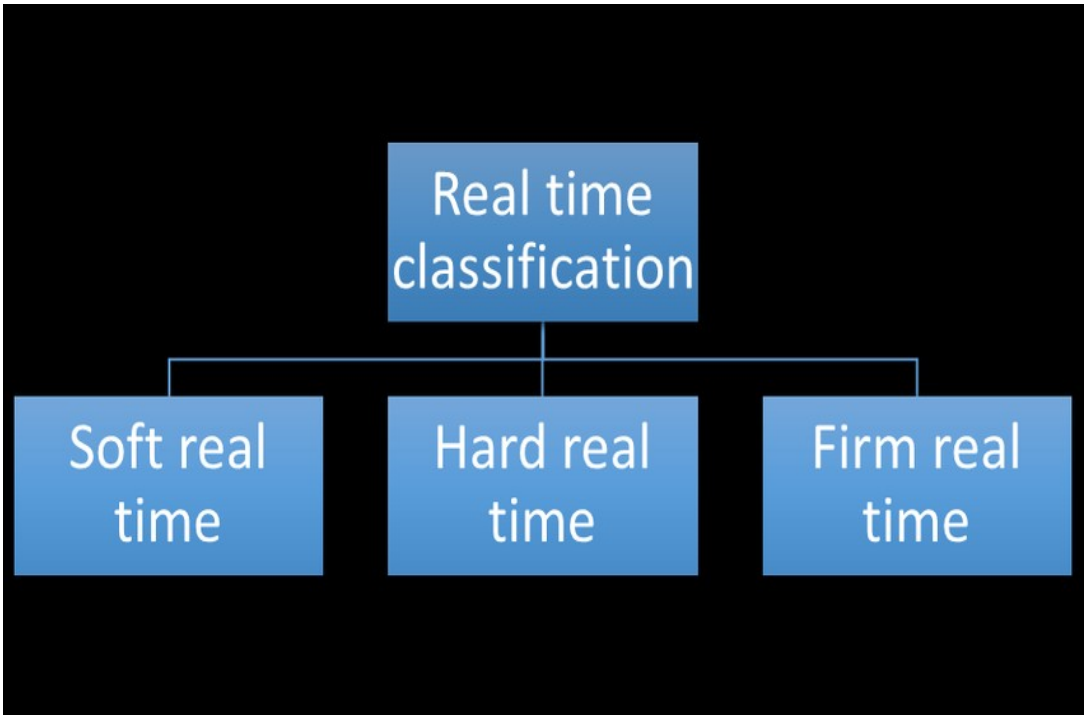

Рис. 1.1 – Класифікація систем реального часу за часовими обмеженнями [22]

Однією з ключових особливостей систем реального часу є їх здатність обробляти велику кількість подій, що надходять із різних джерел, у режимі безперервного потоку. Для цього застосовуються спеціалізовані архітектурні підходи, які забезпечують масштабованість та високу продуктивність. У випадку моніторингу новин джерелами подій можуть виступати соціальні мережі, інформаційні агентства, новинні портали, API-сервіси та RSS-стрічки [6, 31].

Важливим аспектом функціонування систем реального часу є асинхронна обробка даних. Асинхронні механізми дозволяють системі одночасно обробляти декілька потоків інформації без блокування основних процесів, що наочно продемонстровано на рисунку 1.2. Такий підхід є особливо актуальним для систем моніторингу глобальних новин, у яких кількість вхідних повідомлень може різко зростати під час резонансних подій або кризових ситуацій [2-3, 22, 27, 29].

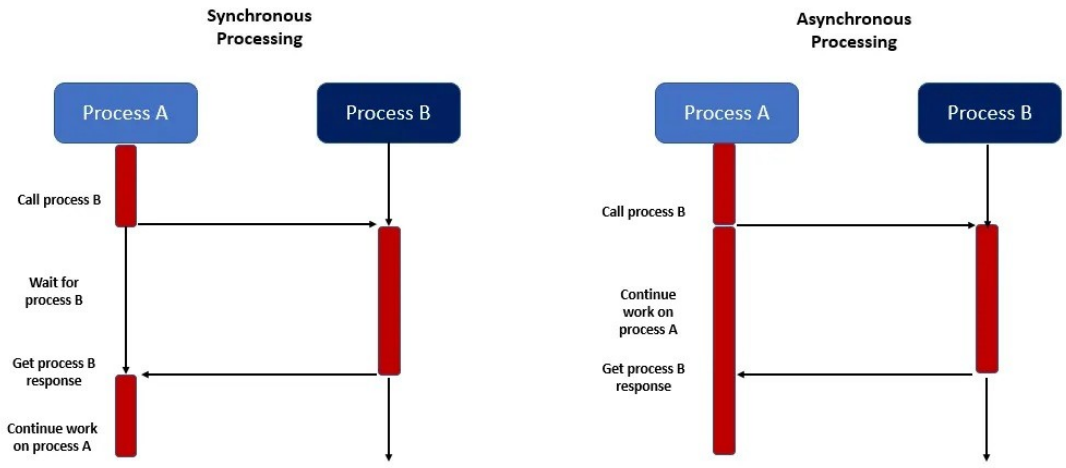

Рис. 1.2 – Порівняння синхронної та асинхронної моделей обробки процесів [27]

Застосування асинхронної архітектури дає змогу підвищити ефективність використання обчислювальних ресурсів, зменшити затримки та забезпечити стабільну роботу системи навіть за умов високого навантаження. У таких системах широко використовуються черги повідомлень, подійно-орієнтовані моделі та неблокуючі операції введення-виведення, що дозволяє забезпечити обробку даних у режимі реального часу [4, 12, 14, 17, 37].

Ще однією важливою характеристикою систем реального часу є їхня надійність. Для систем моніторингу новин важливо забезпечити безперебійну роботу, оскільки втрата частини подій або перебої у доставці повідомлень можуть призвести до неповного інформування користувачів. Тому при проєктуванні таких систем особлива увага приділяється механізмам обробки помилок, логуванню, повторним спробам доставки повідомлень та масштабуванню [6, 30].

Таким чином, системи реального часу є ключовим компонентом сучасних інформаційних рішень для моніторингу глобальних новин. Вони дозволяють забезпечити своєчасне отримання актуальної інформації, ефективну обробку великих потоків даних та оперативну доставку новин кінцевим користувачам через сучасні канали комунікації, зокрема месенджери.

Важливою характеристикою систем реального часу є спосіб організації обробки подій. Залежно від внутрішньої архітектури виділяють системи з послідовною та паралельною обробкою даних. У послідовних системах кожна подія обробляється по черзі, що може призводити до затримок при значному потоці інформації. Для систем моніторингу глобальних новин такий підхід є малоефективним, оскільки кількість подій може змінюватися динамічно та досягати пікових значень у короткі проміжки часу.

Паралельна та асинхронна обробка даних є більш доцільним підходом для реалізації систем реального часу. Вона дозволяє розподіляти навантаження між декількома потоками виконання або процесами, що суттєво підвищує пропускну здатність системи. Асинхронна модель передбачає, що операції збору, обробки та доставки новин виконуються незалежно одна від одної, що мінімізує блокування ресурсів та забезпечує стабільну роботу системи навіть при збільшенні кількості джерел інформації [19, 24, 31].

Суттєву роль у системах реального часу відіграє подійно-орієнтована архітектура. У таких системах основою функціонування є події, які виникають у результаті появи нових даних або змін стану джерел. Подіями можуть бути публікації нових новин у соціальних мережах, оновлення стрічок новин, повідомлення від API або зміни в інформаційних потоках. Кожна подія обробляється відповідними модулями системи та може ініціювати подальші дії, зокрема фільтрацію, аналіз або надсилання повідомлень користувачам[4, 12-17].

Подійно-орієнтований підхід забезпечує високу гнучкість системи та дозволяє легко масштабувати її функціональність. У разі підключення нового джерела новин або каналу доставки інформації достатньо додати відповідний обробник подій без суттєвих змін у загальній архітектурі. Це є важливою перевагою для систем глобального моніторингу, які повинні адаптуватися до постійних змін інформаційного середовища.

Ще одним ключовим аспектом систем реального часу є затримка обробки та доставки інформації. Затримка визначається як інтервал часу між моментом виникнення події та моментом її отримання кінцевим користувачем. Для систем моніторингу новин мінімізація затримки є одним із головних критеріїв ефективності. Висока затримка може призвести до втрати актуальності новинної інформації та зниження довіри користувачів до системи.

Зменшення затримок досягається за рахунок оптимізації алгоритмів обробки, використання асинхронних механізмів введення-виведення, черг повідомлень та ефективних протоколів передачі даних. У сучасних системах широко застосовуються неблокуючі мережеві взаємодії, що дозволяє системі обробляти велику кількість одночасних запитів без втрати продуктивності.

Окрім затримки, важливою характеристикою систем реального часу є їх здатність працювати в умовах відмов або нестабільності джерел даних. Глобальні новинні джерела можуть тимчасово ставати недоступними, змінювати формати даних або обмежувати доступ до API. Тому система моніторингу повинна бути спроєктована з урахуванням можливих збоїв, використовуючи механізми повторних запитів, резервного збору інформації та альтернативних джерел даних.

З точки зору програмної реалізації систем реального часу важливим є вибір відповідних технологій та інструментів, що забезпечують асинхронну обробку даних і подій. Сучасні мови програмування та фреймворки надають широкі можливості для створення подійно-орієнтованих і масштабованих систем, здатних обробляти великі обсяги інформації з мінімальними затримками. Застосування таких підходів дозволяє будувати гнучкі архітектурні рішення, які відповідають вимогам реального часу.

У системах моніторингу глобальних новин особливу увагу необхідно приділяти не лише обробці інформаційних потоків, але й контролю показників роботи системи. Потік даних з глобальних джерел може характеризуватися високою інтенсивністю, нерівномірністю надходження та наявністю дубльованої або нерелевантної інформації. Тому важливо забезпечити постійний моніторинг стану системи, якості обробки даних та результатів доставки повідомлень користувачам.

Оцінювання ефективності систем реального часу є важливим етапом їх проєктування та експлуатації. Для систем моніторингу глобальних новин основними критеріями якості виступають показники продуктивності, надійності, масштабованості та своєчасності доставки інформації. Узагальнену класифікацію показників, що підлягають моніторингу в системах реального часу, наведено на рисунку 1.5. Сукупний аналіз таких показників дозволяє оцінити відповідність системи вимогам реального часу та забезпечити стабільну її роботу за умов високих навантажень.

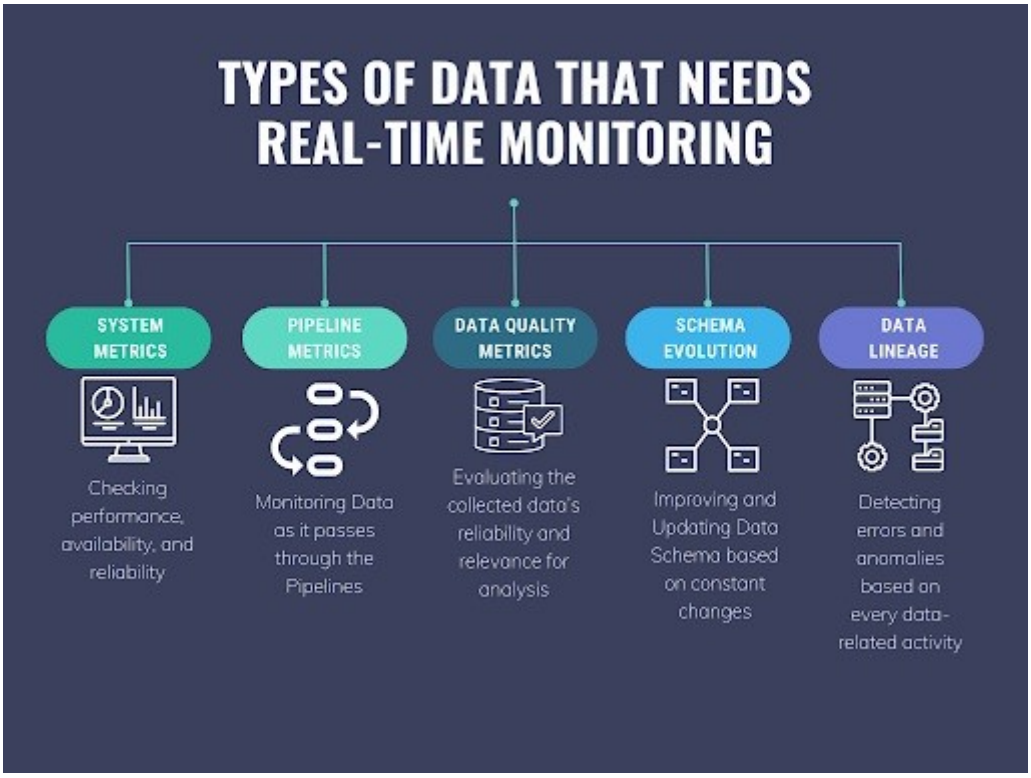

Рис. 1.3 – Основні типи показників, що потребують моніторингу в системах реального часу [12]

Одним із ключових показників ефективності є пропускна здатність системи, яка визначає кількість подій або повідомлень, що можуть бути оброблені за одиницю часу. Для систем моніторингу новин важливо зберігати стабільну продуктивність навіть у періоди різкого зростання інформаційних потоків. Асинхронна архітектура та використання черг повідомлень дають змогу рівномірно розподіляти навантаження між компонентами системи та запобігати виникненню критичних перевантажень.

Іншим важливим критерієм є час відгуку системи, який характеризує проміжок часу між появою новини в інформаційному джерелі та її отриманням кінцевим користувачем. Для систем реального часу мінімізація цього показника є вкрай важливою, оскільки безпосередньо впливає на актуальність новинної інформації. Зменшення часу відгуку досягається шляхом оптимізації алгоритмів обробки подій, використання неблокуючих операцій та ефективних механізмів взаємодії між модулями системи [12, 27-29].

Надійність системи реального часу визначається її здатністю безперервно функціонувати в умовах збоїв або нестабільності зовнішніх джерел даних. Для систем моніторингу глобальних новин важливо забезпечити збереження цілісності інформації та можливість відновлення після помилок без втрати критично важливих повідомлень. Реалізація механізмів журналювання подій, повторної обробки повідомлень і резервування джерел даних дозволяє підвищити рівень надійності системи[13, 19].

Масштабованість є ще одним важливим критерієм ефективності систем реального часу. У процесі експлуатації може виникати необхідність підключення нових джерел новин або збільшення кількості користувачів системи. Асинхронні та модульні підходи до проєктування дозволяють розширювати систему без істотних змін її базової архітектури, забезпечуючи можливість довготривалої та стабільної експлуатації.

Окрему увагу в системах моніторингу глобальних новин слід приділяти механізмам фільтрації та пріоритизації повідомлень. Не вся інформація має однакову цінність для користувачів, тому застосування алгоритмів фільтрації за ключовими словами, джерелами та тематикою дозволяє підвищити релевантність доставлених повідомлень і зменшити інформаційне навантаження. Якість таких алгоритмів визначається точністю відбору новин і швидкістю їх обробки, що також належить до ключових показників ефективності системи.

Таким чином, якість систем реального часу для моніторингу глобальних новин визначається сукупністю технічних та функціональних показників, контроль яких є невід’ємною частиною їх проєктування та експлуатації. Урахування наведених критеріїв дозволяє створювати ефективні, надійні та масштабовані програмні рішення, здатні забезпечити своєчасне інформування користувачів у динамічному інформаційному середовищі [13, 19, 31].

1.2. Глобальні інформаційні потоки та джерела новин

У сучасному цифровому середовищі глобальні інформаційні потоки стали одним із ключових чинників формування інформаційної картини світу. Стрімкий розвиток інтернету, мобільних технологій і соціальних мереж зумовив суттєве зростання обсягів даних, що генеруються та розповсюджуються у режимі реального часу. Новинна інформація надходить із численних джерел та характеризується високою швидкістю оновлення, що створює необхідність використання спеціалізованих систем для її оперативного збору, обробки та аналізу [19, 31].

Глобальні інформаційні потоки відрізняються неоднорідною структурою та різноманіттям форматів подання даних. До основних джерел новин належать інформаційні агентства, новинні портали, соціальні мережі, блоги, форуми, а також автоматизовані канали розповсюдження інформації, зокрема API та RSS. Кожен із зазначених типів джерел має власні особливості формування та передачі інформації, що безпосередньо впливає на підходи до їх моніторингу [19, 31].

Однією з характерних рис глобальних інформаційних потоків є нерівномірність надходження новинної інформації. Інтенсивність інформаційних потоків може суттєво змінюватися залежно від суспільно-політичної обстановки, економічних змін або надзвичайних подій. У періоди підвищеної інформаційної активності система моніторингу повинна забезпечувати стабільну роботу та своєчасну обробку великої кількості повідомлень без втрати продуктивності [13, 19].

Важливу роль у формуванні глобальних новинних потоків відіграють соціальні мережі, які стали одним із найбільш оперативних каналів поширення інформації. Повідомлення користувачів, офіційні публікації державних органів, компаній та медіаресурсів часто з’являються в соціальних мережах раніше, ніж на традиційних новинних платформах. У зв’язку з цим соціальні мережі є важливим джерелом даних для систем моніторингу новин у реальному часі [19, 31].

Разом із високою швидкістю поширення інформації соціальні мережі характеризуються значною кількістю шуму, дубльованих та неперевірених повідомлень. Це зумовлює необхідність використання ефективних механізмів фільтрації, агрегації та пріоритизації новинної інформації. Об’єднання повідомлень з різних джерел, що стосуються однієї події, дозволяє формувати узагальнене уявлення про подію та зменшувати інформаційне навантаження на користувача [18, 31].

Окрім соціальних мереж, важливим джерелом глобальних новин є інформаційні агентства та новинні портали, які зазвичай надають структуровану та верифіковану інформацію. Взаємодія з такими джерелами часто здійснюється за допомогою відкритих програмних інтерфейсів або RSS-каналів, що дозволяє автоматизувати процес збору новинної інформації. Використання API забезпечує стандартизований доступ до даних, можливість фільтрації за тематикою, мовою або регіоном, а також контроль обсягу оброблюваної інформації [6, 19, 31].

Разом з тим, доступ до зовнішніх джерел інформації може супроводжуватися технічними обмеженнями, такими як ліміти на кількість запитів або обмеження швидкості обміну даними. Це вимагає застосування механізмів оптимізації взаємодії з джерелами, зокрема керування запитами, кешування результатів та регулювання частоти звернень до сервісів. Неврахування таких аспектів може негативно впливати на стабільність роботи системи моніторингу [18, 31].

Під час проєктування систем для роботи з глобальними інформаційними потоками особливу увагу необхідно приділяти способам організації обміну даними між джерелами новин та системою обробки. Вибір відповідної моделі взаємодії безпосередньо впливає на продуктивність, масштабованість та здатність системи працювати у режимі реального часу. Саме тому в сучасних системах моніторингу новин все ширше застосовуються асинхронні підходи до обробки інформаційних потоків [12, 14, 17, 17, 19].

Вибір моделі взаємодії між джерелами новин і системою їх обробки має вирішальне значення для забезпечення ефективного моніторингу глобальних інформаційних потоків. Залежно від архітектурних рішень, системи можуть використовувати синхронну або асинхронну модель обміну даними, кожна з яких має власні переваги та обмеження. Синхронна модель є простою з точки зору реалізації, проте за умов високої інтенсивності інформаційних потоків може призводити до блокування процесів і зростання затримок [2, 3, 13, 27, 29].

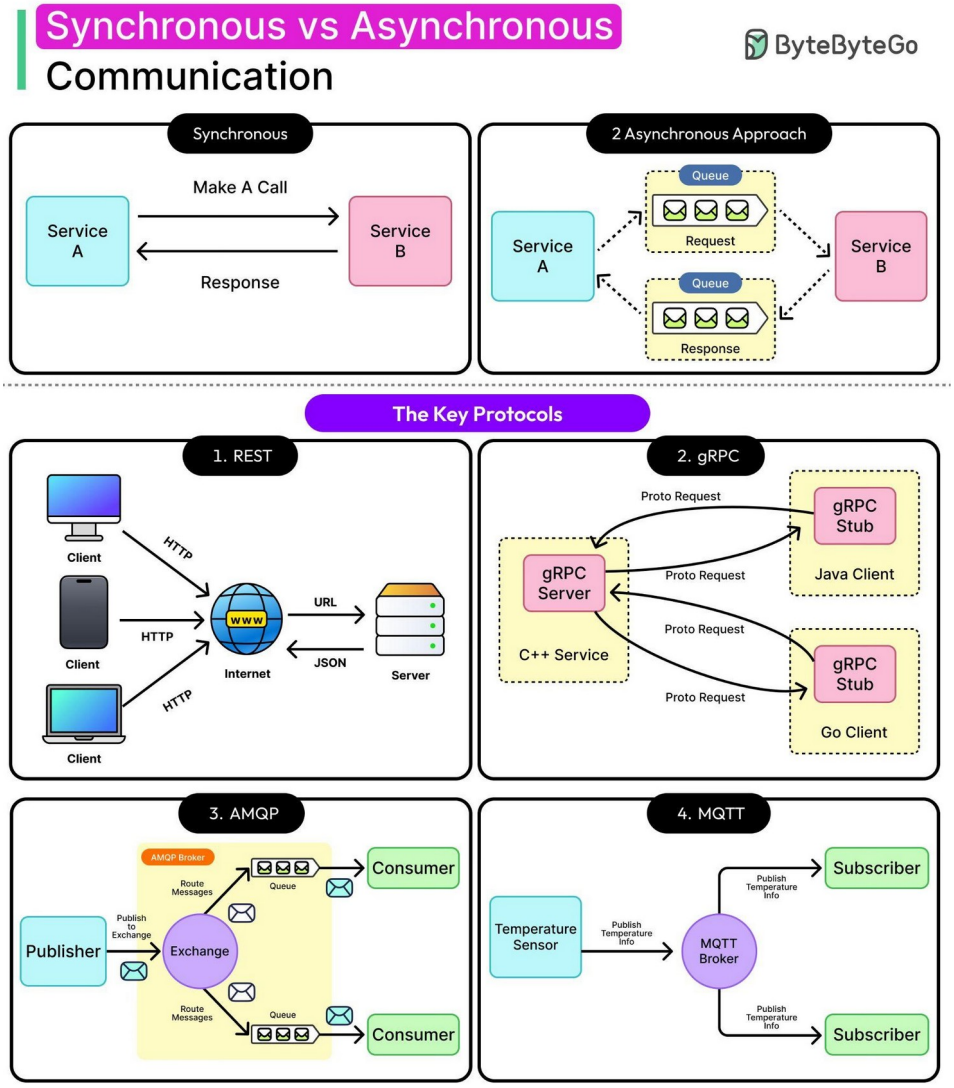

Рис. 1.4 – Порівняння синхронної та асинхронної взаємодії в інформаційних системах [31]

Асинхронна модель взаємодії є більш придатною для систем моніторингу глобальних новин, оскільки дозволяє обробляти повідомлення як послідовність подій, не залежачи від швидкості відповіді окремих джерел. Використання асинхронних механізмів обміну даними забезпечує більш стабільну роботу системи, зменшує ризик втрати повідомлень та створює умови для динамічного масштабування при зростанні кількості джерел новин або обсягу вхідних даних [2, 3, 12, 14, 17, 17, 27, 29].

Для підвищення ефективності обробки глобальних інформаційних потоків широко застосовується використання проміжних механізмів зберігання та передачі повідомлень, зокрема черг. Черги повідомлень дозволяють відокремити процес отримання новин від процесу їх обробки, що особливо важливо у періоди пікових навантажень. Завдяки такому підходу система моніторингу здатна працювати з великими обсягами інформації без суттєвого зниження продуктивності [19, 31].

Застосування асинхронної моделі взаємодії також створює сприятливі умови для інтеграції системи моніторингу з різними каналами розповсюдження інформації. Використання месенджерів і платформ обміну повідомленнями дозволяє забезпечити оперативне доведення новин до кінцевих користувачів у зручному для них форматі. Такий підхід підвищує практичну цінність системи та розширює можливості її використання в реальних умовах [6, 7, 8, 33, 34].

Таким чином, аналіз глобальних інформаційних потоків та джерел новин свідчить про доцільність використання асинхронних архітектурних рішень у системах моніторингу реального часу. Поєднання гнучких моделей взаємодії, механізмів черг повідомлень і інтеграції з сучасними каналами розповсюдження інформації створює основу для побудови ефективних, масштабованих і надійних систем моніторингу глобальних новин, що є необхідною передумовою для подальшого проєктування та реалізації програмного рішення [6, 12, 14, 17, 17, 18, 19, 31].

1.3. Подійно-орієнтований підхід та асинхронні повідомлення в системах моніторингу

Сучасні системи моніторингу глобальних новин у режимі реального часу потребують використання архітектурних підходів, здатних ефективно працювати з великою кількістю подій та асинхронних інформаційних потоків. Одним із найбільш поширених і доцільних підходів для реалізації таких систем є подійно-орієнтована архітектура, яка ґрунтується на реагуванні системи на події, що виникають у зовнішньому середовищі або всередині самої системи [4, 12, 14, 17, 17].

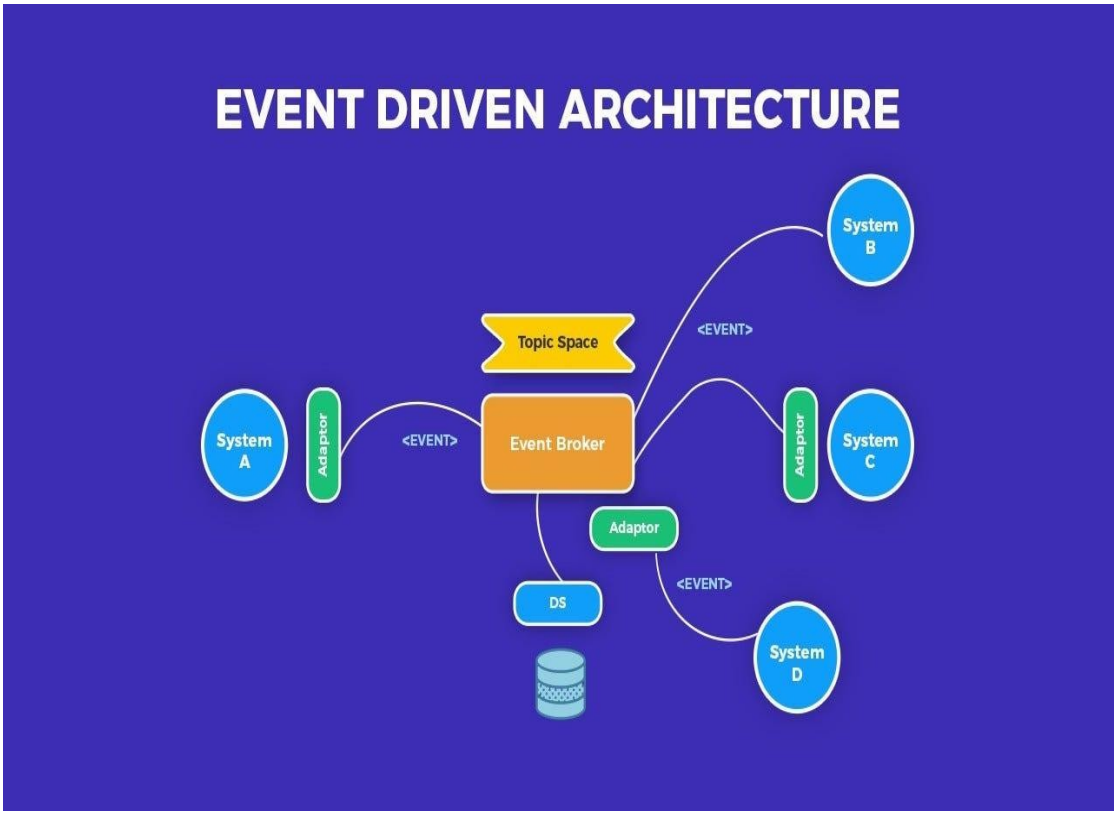

Рис. 1.5 – Загальна схема подійно-орієнтованої архітектури системи [14]

Загальну схему подійно-орієнтованої архітектури системи наведено на рисунку 1.7. У такій архітектурі інформаційні події надходять від джерел до проміжного компонента, який виконує функцію брокера повідомлень та забезпечує їх маршрутизацію до відповідних споживачів. Використання подійно-орієнтованого підходу дозволяє зменшити зв’язаність між компонентами системи, забезпечити асинхронну обробку подій та підвищити масштабованість і стійкість системи в умовах високих навантажень [12, 14, 17, 31].

Подією в контексті систем моніторингу новин може виступати публікація нового повідомлення у джерелі інформації, поява оновлення в соціальній мережі, надходження даних через API або виникнення системної події, пов’язаної зі збоєм або перевищенням навантаження. Кожна така подія ініціює виконання певного набору дій, визначених логікою функціонування системи, без необхідності її безперервного опитування джерел даних [19, 31].

Однією з ключових переваг подійно-орієнтованого підходу є відсутність жорсткої залежності між джерелом події та компонентами, які її обробляють. Це дозволяє зменшити зв’язаність між модулями системи, підвищити гнучкість архітектури та спростити подальше розширення функціональності. Для систем моніторингу глобальних новин така властивість є особливо важливою, оскільки кількість джерел та способів отримання інформації може змінюватися в процесі експлуатації [6, 31].

Асинхронні повідомлення є одним з основних механізмів реалізації подійно-орієнтованого підходу. В асинхронних системах надходження повідомлень не блокує виконання основних процесів, що дозволяє обробляти події паралельно та забезпечувати високу пропускну здатність системи. Асинхронна модель взаємодії є більш ефективною для систем реального часу, де важливо мінімізувати затримки та уникнути перевантаження окремих компонентів [2, 3, 13, 27, 29].

У системах моніторингу новин асинхронні повідомлення можуть використовуватися для передачі даних між модулями збору інформації, модулем фільтрації, компонентами аналізу та підсистемами доставки повідомлень користувачам. Такий підхід дозволяє відокремити етапи обробки інформації один від одного та забезпечити їх незалежне масштабування залежно від навантаження [12, 14, 17, 17, 19].

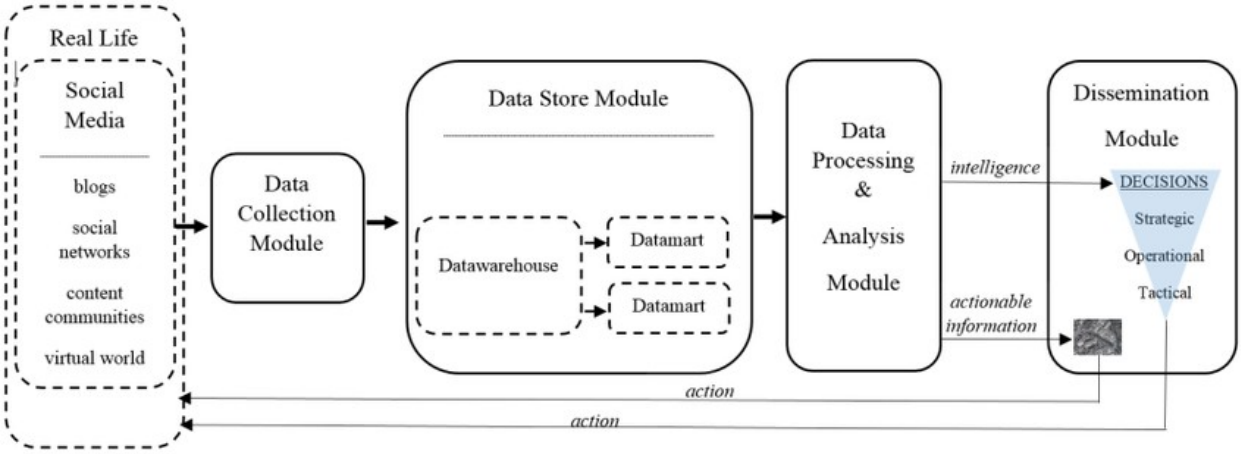

Рис. 1.6 – Узагальнена структурна схема системи моніторингу глобальних інформаційних потоків [18]

Узагальнену структурну схему системи моніторингу глобальних інформаційних потоків наведено на рисунку 1.9. Система складається з модулів збору даних, зберігання, обробки та поширення інформації, між якими реалізовано подійно-орієнтовану та асинхронну взаємодію. Така архітектура дозволяє відокремити етапи обробки подій, зменшити залежність між компонентами системи та підвищити її масштабованість і стійкість при роботі з великими обсягами новинної інформації [6, 18, 19, 31].

Важливим аспектом подійно-орієнтованих систем є використання проміжних компонентів для обміну повідомленнями, які виконують роль буфера між виробниками та споживачами подій. Наявність таких компонентів дозволяє системі залишатися стабільною навіть у періоди пікової активності, коли швидкість надходження подій перевищує можливості їх миттєвої обробки. Завдяки цьому знижується ймовірність втрати повідомлень та підвищується загальна надійність системи [4, 12, 14, 17].

Подійно-орієнтований підхід також сприяє підвищенню масштабованості систем моніторингу глобальних новин. Кожен окремий компонент системи може бути розгорнутий у множинних екземплярах, що дозволяє рівномірно розподіляти навантаження між ними. Така архітектура добре пристосована до роботи в умовах динамічного зростання кількості джерел новин або користувачів системи [6, 18, 31].

Таким чином, використання подійно-орієнтованої архітектури та асинхронних повідомлень є обґрунтованим вибором для побудови систем реального часу, орієнтованих на моніторинг глобальних новин. Застосування цих підходів дозволяє забезпечити гнучкість, масштабованість і стабільну роботу системи за умов високих інформаційних навантажень, що створює необхідні передумови для практичної реалізації програмного рішення [12, 13, 14, 17, 17].

Подійно-орієнтований підхід дозволяє ефективно організувати обробку глобальних інформаційних потоків за рахунок чіткого розмежування відповідальності між компонентами системи. Кожен модуль виконує визначену функцію та реагує лише на ті події, які належать до його області обробки. Такий підхід спрощує модифікацію та розширення системи, оскільки додавання нового джерела новин або нового каналу доставки не потребує суттєвих змін у вже реалізованих компонентах [18, 31].

Використання асинхронних повідомлень у подійно-орієнтованих системах дозволяє зменшити затримки при обробці новин та уникнути блокування виконання процесів. Кожна подія обробляється незалежно від інших, що забезпечує можливість паралельної обробки великої кількості повідомлень. Це є критично важливим для систем моніторингу глобальних новин, у яких інформаційні потоки можуть швидко зростати під час значущих суспільних або технологічних подій [2, 3, 13, 27, 29].

Ще однією важливою перевагою подійно-орієнтованого підходу є підвищення надійності системи. У разі тимчасової недоступності окремих компонентів або каналів доставки події можуть бути збережені та оброблені пізніше, без втрати інформації. Це створює умови для стабільної роботи системи навіть у разі виникнення помилок або пікових навантажень [6, 30].

Асинхронні повідомлення також сприяють підвищенню масштабованості системи моніторингу новин. За необхідності збільшення пропускної здатності можна масштабувати лише ті компоненти, на які припадає найбільше навантаження, не змінюючи загальну структуру системи. Такий підхід є економічно доцільним і дозволяє адаптувати систему до змін у кількості джерел новин або користувачів [18, 31].

Отже, подійно-орієнтована архітектура у поєднанні з асинхронними повідомленнями створює ефективну основу для побудови систем моніторингу глобальних новин у режимі реального часу. Застосування даних підходів дозволяє забезпечити високу продуктивність, гнучкість та стійкість системи, що є передумовою для її практичної реалізації та подальшого розвитку [6, 12, 13, 14, 17, 17, 18, 19, 31].

1.4. Аналіз існуючих систем моніторингу новин та обґрунтування розробки власного рішення

На сьогоднішній день існує значна кількість програмних рішень та сервісів, призначених для моніторингу новинної інформації у режимі реального часу. Такі системи використовуються для відстеження подій у сфері політики, економіки, фінансів, інформаційної безпеки та соціальних процесів. Основною метою подібних рішень є оперативне отримання та аналіз новин з різних джерел з метою своєчасного інформування користувачів [19, 31].

Більшість сучасних систем моніторингу новин ґрунтуються на використанні централізованих платформ, які агрегують інформацію з новинних сайтів, соціальних мереж та спеціалізованих інформаційних ресурсів. Такі платформи часто надають веб-інтерфейси та аналітичні інструменти, що дозволяють здійснювати пошук, фільтрацію та візуалізацію новинної інформації. Однак подібні рішення не завжди відповідають вимогам гнучкості та масштабованості, необхідним для роботи в умовах динамічних інформаційних потоків [6, 31].

Суттєвим недоліком багатьох існуючих систем є обмежені можливості налаштування та інтеграції з іншими програмними продуктами. Користувачі часто змушені працювати в рамках заздалегідь визначеної логіки обробки новин, що ускладнює адаптацію системи до індивідуальних потреб або специфіки конкретної предметної області. Крім того, доступ до розширеного функціоналу часто є платним або обмеженим за кількістю запитів і джерел даних [18, 31].

Окрему групу рішень становлять системи, орієнтовані на моніторинг соціальних мереж. Такі системи забезпечують високу швидкість отримання повідомлень та можуть бути використані для виявлення трендів або реакцій користувачів на події. Водночас інформація, що надходить з соціальних платформ, характеризується високим рівнем шуму та потребує додаткової обробки і фільтрації, що не завжди реалізовано достатньо ефективно в існуючих рішеннях [19, 31].

Важливим аспектом при аналізі існуючих систем моніторингу новин є архітектурні особливості їх побудови. Значна частина рішень використовує синхронні підходи до обробки даних або ж обмежену асинхронність, що знижує ефективність роботи при різкому зростанні навантаження. У таких системах затримки на етапах збору, обробки або доставки інформації можуть призводити до втрати актуальності новин, що є критичним фактором для користувачів, орієнтованих на оперативне отримання інформації [13, 19, 24].

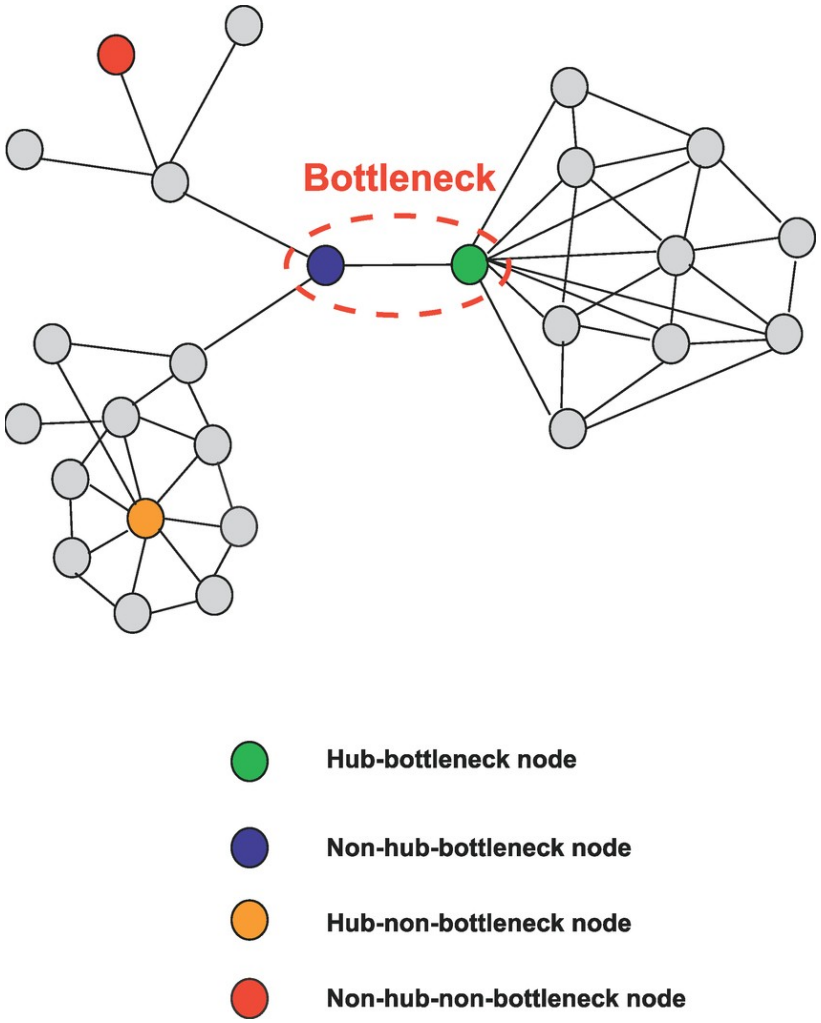

Рис. 1.7 – Ілюстрація виникнення вузького місця у централізованій системі обробки інформації [31]

Крім того, багато існуючих сервісів не підтримують повноцінної роботи з декількома каналами доставки інформації одночасно. Як правило, новини доступні через веб-інтерфейс або електронну пошту, тоді як інтеграція з сучасними засобами комунікації, такими як месенджери або платформи миттєвих повідомлень, реалізована частково або відсутня. Це обмежує можливості оперативного інформування користувачів та знижує зручність використання системи в умовах постійної інформаційної динаміки [6, 7, 8, 33, 34].

Аналіз існуючих рішень також показує, що механізми фільтрації та пріоритизації новин не завжди відповідають реальним потребам користувачів. У багатьох системах фільтрація виконується за обмеженим набором параметрів, що ускладнює відокремлення дійсно важливої інформації від другорядної. В умовах великих інформаційних потоків це може призводити як до перевантаження користувача надмірною кількістю повідомлень, так і до втрати критично важливих новин [19, 31].

Таким чином, проведений аналіз існуючих систем моніторингу новин дозволяє зробити висновок, що на сучасному етапі розвитку інформаційних технологій існує потреба у створенні гнучкого, масштабованого та асинхронного програмного рішення, здатного працювати з різноманітними джерелами новин і забезпечувати оперативну доставку інформації користувачам. Особливо актуальним є використання подійно-орієнтованої архітектури та асинхронних механізмів обміну повідомленнями, що дозволяють ефективно обробляти великі обсяги даних у режимі реального часу [12, 13, 14, 17, 17, 19].

Враховуючи виявлені недоліки існуючих рішень, доцільною є розробка власної системи моніторингу глобальних новин, яка базується на використанні асинхронних повідомлень та інтеграції з сучасними каналами комунікації, зокрема месенджерами Telegram та Discord. Реалізація такого підходу дозволить забезпечити своєчасне інформування користувачів, підвищити гнучкість налаштувань системи та створити універсальну платформу для подальшого розширення і адаптації під різні інформаційні потреби [6, 7, 8, 10, 12, 14, 17, 17, 28, 33, 34].

1.5. Висновки до розділу 1

У першому розділі кваліфікаційної роботи розглянуто теоретичні основи побудови систем реального часу для моніторингу глобальних новин. Проаналізовано особливості функціонування таких систем в умовах інтенсивних інформаційних потоків та визначено ключові вимоги до їх ефективності.

Окрему увагу приділено джерелам новинної інформації та моделям обміну даними. Встановлено, що використання асинхронної моделі взаємодії та подійно-орієнтованого підходу є найбільш доцільним для забезпечення своєчасної обробки та доставки інформації користувачам.

Також проведено аналіз існуючих систем моніторингу новин, за результатами якого визначено основні обмеження сучасних рішень, зокрема недостатню гнучкість, складність масштабування та затримки при високому навантаженні. Обґрунтовано доцільність розробки власної системи з використанням асинхронних повідомлень та інтеграції з сучасними каналами комунікації.

Отримані результати першого розділу слугують теоретичною основою для подальшого аналізу програмних технологій, проєктування архітектури та практичної реалізації системи реального часу для моніторингу глобальних новин, що буде розглянуто у наступних розділах роботи.

РОЗДІЛ 2. АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ МОНІТОРИНГУ ГЛОБАЛЬНИХ НОВИН

2.1. Аналіз технологій збору новинної інформації

Одним із ключових етапів у побудові системи моніторингу глобальних новин є організація ефективного механізму збору інформації з різноманітних джерел. Джерела новинної інформації можуть включати новинні веб-сайти, RSS-стрічки, соціальні мережі, а також відкриті інформаційні платформи. Кожне з цих джерел має власні особливості доступу, форматів даних та інтенсивності оновлення інформації [19, 31].

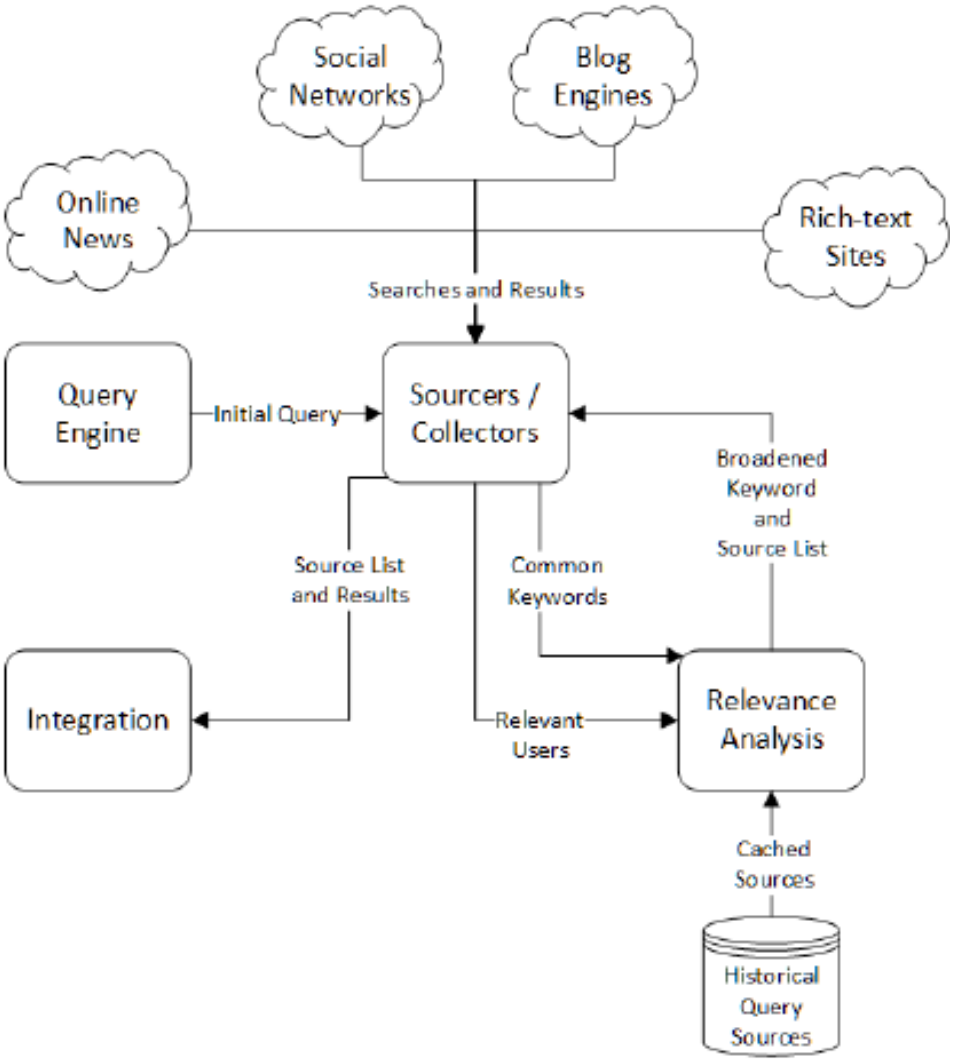

Рис. 2.1 – Узагальнена схема збору новинної інформації з різних джерел [11]

Для автоматизованого отримання новин широко використовуються RSS-стрічки, які забезпечують структурований формат передачі повідомлень та мінімізують обсяг оброблюваних даних. Використання RSS дозволяє зменшити навантаження на систему та спростити попередню обробку новинної інформації. Водночас не всі сучасні інформаційні ресурси підтримують RSS, що зумовлює необхідність застосування альтернативних методів збору даних [19, 31].

Іншим поширеним підходом є використання веб-скрапінгу, який забезпечує можливість отримання новин безпосередньо з веб-сторінок. Даний метод характеризується високою гнучкістю, однак потребує додаткових обчислювальних ресурсів та регулярного оновлення алгоритмів збору у разі зміни структури веб-ресурсів. Крім того, застосування веб-скрапінгу повинно враховувати політику доступу та обмеження, встановлені власниками сайтів [6, 31].

Значну роль у сучасних системах моніторингу новин відіграють соціальні мережі, які є джерелом оперативної інформації про події у реальному часі. Дані платформи дозволяють отримувати повідомлення з високою швидкістю, однак такі дані характеризуються підвищеним рівнем шуму та потребують застосування механізмів фільтрації й попереднього аналізу. Для збору інформації з соціальних мереж можуть використовуватися офіційні програмні інтерфейси або альтернативні підходи доступу до відкритих даних [7, 8, 19, 33, 34].

Таким чином, вибір технологій збору новинної інформації повинен базуватися на поєднанні різних методів доступу до даних, що дозволяє забезпечити повноту, актуальність та надійність отриманої інформації. Використання комбінованого підходу створює передумови для побудови ефективної системи моніторингу глобальних новин у режимі реального часу [6, 19, 31].

2.2. Аналіз технологій асинхронної обробки даних та обміну повідомленнями

Для систем моніторингу глобальних новин, що працюють у режимі реального часу, ключовим аспектом є ефективна обробка великої кількості подій та повідомлень, які надходять нерівномірно та з різною інтенсивністю. У таких умовах використання синхронних механізмів обміну даними може призводити до блокування процесів, збільшення затримок та зниження загальної продуктивності системи. Тому сучасні інформаційні системи широко застосовують асинхронні підходи до обробки даних і взаємодії між компонентами.

Асинхронна обробка даних ґрунтується на принципі незалежного виконання операцій, при якому відправлення повідомлення не потребує негайного отримання відповіді. Подібний підхід дозволяє компонентам системи працювати паралельно та зменшує взаємозалежність між ними. Завдяки цьому забезпечується більш ефективне використання обчислювальних ресурсів та підвищується стійкість системи до пікових навантажень.

Одним із основних механізмів реалізації асинхронної взаємодії є використання черг повідомлень. Черги дозволяють тимчасово зберігати повідомлення між етапами обробки та забезпечують буферизацію даних у випадках, коли швидкість надходження інформації перевищує можливості її обробки.

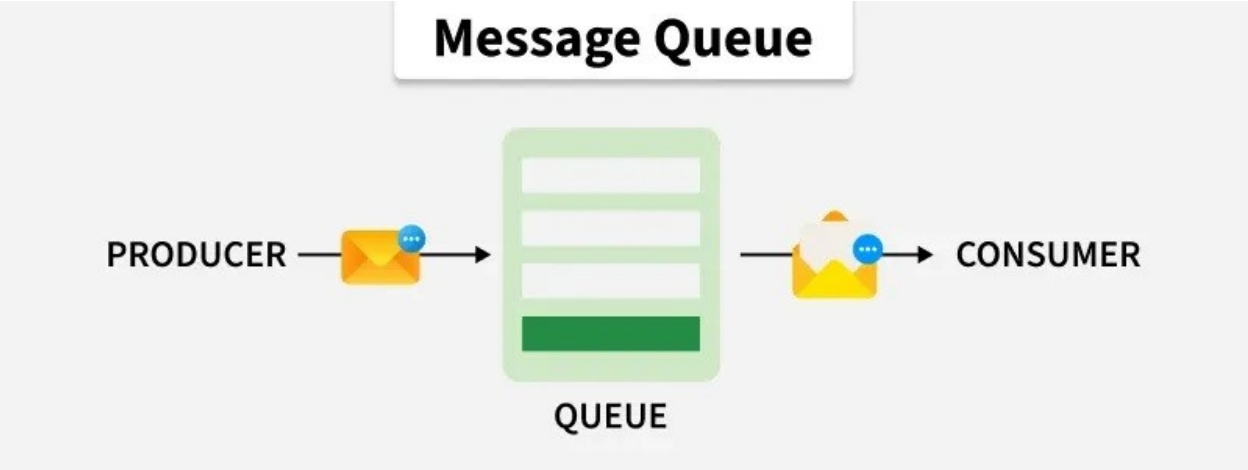

Рис. 2.2 – Асинхронна модель обробки повідомлень з використанням черги [18]

У системах моніторингу новин це особливо важливо під час резонансних подій, коли кількість повідомлень може різко зростати за короткий проміжок часу.

Архітектурно асинхронна обробка даних найчастіше реалізується у межах подійно-орієнтованих систем, де кожне нове повідомлення або зміна стану розглядається як окрема подія. Компоненти системи реагують на появу подій незалежно один від одного, отримуючи необхідні дані через механізми обміну повідомленнями. Такий підхід сприяє підвищенню масштабованості системи та спрощує її розширення шляхом додавання нових модулів без суттєвих змін в існуючій архітектурі.

Важливою перевагою асинхронних механізмів є підвищення надійності системи. У разі тимчасової недоступності окремого компонента повідомлення можуть бути збережені в черзі та оброблені після відновлення роботи відповідного модуля. Це зменшує ризик втрати даних та забезпечує безперервність функціонування системи моніторингу.

Разом з тим, використання асинхронної обробки даних вимагає врахування додаткових аспектів, зокрема управління чергами, забезпечення впорядкованості повідомлень та контролю часу їх обробки. У контексті систем моніторингу глобальних новин особливо важливим є досягнення балансу між швидкістю доставки інформації та стабільністю роботи системи в умовах високого навантаження.

Таким чином, аналіз технологій асинхронної обробки даних та обміну повідомленнями показує, що їх використання є обґрунтованим і доцільним для побудови систем реального часу з моніторингу новин. Застосування асинхронних механізмів дозволяє забезпечити високий рівень продуктивності, масштабованості та надійності системи, що є критично важливим для ефективного забезпечення користувачів актуальною інформацією.

2.3. Аналіз каналів доставки новинної інформації користувачам

Одним із важливих компонентів системи моніторингу глобальних новин є механізм доставки обробленої інформації кінцевим користувачам. Ефективність такого механізму визначається швидкістю надходження повідомлень, надійністю передачі даних та зручністю взаємодії користувача з системою. У сучасних умовах традиційні канали доставки інформації, такі як веб-інтерфейси або електронна пошта, не завжди здатні забезпечити необхідний рівень оперативності [13, 19].

Суттєву популярність у системах реального часу набули платформи миттєвих повідомлень, які дозволяють доставляти інформацію практично без затримок. Месенджери забезпечують постійний зв’язок із користувачем, підтримують сповіщення в реальному часі та дають можливість інтерактивної взаємодії з інформаційною системою. Це робить їх доцільними для використання у системах моніторингу новин, де актуальність повідомлень відіграє вирішальну роль [2, 3, 22, 27, 29].

Telegram є одним із найбільш поширених месенджерів, який активно використовується для інформаційних каналів та автоматизованих ботів. Його перевагами є швидка доставка повідомлень, підтримка публічних і приватних каналів, можливість групової взаємодії, а також зручні засоби для автоматизації розсилок. Це дозволяє ефективно поширювати новини серед широкого кола користувачів або окремих тематичних спільнот [1, 21, 33, 34, 35].

Discord, у свою чергу, орієнтований на інтерактивне спілкування у межах тематичних спільнот та серверів. Платформа підтримує текстові канали, миттєві сповіщення та розширені можливості керування доступом користувачів. Використання Discord як каналу доставки новинної інформації дозволяє поєднувати автоматизовані сповіщення з живим обговоренням подій, що підвищує залученість користувачів та ефективність поширення інформації [7, 8, 9, 10, 25, 28].

Використання месенджерів як каналів доставки новин органічно поєднується з асинхронною архітектурою системи. Повідомлення, що формуються в процесі аналізу новинних потоків, можуть передаватися користувачам незалежно від часу їх створення та оброблятися клієнтськими додатками без блокування інших компонентів системи. Такий підхід забезпечує масштабованість та стабільність доставки інформації навіть за значного зростання кількості користувачів [2, 3, 12, 14, 17, 17, 27, 29].

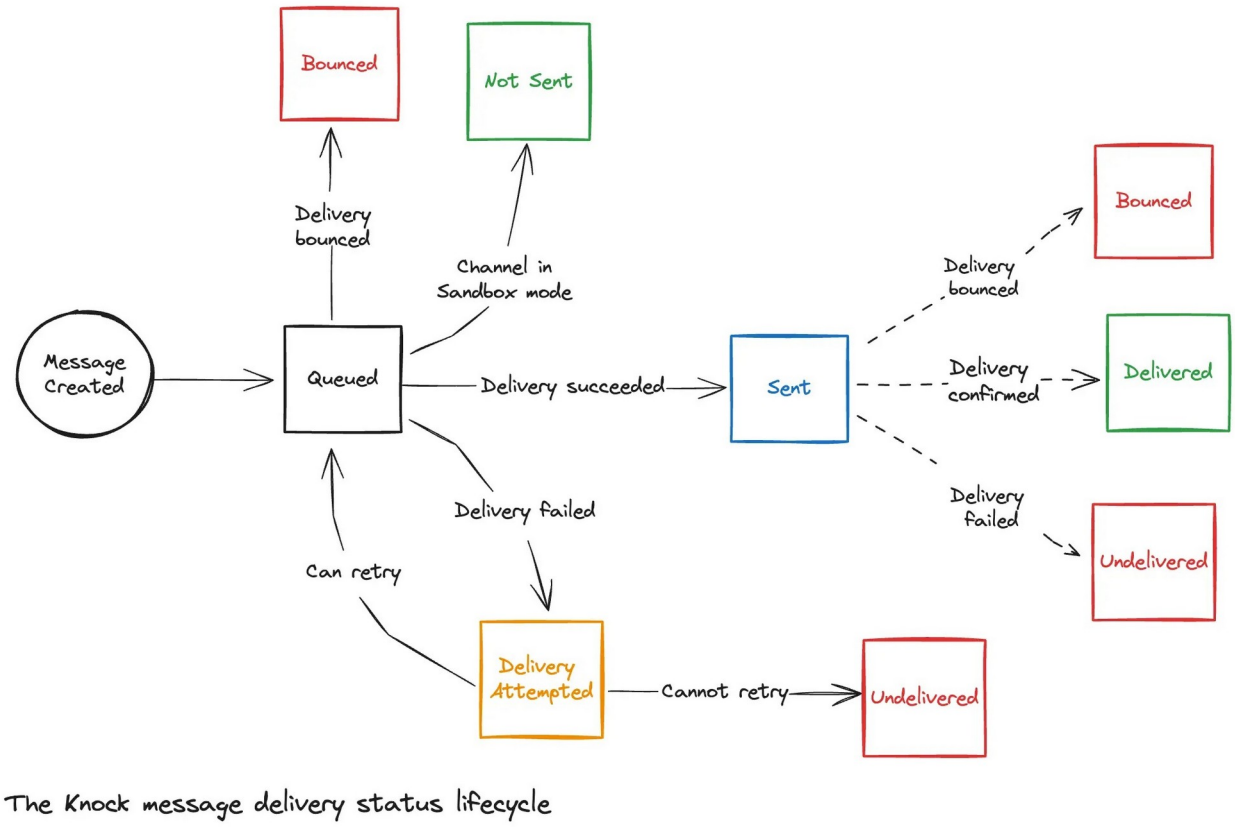

Рис. 2.3 – Життєвий цикл асинхронної доставки повідомлень у системі [18]

На рисунку 2.3 зображено узагальнений життєвий цикл асинхронної доставки повідомлень. Повідомлення після формування поміщається до черги, де очікує на обробку та подальшу передачу користувачу. У разі тимчасових помилок підтримується механізм повторної спроби доставки, що підвищує надійність функціонування системи. Такий підхід дозволяє забезпечити стабільну роботу сервісу доставки новин навіть за умов пікового навантаження або короткочасних збоїв у каналах зв’язку [12, 14, 17, 17, 30].

Таким чином, аналіз каналів доставки новинної інформації свідчить про доцільність використання месенджерів Telegram та Discord у системах моніторингу глобальних новин. Їх функціональні можливості дозволяють забезпечити високий рівень оперативності, зручності та інтерактивності, що відповідає вимогам сучасних систем реального часу [6, 7, 8, 10, 12, 13, 14, 17, 17, 28, 33, 34].

2.4. Висновки до розділу 2

У другому розділі кваліфікаційної роботи проведено аналіз сучасних інформаційних технологій та програмних підходів, що застосовуються для побудови систем моніторингу глобальних новин у режимі реального часу.

Розглянуто основні методи збору новинної інформації з різноманітних джерел, зокрема новинних веб-сайтів, RSS-стрічок та соціальних платформ. Встановлено, що поєднання різних підходів до отримання даних дозволяє забезпечити повноту та актуальність інформаційних потоків.

Проаналізовано асинхронні механізми обробки та передачі повідомлень, які є ключовими для роботи систем реального часу в умовах високого навантаження. Використання черг повідомлень і подійно-орієнтованих підходів дозволяє підвищити масштабованість, надійність та стабільність функціонування системи.

Окрему увагу приділено аналізу каналів доставки новинної інформації користувачам. Обґрунтовано доцільність використання месенджерів Telegram та Discord як основних каналів інформування, що забезпечують оперативність, інтерактивність та зручність взаємодії з користувачем.

Отримані результати другого розділу слугують підґрунтям для проєктування архітектури системи та практичної реалізації програмного рішення для моніторингу глобальних новин у режимі реального часу, що буде розглянуто у наступному розділі роботи.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ РЕАЛЬНОГО ЧАСУ ДЛЯ МОНІТОРИНГУ ГЛОБАЛЬНИХ НОВИН

3.1. Загальна архітектура системи моніторингу глобальних новин

Проєктована система моніторингу глобальних новин реалізується на основі модульної та подійно-орієнтованої архітектури, що забезпечує гнучкість, масштабованість та можливість подальшого розширення функціоналу. Такий підхід дозволяє розподілити систему на незалежні логічні компоненти, кожен з яких відповідає за виконання окремого набору функцій [6, 12, 14, 17, 17].

Загальна архітектура системи включає модулі збору новинної інформації, обробки та аналізу даних, зберігання результатів, а також модулі доставки повідомлень користувачам через месенджери Telegram та Discord. Взаємодія між модулями здійснюється з використанням асинхронних повідомлень, що дозволяє уникнути блокування процесів та забезпечує ефективну роботу системи за умов змінного навантаження [2, 3, 12, 14, 17, 17, 27, 29].

Модуль збору даних відповідає за отримання новин з різноманітних відкритих джерел, включаючи RSS-стрічки, новинні веб-сайти та соціальні платформи. Отримані дані передаються до модуля попередньої обробки, де виконується очищення, нормалізація та фільтрація інформації [19, 31].

Модуль обробки та аналізу даних забезпечує класифікацію новин, виявлення ключових тем та формування повідомлень для подальшої доставки. Для передачі повідомлень між компонентами використовується черга повідомлень, що дозволяє рівномірно розподіляти навантаження та забезпечити стійкість системи до тимчасових збоїв окремих модулів [12, 14, 17, 17, 31].

Завершальним етапом функціонування системи є доставка обробленої новинної інформації кінцевим користувачам. Для цього використовується окремий модуль інтеграції з месенджерами, який здійснює взаємодію з API Telegram та Discord у асинхронному режимі [7, 8, 9, 10, 25, 28, 33, 34].

Запропонована архітектура дозволяє забезпечити своєчасне інформування користувачів, високу продуктивність та надійність роботи системи, а також створює основу для подальшого розвитку та масштабування програмного рішення [6, 12, 13, 14, 17, 17, 18, 19, 31].

3.2. Опис функціональних модулів системи

Проєктована система моніторингу глобальних новин складається з набору взаємопов’язаних функціональних модулів, кожен з яких виконує визначену роль у процесі збору, обробки та доставки новинної інформації. Поділ системи на окремі модулі дозволяє забезпечити гнучкість архітектури, спростити супровід програмного забезпечення та підвищити надійність роботи системи [6, 12, 14, 17, 17].

Модуль збору даних (Data Collection Module)

Модуль збору даних призначений для автоматизованого отримання новинної інформації з різноманітних відкритих джерел. До таких джерел належать новинні веб-сайти, RSS-стрічки та соціальні інформаційні платформи. Основним завданням модуля є регулярне опитування джерел та формування вхідного потоку новин для подальшої обробки [19, 31].

Модуль реалізує механізми доступу до джерел із урахуванням їх специфіки, що дозволяє поєднувати різні формати даних у єдиний інформаційний потік. Асинхронний принцип роботи модуля дає змогу масштабувати процес збору даних без блокування інших компонентів системи [2, 3, 22, 27, 29].

Модуль попередньої обробки та фільтрації (Preprocessing & Filtering Module)

Після отримання даних новинна інформація передається до модуля попередньої обробки та фільтрації. Основною функцією даного модуля є очищення текстових даних від зайвих символів, нормалізація форматів та видалення дубльованих повідомлень [19, 31].

Крім того, у цьому модулі реалізуються базові механізми фільтрації, що дозволяють відокремлювати релевантні новини відповідно до заданих тем або ключових слів. Такий підхід сприяє зменшенню навантаження на подальші етапи обробки та підвищує загальну ефективність системи [6, 31].

Модуль асинхронної взаємодії та черги повідомлень (Message Queue Module)

Для забезпечення асинхронної взаємодії між основними компонентами системи використовується модуль черги повідомлень. Він відповідає за тимчасове зберігання повідомлень та передачу їх між модулями у неблокуючому режимі [12, 14, 17, 17].

Застосування черг дозволяє рівномірно розподіляти навантаження, забезпечувати повторну обробку повідомлень у разі помилок та гарантувати стабільну роботу системи в умовах високої інтенсивності інформаційних потоків [4, 12, 14, 17, 17].

Модуль аналізу новинної інформації (Analysis Module)

Модуль аналізу виконує обробку структурованих новинних даних з метою виявлення ключових тем, визначення важливості повідомлень та формування готових інформаційних повідомлень для користувача [6, 31].

У рамках даного модуля можуть реалізовуватися алгоритми тематичної класифікації, оцінювання значущості новин та групування подій. Результати аналізу передаються до модуля доставки для подальшого інформування користувачів [19, 31].

Модуль доставки повідомлень (Notification Service)

Завершальним елементом архітектури є модуль доставки повідомлень, який забезпечує взаємодію системи з платформами миттєвих повідомлень Telegram та Discord. Даний модуль відповідає за формування повідомлень у відповідному форматі та їх асинхронну відправку користувачам [7, 8, 9, 10, 25, 28, 33, 34].

Асинхронний підхід до доставки дозволяє підтримувати повторну спробу відправлення повідомлень у разі тимчасових збоїв та забезпечує високий рівень надійності інформування користувачів [2, 3, 27, 29, 30].

Завдяки такій модульній структурі система моніторингу глобальних новин є масштабованою, стійкою до навантажень та придатною для подальшого розширення функціональних можливостей [6, 12, 13, 14, 17, 17, 18, 19, 31].

3.3. Реалізація програмної системи моніторингу глобальних новин

Реалізація системи реального часу для моніторингу глобальних новин виконана з урахуванням результатів архітектурного проєктування та орієнтована на використання асинхронної моделі обробки подій. Програмна система побудована за модульним принципом, що дозволяє логічно відокремити процеси збору, обробки та доставки новинної інформації, а також забезпечити масштабованість і стійкість до навантажень [6, 12, 14, 17].

Основу реалізації становить подійно-орієнтований підхід, за якого обробка новинних повідомлень і їх доставка користувачам здійснюється у неблокуючому режимі. Це дає можливість системі паралельно обробляти значні обсяги інформації з різних джерел та забезпечувати своєчасне інформування користувачів [12, 13, 14, 17].

Збір новинної інформації реалізується у фоновому режимі, після чого отримані дані проходять етап попередньої обробки, нормалізації та формування структурованих повідомлень. Для передачі повідомлень між основними компонентами системи використовується внутрішній механізм черг, що забезпечує асинхронну взаємодію та дозволяє уникнути блокування виконання програми у разі пікових навантажень [2, 3, 12, 14, 17, 27, 29].

Доставка новин користувачам здійснюється за допомогою інтеграції з платформами миттєвих повідомлень Telegram та Discord. Реалізація сервісу доставки побудована на основі асинхронного циклу обробки подій, що дозволяє незалежно обробляти вхідні повідомлення, запити користувачів та процеси відправлення новин. У разі виникнення тимчасових помилок підтримується повторна спроба доставки, що підвищує надійність роботи системи [7, 8, 9, 10, 25, 28, 30, 33, 34].

Для підтвердження практичної реалізації асинхронної архітектури наведено узагальнений фрагмент програмного коду, який ілюструє запуск сервісу доставки повідомлень та ініціалізацію основних компонентів системи [2, 3, 27, 29].

```
import asyncio
async def start_system():
    await init_message_clients()
    await run_event_loop()
if __name__ == "__main__":
    asyncio.run(start_system())
```

Наведений фрагмент програмного коду демонструє використання асинхронної моделі виконання, яка дозволяє системі реалізовувати неблокуючу обробку подій та забезпечувати стабільну доставку новинної інформації незалежно від інтенсивності інформаційних потоків. Такий підхід є ключовим для систем реального часу, орієнтованих на оперативне інформування користувачів [2, 3, 13, 27, 29].

Повна реалізація програмної системи, включаючи логіку роботи ботів, обробку повідомлень та інтеграцію з зовнішніми сервісами, наведена у додатку до кваліфікаційної роботи.

3.4. Алгоритм функціонування системи

Алгоритм функціонування системи реального часу для моніторингу глобальних новин побудований на подійно-орієнтованій та асинхронній моделі обробки даних. Основною ідеєю є поетапна обробка новинних повідомлень із використанням внутрішньої черги подій, що дозволяє забезпечити стабільну роботу системи незалежно від інтенсивності надходження інформаційних потоків [12, 13, 14, 17].

Функціонування системи складається з послідовності логічних етапів, які виконуються асинхронно та не блокують виконання інших компонентів [2, 3, 27, 29].

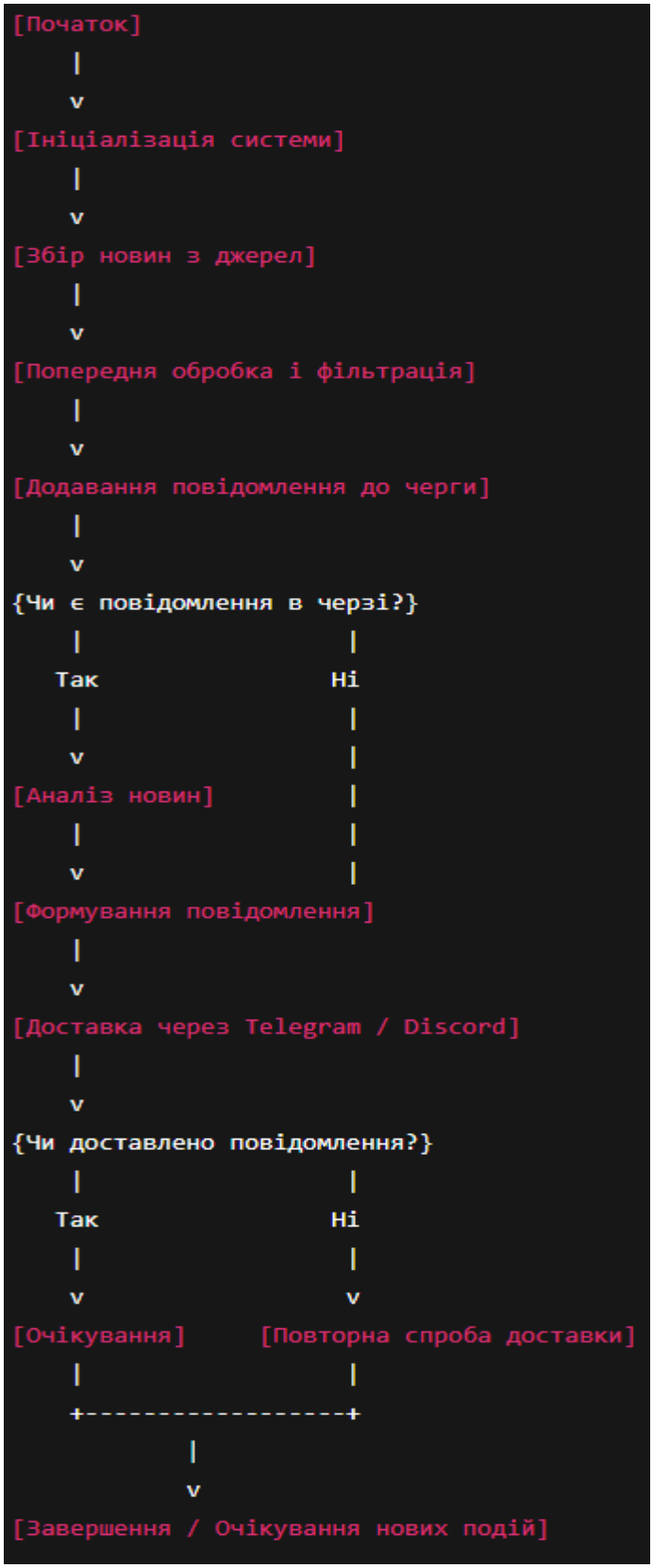

Рис. 3.1 – Алгоритм функціонування системи моніторингу глобальних новин

На рисунку 3.1 представлено алгоритм функціонування системи моніторингу глобальних новин. Алгоритм ґрунтується на подійно-орієнтованій та асинхронній моделі обробки інформації, що дозволяє забезпечити безперервну роботу системи та своєчасну доставку новин користувачам навіть за умов високого навантаження [12, 13, 14, 17].

Основні етапи алгоритму роботи системи

На першому етапі ініціюється процес збору новинної інформації з відкритих джерел. Дані надходять у систему у фоновому режимі та передаються до модуля попередньої обробки без затримки інших процесів [19, 31].

Другий етап передбачає очищення та нормалізацію отриманих повідомлень. На цьому етапі виконується видалення дубльованих новин, усунення службових символів та приведення даних до уніфікованого формату [19, 31].

Після завершення попередньої обробки підготовлені повідомлення поміщаються до внутрішньої черги повідомлень, яка використовується для асинхронної передачі даних між модулями системи. Такий підхід дозволяє згладжувати пікові навантаження та підтримувати стабільну швидкість обробки [4, 12, 14, 17].

Наступним етапом є аналіз новинної інформації та формування повідомлень для користувача. За результатами аналізу створюється структуроване повідомлення, яке передається до модуля доставки [6, 31].

Завершальним етапом є асинхронна доставка новинних повідомлень кінцевим користувачам через інтегровані платформи миттєвих повідомлень. У разі виникнення тимчасових помилок система виконує повторну спробу доставки повідомлень [7, 8, 9, 10, 25, 28, 30, 33, 34].

Таблиця 3.1 – Послідовність етапів алгоритму функціонування системи

№ етапу	Назва етапу	Опис
1	Збір даних	Отримання новин з відкритих інформаційних джерел
2	Попередня обробка	Очищення, нормалізація та фільтрація повідомлень
3	Черга повідомлень	Асинхронна передача новин між модулями системи
4	Аналіз інформації	Формування структурованих повідомлень
5	Доставка повідомлень	Асинхронна відправка новин користувачам

Узагальнений алгоритм у вигляді псевдокоду

Для наочності роботи системи нижче наведено узагальнений псевдокод, що ілюструє послідовність взаємодії основних компонентів системи в асинхронному режимі [2, 3, 27, 29].

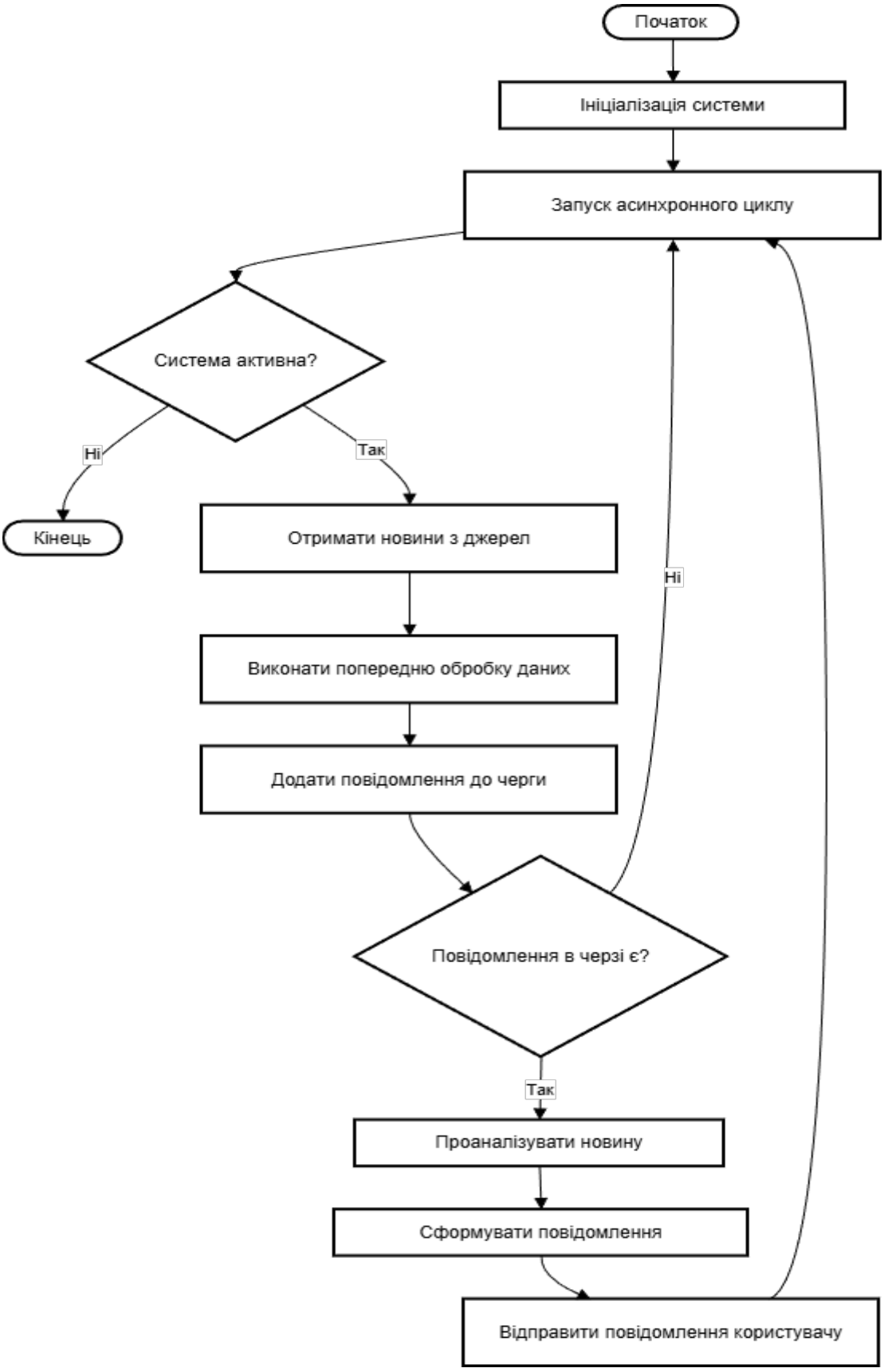

Рис. 3.2 – Алгоритм функціонування системи моніторингу глобальних новин

Фрагмент програмного коду асинхронної обробки подій

Для підтвердження практичної реалізації алгоритму наведено узагальнений фрагмент програмного коду, що демонструє асинхронну обробку подій у системі [2, 3, 27, 29].

```
async def process_events():
    while True:
        message = await message_queue.get()
        processed = await analyze_message(message)
        await send_notification(processed)
```

Наведений фрагмент коду ілюструє неблокуючу обробку повідомлень та асинхронну доставку інформації користувачам, що є ключовою характеристикою систем реального часу [2, 3, 13, 27, 29].

Запропонований алгоритм функціонування системи забезпечує ефективну організацію процесів збору, обробки та доставки новинної інформації. Поєднання асинхронних механізмів, черг повідомлень і модульної архітектури дозволяє забезпечити надійну роботу системи та адаптацію до змінних умов інформаційного середовища [6, 12, 13, 14, 17, 17, 18, 19, 31].

3.5. Висновки до розділу 3

У третьому розділі кваліфікаційної роботи здійснено проєктування та опис практичної реалізації системи реального часу для моніторингу глобальних новин. На основі визначених вимог і результатів попереднього аналізу побудовано модульну архітектуру системи, що базується на подійно-орієнтованому та асинхронному підходах до обробки інформації.

Розроблено загальну структуру системи та визначено призначення основних програмних компонентів, що забезпечують процеси збору, попередньої обробки, аналізу та доставки новинної інформації. Реалізація асинхронної взаємодії між модулями дозволяє ефективно працювати з великими обсягами даних та забезпечувати стабільність функціонування системи в умовах змінного навантаження.

Описано алгоритм функціонування системи, який відображає послідовність обробки новинних подій та механізм асинхронної доставки повідомлень кінцевим користувачам через платформи миттєвих повідомлень. Запропонований алгоритм відповідає вимогам систем реального часу та забезпечує своєчасне інформування користувачів.

Результати третього розділу створюють практичне підґрунтя для проведення дослідження функціонування розробленої системи, тестування її можливостей та оцінювання ефективності, що буде розглянуто у наступному розділі кваліфікаційної роботи.

РОЗДІЛ 4. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ ГЛОБАЛЬНИХ НОВИН У РЕЖИМІ РЕАЛЬНОГО ЧАСУ

4.1. Загальна архітектура та модульна структура системи

Після проведення аналізу предметної області та існуючих підходів до побудови систем реального часу було розроблено програмну систему моніторингу глобальних новин, орієнтовану на асинхронну обробку інформаційних потоків і оперативну доставку повідомлень користувачам. Система реалізована у вигляді багатомодульного програмного рішення, в якому кожен компонент виконує чітко визначену функцію та взаємодіє з іншими модулями за допомогою визначених інтерфейсів [6, 12, 13, 14, 17].

Архітектура системи побудована за модульним принципом, що дозволяє забезпечити гнучкість, масштабованість та зручність подальшого розширення функціональних можливостей. Загальна логіка роботи системи полягає у безперервному отриманні інформації з зовнішніх джерел, її попередній обробці, збереженні службових даних у базі даних та подальшій асинхронній доставці новин кінцевим користувачам через сучасні канали комунікації [6, 12, 14, 17, 17, 18, 19, 31].

Загальна структура системи

До складу розробленої системи входять наступні основні підсистеми:

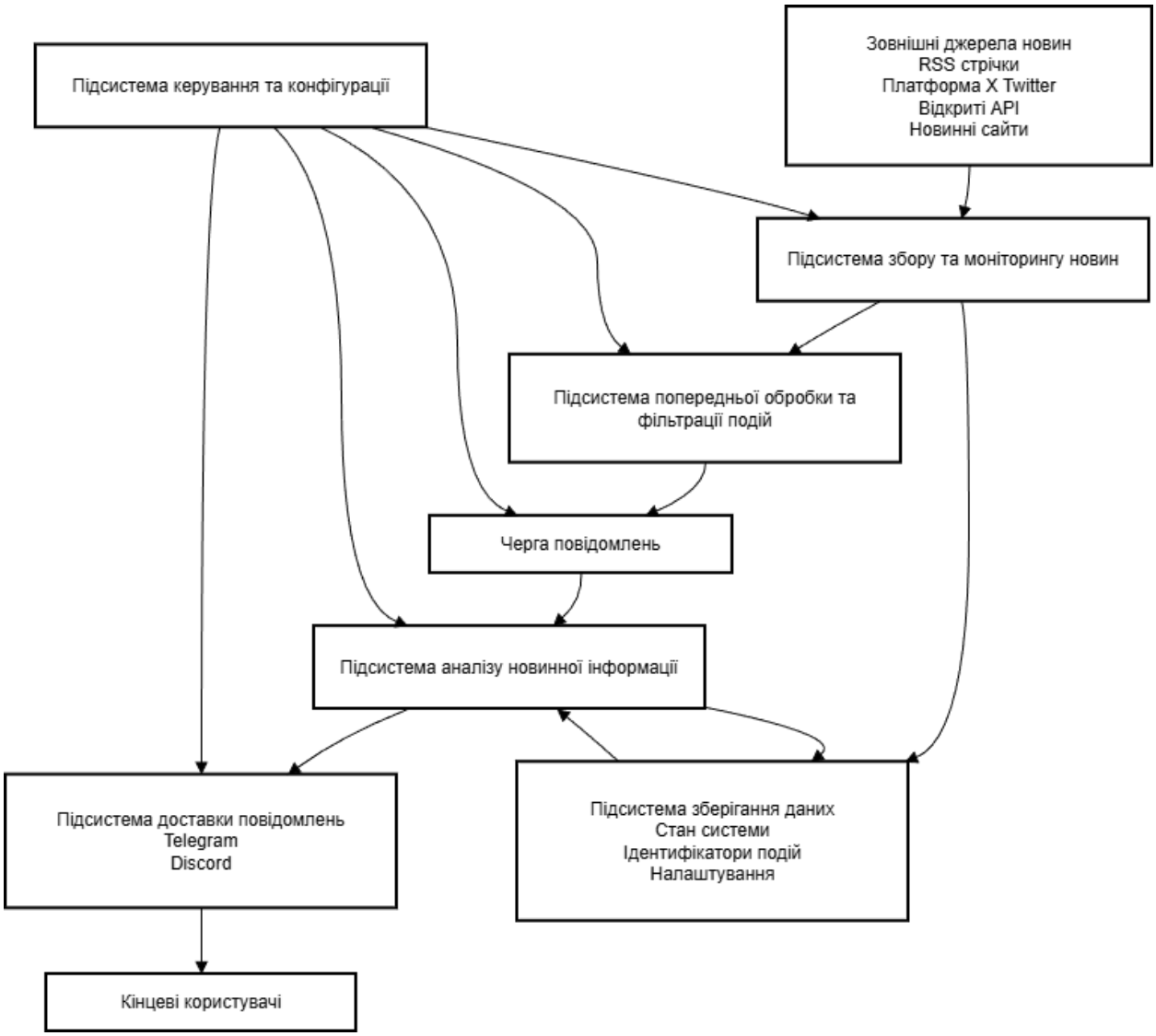

Рис. 4.1 – Загальна структурна схема системи моніторингу глобальних новин

Взаємодія між підсистемами здійснюється у подійно-орієнтованому режимі, що дозволяє обробляти новини незалежно від швидкості їх надходження та мінімізує затримки в роботі системи [12, 14, 17].

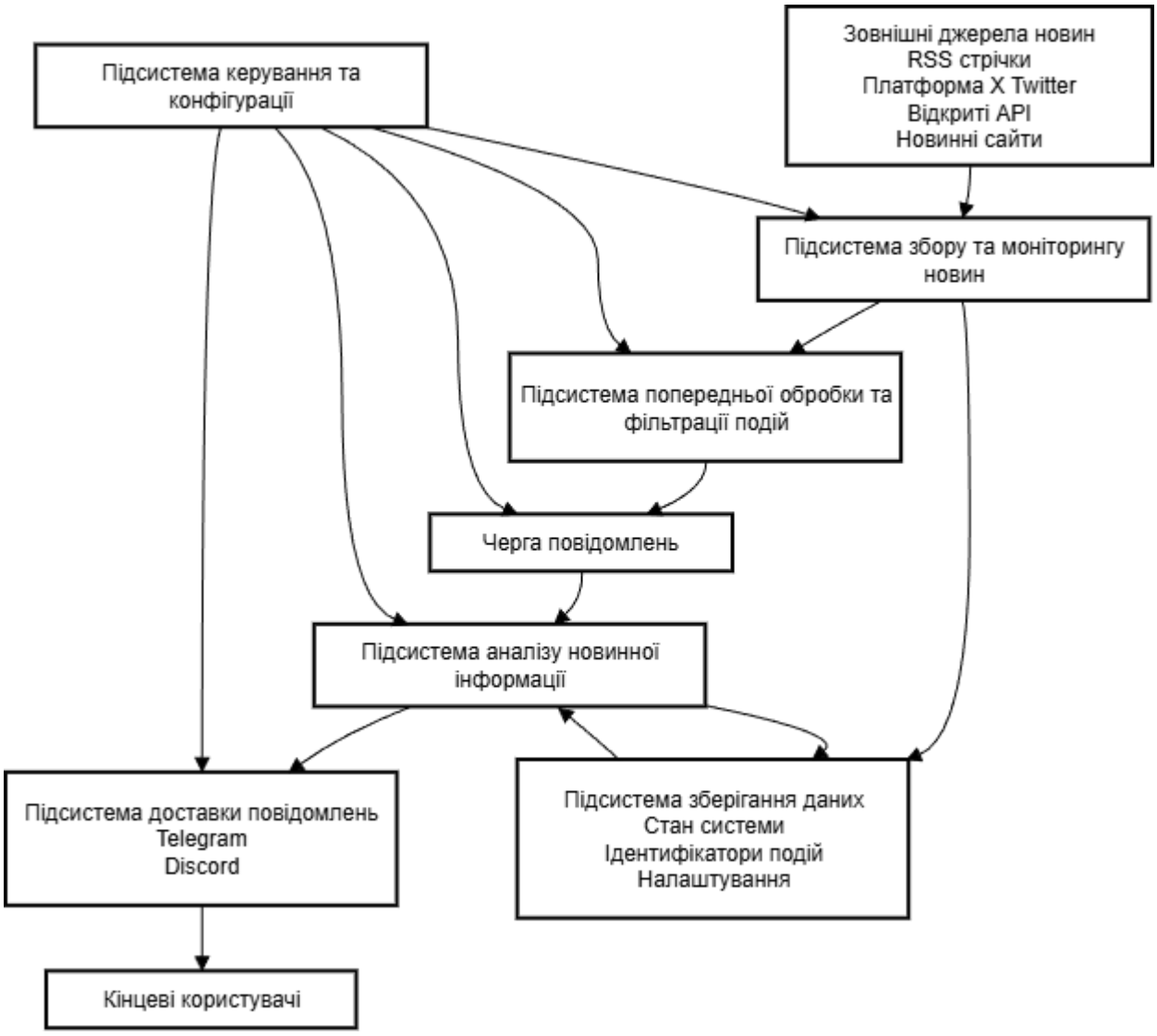

Рис. 4.2 – Загальна структурна схема системи моніторингу глобальних новин

Підсистема збору та моніторингу новин

Підсистема збору інформації відповідає за періодичне або безперервне отримання новин з зовнішніх джерел. У розробленій системі джерелами даних виступають відкриті інформаційні платформи та соціальні мережі, зокрема Twitter/X, де новинні події з’являються практично в режимі реального часу [19, 31].

Основним завданням модуля моніторингу є:

- відстеження активності заданих користувачів або джерел;
- отримання нових повідомлень (подій);
- визначення, чи є подія новою, або вона вже була оброблена раніше;

- передача нових подій до наступних компонентів системи.

Модуль реалізовано із використанням асинхронної моделі виконання, що дозволяє ефективно працювати з великою кількістю джерел без блокування основного процесу виконання програми [2, 3, 22, 27, 29].

Підсистема зберігання даних

Важливим елементом архітектури системи є база даних, яка використовується для збереження службової інформації та керування станом системи. На відміну від традиційних новинних агрегаторів, у розробленій системі база даних не призначена для довготривалого зберігання повного тексту новин. Її основною функцією є фіксація поточного стану моніторингу [6, 16].

У базі даних зберігається наступна інформація:

- список користувачів або джерел, за якими ведеться моніторинг;
- ідентифікатори останніх оброблених новин або повідомлень;
- налаштування системи та параметри роботи;
- службові дані, необхідні для відновлення роботи після перезапуску системи.

Використання бази даних дозволяє запобігати дублюванню повідомлень, оскільки кожна нова подія перевіряється на унікальність перед її обробкою та доставкою користувачам [6, 16].

Підсистема доставки повідомлень

Підсистема доставки відповідає за передачу оброблених новин кінцевим користувачам. У межах даного проєкту основним каналом взаємодії з користувачами обрано платформу Discord. Це зумовлено її підтримкою миттєвих сповіщень, можливістю інтерактивної взаємодії та гнучкими налаштуваннями доступу [7, 8, 9, 10, 25, 28].

Крім того, архітектура системи передбачає можливість підключення альтернативних каналів доставки, таких як Telegram, без суттєвих змін у логіці роботи інших компонентів. Це досягається завдяки чіткому розділенню обов’язків між модулями та використанню уніфікованих інтерфейсів передачі повідомлень [1, 21, 33, 34, 35].

Подійно-орієнтована та асинхронна модель роботи

Уся система побудована за принципами подійно-орієнтованої архітектури. Кожна новина або інформаційне повідомлення розглядається як окрема подія, що обробляється незалежно від інших подій. Асинхронний підхід дозволяє:

- зменшити затримки при обробці великих обсягів даних;
- забезпечити стабільну роботу системи при пікових навантаженнях;
- ефективно використовувати ресурси сервера [12, 13, 14, 17].

Завдяки цьому система здатна масштабуватися відповідно до зростання кількості джерел новин або користувачів без потреби у суттєвій зміні архітектури [6, 18, 19, 31].

Модульність та можливість розширення

Модульна структура системи забезпечує зручність її підтримки та подальшого розвитку. Кожен модуль може бути доопрацьований або замінений без впливу на інші компоненти. Це створює умови для:

- підключення нових джерел новин;
- розширення функціоналу фільтрації та аналізу інформації;
- додавання нових каналів доставки повідомлень;
- інтеграції з іншими інформаційними системами [6, 18, 19, 31].

Таким чином, розроблена архітектура є універсальною основою для побудови системи моніторингу глобальних новин у режимі реального часу, яка поєднує асинхронну обробку, збереження стану у базі даних та оперативну взаємодію з користувачами [6, 12, 13, 14, 16, 17, 18, 19, 31].

4.2. Організація зберігання даних та керування станом системи

Однією з ключових задач при розробці системи моніторингу глобальних новин у режимі реального часу є забезпечення коректного зберігання службових даних та керування поточним станом системи. Без використання механізмів збереження даних неможливо гарантувати стабільну роботу системи, уникнення дублювання повідомлень та відновлення функціонування після перезапуску [6, 13].

У розробленій системі база даних використовується не як сховище повного новинного контенту, а як інструмент підтримки логіки моніторингу та контролю оброблених подій. Такий підхід дозволяє мінімізувати обсяг збереженої інформації та підвищити продуктивність системи [6, 19].

Призначення бази даних у системі

База даних виконує наступні основні функції:

- збереження списку джерел або користувачів, за якими здійснюється моніторинг;
- фіксація ідентифікаторів останніх оброблених новин;
- збереження інформації про вже доставлені повідомлення;
- підтримка унікальності подій;
- забезпечення можливості відновлення стану системи після зупинки або перезапуску [6, 16].

Завдяки цьому система здатна працювати безперервно та коректно обробляти лише нові події, які ще не були передані користувачам [6].

Структура збережених даних

Для реалізації зазначених функцій у базі даних зберігається кілька логічних груп даних. Узагальнену структуру можна подати у вигляді таблиці.

Таблиця 4.1 – Основні структури даних системи

Назва об’єкта	**Опис**
TrackedUsers	Список користувачів або джерел новин, за якими здійснюється моніторинг
LastPostId	Ідентифікатор останнього обробленого повідомлення для кожного джерела
PublishedPosts	Дані про повідомлення, які вже були надіслані в Discord
Settings	Службові параметри конфігурації системи
Timestamps	Час останнього оновлення або перевірки джерела

Примітка: конкретна реалізація структури може виконуватися у вигляді файлу, SQL або NoSQL-сховища, залежно від обраних засобів реалізації [16].

Механізм запобігання дублюванню повідомлень

Однією з критично важливих задач системи моніторингу новин є уникнення повторного надсилання одного й того ж повідомлення. У розробленій системі ця проблема вирішується шляхом фіксації ідентифікаторів останніх оброблених подій у базі даних [6, 16].

Алгоритм роботи виглядає наступним чином:

1. система отримує нову подію з джерела інформації;
2. ідентифікатор події перевіряється у базі даних;
3. якщо подія вже була оброблена, вона ігнорується;
4. у разі виявлення нової події її дані передаються до модуля доставки;
5. ідентифікатор події зберігається як актуальний стан джерела.

Такий підхід дозволяє уникати дублювання повідомлень навіть у випадках повторної перевірки джерел або перезапуску програми [6].

Збереження стану між сесіями роботи

Використання бази даних дозволяє зберігати стан системи між окремими сесіями її роботи. Навіть після аварійного завершення або перезапуску система може продовжити роботу без втрати контексту моніторингу [6, 13].

Після запуску система:

- зчитує актуальний список джерел;
- отримує останні збережені ідентифікатори повідомлень;
- відновлює черги подій та внутрішні стани;
- продовжує моніторинг з урахуванням попередніх результатів.

Це є особливо важливим для систем реального часу, які працюють у безперервному режимі та мають обробляти великі обсяги інформації [6, 16].

Взаємодія бази даних з іншими модулями

База даних тісно інтегрована з іншими компонентами системи, зокрема з модулем збору новин та модулем доставки повідомлень. Усі операції зчитування та запису виконуються у фоновому режимі та не блокують основні процеси обробки [2, 3, 22, 27, 29].

Асинхронна модель доступу до даних дозволяє:

- зменшити затримки при роботі з джерелами новин;
- забезпечити стійкість системи при високому навантаженні;
- мінімізувати ризик втрати даних [2, 3, 27, 29, 30].

Переваги обраного підходу

Використання бази даних як механізму керування станом системи має такі переваги:

- простота реалізації та підтримки;
- висока надійність роботи;

- відсутність дублювання новин;
- швидке відновлення після збоїв;
- можливість масштабування системи [6, 18, 19, 31].

Таким чином, підсистема зберігання даних є невід’ємною частиною розробленої системи моніторингу глобальних новин і забезпечує коректну, стабільну та ефективну роботу програмного рішення у режимі реального часу [6, 12, 13, 14, 16, 17, 18, 19, 31].

Фрагмент реалізації перевірки нових подій на основі бази даних

Для реалізації механізму уникнення дублювання новинної інформації в системі використовується база даних, у якій зберігається час останньої обробленої події для кожного користувача. На основі цих даних здійснюється фільтрація нових повідомлень. Відповідна логіка реалізована у функції get_tweets, наведеної у рисунку 4.3. [16].

annel | get_tweets.py | sync_db.py | checker.py | presence_updater.py | src\utils.py | classes.py | list_users.py | sync

```python
import os
from typing import Optional

from src.db_function.readonly_db import connect_readonly
from tweety.types import Tweet

from src.notification.date_comparator import date_comparator


async def get_tweets(tweets: list[Tweet], username: str) -> Optional[list[Tweet]]:  3 usages (1 dynamic)

    async with connect_readonly(os.path.join(os.getenv('DATA_PATH'), 'tracked_accounts.db')) as db:
        async with db.execute('SELECT lastest_tweet FROM user WHERE username = ?', (username,)) as cursor:
            row = await cursor.fetchone()
            last_tweet_at = row[0]

    tweets = [tweet for tweet in tweets if tweet.author.username == username and date_comparator(tweet.created_on, last_tweet_at) == 1]

    if tweets != []:
        return sorted(tweets, key=lambda x: x.created_on)
    else:
        return None

```

Рис. 4.3 – Фрагмент функції `get_tweets` для вибору нових твітів

У наведеному фрагменті коду реалізовано логіку отримання лише нових повідомлень від відстежуваного користувача.

Спочатку виконується підключення до бази даних `tracked_accounts.db` у режимі лише читання. Із таблиці `user` за іменем користувача `username` вибирається значення поля `lastest_tweet`, яке зберігає мітку часу останнього обробленого повідомлення.

Після цього вхідний список твітів фільтрується за двома умовами:

- твіт має належати потрібному користувачу (`tweet.author.username == username`);
- час створення твіта `tweet.created_on` повинен бути пізнішим за останню збережену подію (`date_comparator( tweet.created_on, last_tweet_at ) == 1`).

Таким чином, у список `tweets` потрапляють лише ті події, які є новими відносно стану, зафіксованого в базі даних. Якщо список нових твітів не порожній, вони додатково сортуються за часом створення, що дозволяє надалі коректно формувати порядок доставки повідомлень користувачам. Якщо нових подій немає, функція повертає `None`.

Реалізація такої перевірки на основі даних із таблиці `user` дозволяє уникати повторної обробки вже доставлених новин, зменшує навантаження на систему та гарантує, що кінцеві користувачі отримують лише актуальні оновлення [6, 16].

4.3. Реалізація модуля моніторингу новин (бекграунд-сервіс)

Модуль моніторингу глобальних новин у розробленій системі реалізований у вигляді бекграунд-сервісу, який забезпечує безперервне відстеження зовнішніх джерел інформації та своєчасне виявлення нових подій. На відміну від інтерфейсних або керуючих компонентів системи, даний модуль функціонує у фоновому режимі та не потребує активної взаємодії з користувачем під час виконання [12, 13, 14, 17].

Основним призначенням модуля моніторингу є автоматизований збір новин із заданих джерел, аналіз отриманих даних та передача тільки нових, раніше не оброблених повідомлень до підсистеми доставки. Такий підхід дозволяє забезпечити актуальність інформації, уникнути дублювання повідомлень та знизити навантаження на користувачів [6, 31].

Загальна логіка роботи бекграунд-сервісу

Робота модуля моніторингу організована у вигляді нескінченного асинхронного циклу, який повторюється з певним інтервалом. Протягом кожної ітерації виконуються наступні дії:

1. отримання з бази даних списку джерел новин (користувачів або акаунтів), за якими здійснюється моніторинг;
2. звернення до зовнішнього сервісу (Twitter/X) для отримання актуальних повідомлень;
3. порівняння отриманих даних із поточним станом, збереженим у базі даних;
4. відбір лише нових повідомлень, які не були оброблені раніше;
5. збереження інформації про нові події та передача їх у чергу сповіщень;
6. очікування до наступного циклу перевірки.

Завдяки асинхронній моделі виконання кожен із зазначених етапів не блокує роботу системи в цілому, що є критично важливим для систем реального часу [2, 3, 22, 27, 29].

Асинхронна модель виконання

Для реалізації бекграунд-сервісу використовується асинхронний підхід із застосуванням механізмів async/await. Це дозволяє системі виконувати операції введення-виведення (доступ до мережі, бази даних, файлової системи) у неблокуючому режимі та ефективно використовувати ресурси обчислювального середовища [2, 3, 27, 29].

Асинхронний підхід є особливо актуальним у контексті моніторингу глобальних новин, оскільки:

- час відповіді зовнішніх сервісів може бути непередбачуваним;
- кількість джерел новин може змінюватися динамічно;
- система повинна залишатися стабільною навіть при пікових навантаженнях [12, 13, 14, 17].

Реалізація циклу моніторингу

Основна робота бекграунд-сервісу виконується у вигляді нескінченного асинхронного циклу. Узагальнений фрагмент програмної реалізації такого циклу наведено у цьому коді [2, 3, 27, 29].

```
class AccountTracker:
  def __init__(self, bot: commands.Bot):
    self.bot = bot
    self.accounts_data = get_accounts()
    self.db_path = os.path.join(os.getenv('DATA_PATH'), 'tracked_accounts.db')
    # Буфер твітів для кожного клієнта
                self.tweets = {account_name: [] for account_name in self.accounts_data.keys()}
    # Час останнього логування монітора задач
    self.tasksMonitorLogAt = datetime.now(timezone.utc) - timedelta(
      hours=configs['tasks_monitor_log_period']
    )
    # Запуск асинхронного налаштування задач
    bot.loop.create_task(self.setup_tasks())
```

```
    async def setup_tasks(self):
        tasks = []
        async def authenticate_and_track(account_name, account_token):
            try:
                app = Twitter(account_name)
                max_attempts = configs['auth_max_attempts']
                for attempt in range(max_attempts):
                    try:
                        await app.load_auth_token(account_token)
                        break
                    except Exception as e:
                        if attempt < max_attempts - 1:
                            await asyncio.sleep(2)
                        else:
                            raise e
                # запуск окремого оновлювача твітів для клієнта
                self.bot.loop.create_task(
                    self.tweetsUpdater(app)
                ).set_name(f"TweetsUpdater_{account_name}")
                log.info(f"  Account {account_name} authenticated and tracking started")
            except Exception as e:
                log.error(f"  Failed to authenticate {account_name}: {e}")
        for account_name, token in self.accounts_data.items():
            tasks.append(asyncio.create_task(authenticate_and_track(account_name, token)))
        await asyncio.gather(*tasks)
        async with connect_readonly(self.db_path) as db:
```

```
            async with db.execute(
                'SELECT username, client_used FROM user WHERE enabled = 1'
            ) as cursor:
                usernames_and_clients = {row[0]: row[1] async for row in cursor}
        for username, client_used in usernames_and_clients.items():
            self.bot.loop.create_task(
                self.notification(username, client_used)
            ).set_name(username)
        self.bot.loop.create_task(
            self.tasksMonitor(usernames_and_clients)
        ).set_name('TasksMonitor')
```

У наведеному фрагменті реалізовано ініціалізацію модуля моніторингу та запуск асинхронних задач.

У конструкторі класу створюється буфер твітів для кожного клієнта, визначається шлях до бази даних та запускається корутина `setup_tasks`.

У методі `setup_tasks` виконується:

- асинхронна автентифікація всіх облікових записів Twitter;
- запуск окремої задачі `tweetsUpdater` для кожного клієнта;
- створення задач `notification` для кожного відстежуваного користувача;
- запуск монітора задач `tasksMonitor`, який стежить за станом фонових процесів [2, 3, 16, 27, 29].

Фрагмент реалізації моніторингу новин та відправки сповіщень:

```
async def notification(self, username: str, client_used: str):
    while True:
        await asyncio.sleep(configs['tweets_check_period'])
        # Отримуємо лише нові твіти з буфера для цього користувача
        lastest_tweets = await get_tweets(self.tweets[client_used], username)
```

```
            if lastest_tweets is None:
                continue
            async with aiosqlite.connect(self.db_path) as db:
                db.row_factory = aiosqlite.Row
                async with db.cursor() as cursor:
                    await cursor.execute(
                        'SELECT * FROM user WHERE username = ?',
                        (username,)
                    )
                    user = await cursor.fetchone()
                    # Оновлюємо мітку часу останнього твіта в БД
                    async with lock:
                        await cursor.execute(
                            'UPDATE user SET lastest_tweet = ? WHERE username = ?',
                            (str(lastest_tweets[-1].created_on), username)
                        )
                        await db.commit()
                    # Надсилання сповіщень у відповідні Discord-канали
                    for tweet in lastest_tweets:
                        log.info(f'find a new tweet from {username}')
                        await cursor.execute(
                                'SELECT * FROM notification WHERE user_id = ? AND
enabled = 1',
                            (user['id'],)
                        )
                        notifications = await cursor.fetchall()
                        for data in notifications:
                            channel = self.bot.get_channel(int(data['channel_id']))
                                    if channel is not None and is_match_type(tweet,
data['enable_type']) \
```

```
                    and is_match_media_type(tweet, data['enable_media_type']):
                    try:
                        mention = (
                            f"{channel.guild.get_role(int(data['role_id'])).mention} "
                            if data['role_id'] else ''
                        )
                        author, action = tweet.author.name, get_action(tweet)
                        url = re.sub('twitter', DOMAIN_NAME, tweet.url) \
                            if EMBED_TYPE == 'fx_twitter' else tweet.url
                        msg = data['customized_msg'] if data['customized_msg'] \
                              else configs['default_message']
                        msg = msg.format(
                            mention=mention,
                            author=author,
                            action=action,
                            url=url
                        )
                        if EMBED_TYPE != 'built_in':
                            await send_with_throttle(channel.send, msg)
                        else:
                                                img = discord.File('images/twitter.png',
filename='twitter.png')
                            embeds = await gen_embed(tweet)
                                await send_with_throttle(channel.send, None, file=img,
embeds=embeds)
                    except Exception as e:
                        if not isinstance(e, discord.errors.Forbidden):
                            log.error(
                                f'an error occurred at {channel.mention} '
                                f'while sending notification: {e}'
```

)

Даний фрагмент коду реалізує основний цикл моніторингу новин для конкретного користувача.

З інтервалом `tweets_check_period` метод `notification`:

- отримує з буфера `self.tweets` список нових твітів для вибраного клієнта;
- за допомогою функції `get_tweets` відбирає лише раніше необроблені події на основі даних з таблиці `user`;
- оновлює в базі даних мітку часу останнього твіта;
- вибирає активні налаштування сповіщень із таблиці `notification`;
- формує та надсилає повідомлення у відповідні канали Discord, з урахуванням типу події, медіа-контенту та, за потреби, згадування ролей [7, 8, 9, 10, 15, 16, 25, 28].

Таким чином, метод `notification` реалізує зв’язок між буфером твітів, базою даних налаштувань користувача та підсистемою доставки сповіщень у Discord [6, 16].

4.4. Аналіз роботи та тестування системи моніторингу новин

На етапі завершального тестування було перевірено повністю працездатну версію системи моніторингу глобальних новин, реалізовану у вигляді асинхронного Discord-бота з підключенням до соціальної платформи X (Twitter). Тестування виконувалося у реальному середовищі з використанням реальних джерел новин, активної бази користувачів та налаштованих Discord-каналів, що дозволило оцінити систему в умовах, максимально наближених до практичного використання [2, 3, 7, 8, 9, 10, 12, 13, 14, 17, 27, 29].

Запуск бота та ініціалізація фонових процесів

Після запуску програмного забезпечення відбувається початкова ініціалізація всіх компонентів системи. На даному етапі виконується перевірка змінних середовища, конфігураційних параметрів, доступності бази даних, а також підключення до Discord Gateway із використанням токена бота. У журналі подій фіксується успішне підключення до серверів Discord, синхронізація slash-команд та запуск асинхронних задач моніторингу [6, 7, 8, 9, 10, 22, 25, 28].

На рисинку 4.4 представлено приклад журналу виконання програми після запуску бота. У вікні консолі видно повідомлення про:

- успішну автентифікацію Discord-клієнта;
- перевірку конфігурації та бази даних;
- запуск задач оновлення стрічок новин (`tweets updater`);
- виявлення нових публікацій від відстежуваних акаунтів [2, 3, 6, 7, 8, 9, 10, 22, 27, 28, 29].

Ці записи підтверджують, що система коректно запускається та переходить у режим безперервного фонового моніторингу [12, 13, 14, 17].

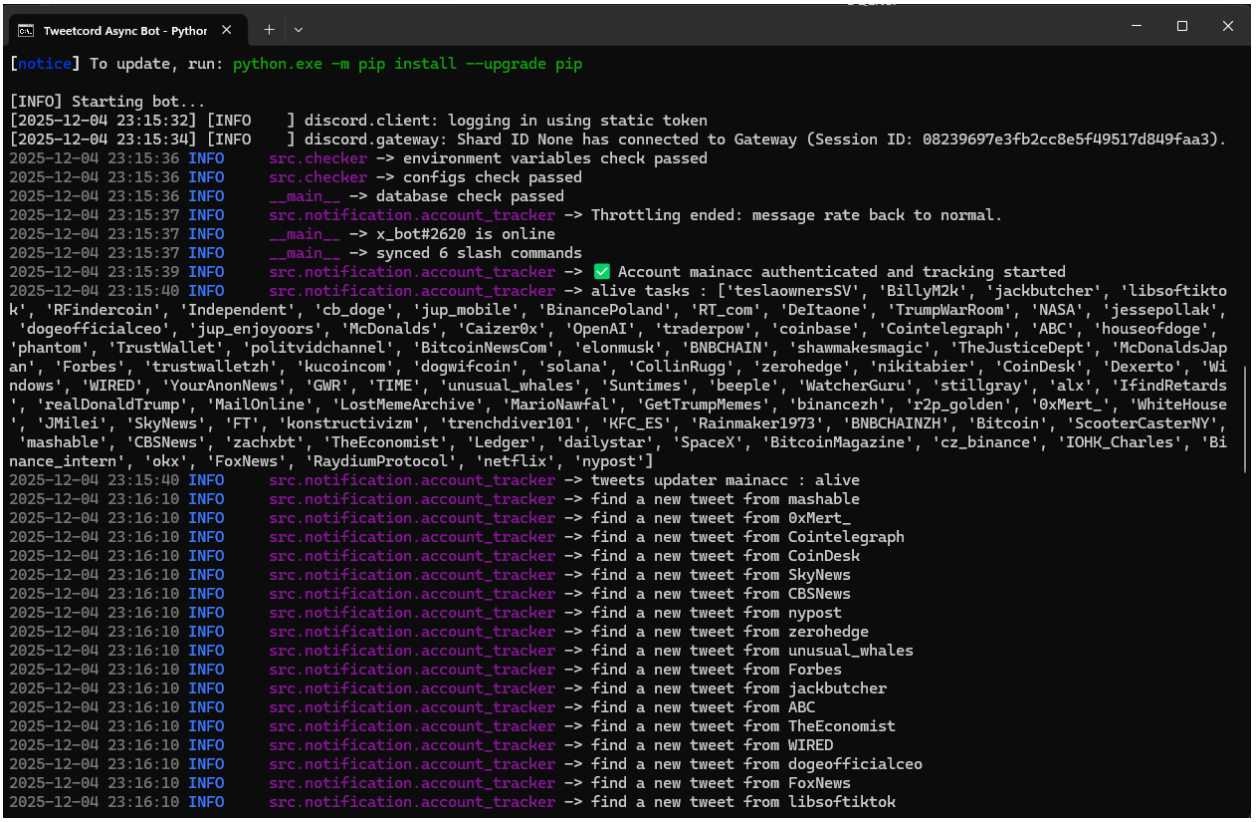

Рис. 4.4 – Журнал запуску та ініціалізації системи моніторингу новин

Отримання та доставка новин у Discord

Після запуску фонових процесів система починає регулярну перевірку нових публікацій із джерел платформи X (Twitter). Кожен новий пост проходить перевірку на унікальність шляхом порівняння ідентифікатора твіту з даними, збереженими в базі даних. Якщо новина є новою, вона зберігається у базі та передається до модуля доставки повідомлень [6, 16].

На рисунку 4.5 показано приклад роботи системи в одному з текстових каналів Discord. Повідомлення відображаються у вигляді вбудованих карток, що містять:

- джерело новини (назва акаунта);
- текст публікації;
- посилання на оригінальний твіт;
- прикріплені зображення або відео (за наявності) [7, 8, 9, 10, 28].

Такий формат подачі забезпечує наочність і дозволяє користувачам оперативно отримувати актуальні новини без необхідності переходу на сторонні ресурси [7, 8, 9, 10, 28].

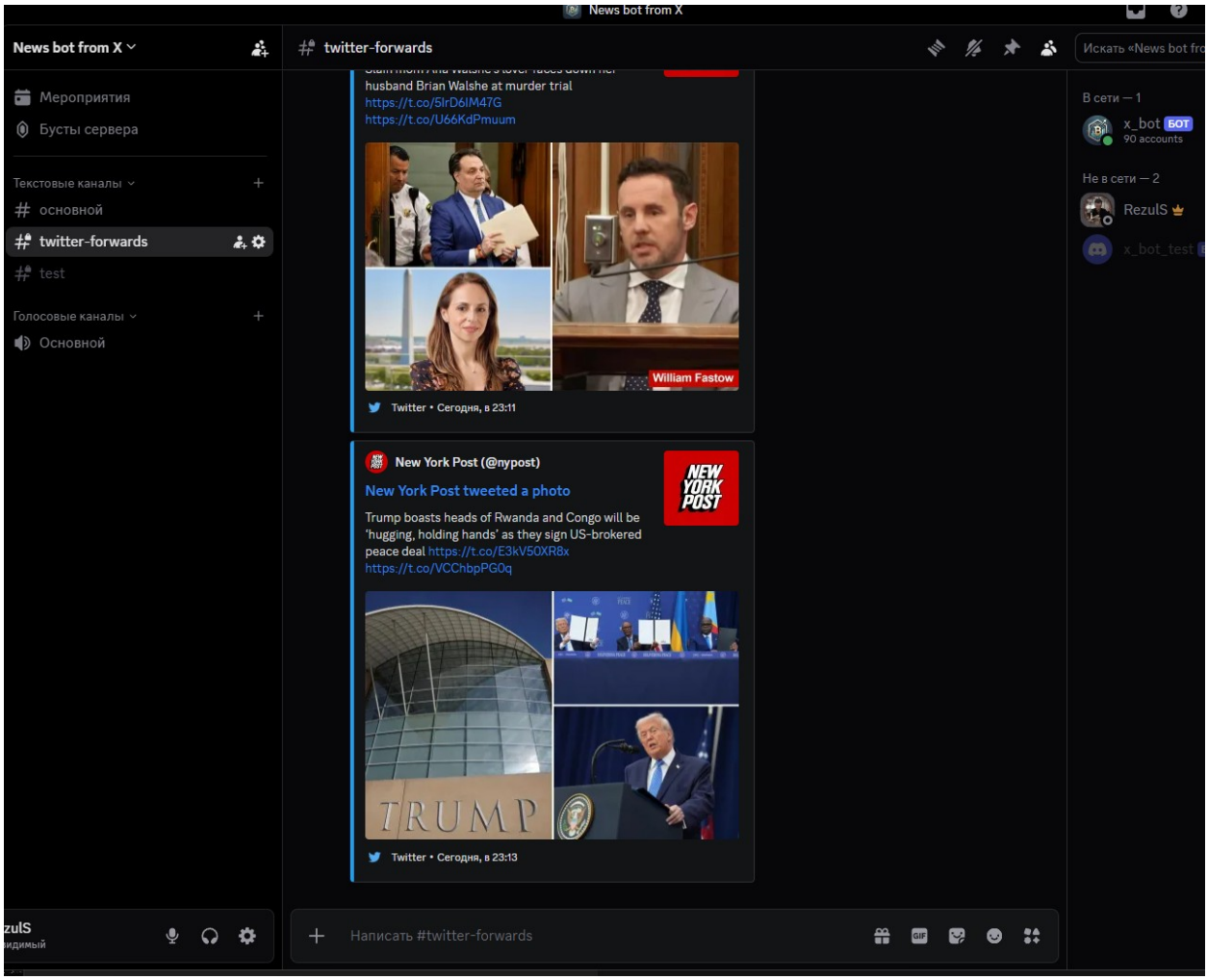

Рис. 4.5 – Приклад автоматичної доставки новин у Discord-канал

Обробка нових подій у реальному часі

У разі появи нової публікації в одному з відстежуваних акаунтів система миттєво реагує на подію. Асинхронний механізм оновлення стрічки дозволяє обробляти новини без блокування інших компонентів системи. Це особливо важливо при одночасному моніторингу великої кількості джерел [2, 3, 12, 13, 14, 17, 27, 29].

На рисунку 4.6 наведено приклад появи нової новини в Discord-каналі. Повідомлення позначене як нове та відображається без затримок, що підтверджує ефективність використаного асинхронного підходу та коректну роботу механізму черг повідомлень [2, 3, 12, 14, 17, 27, 29].

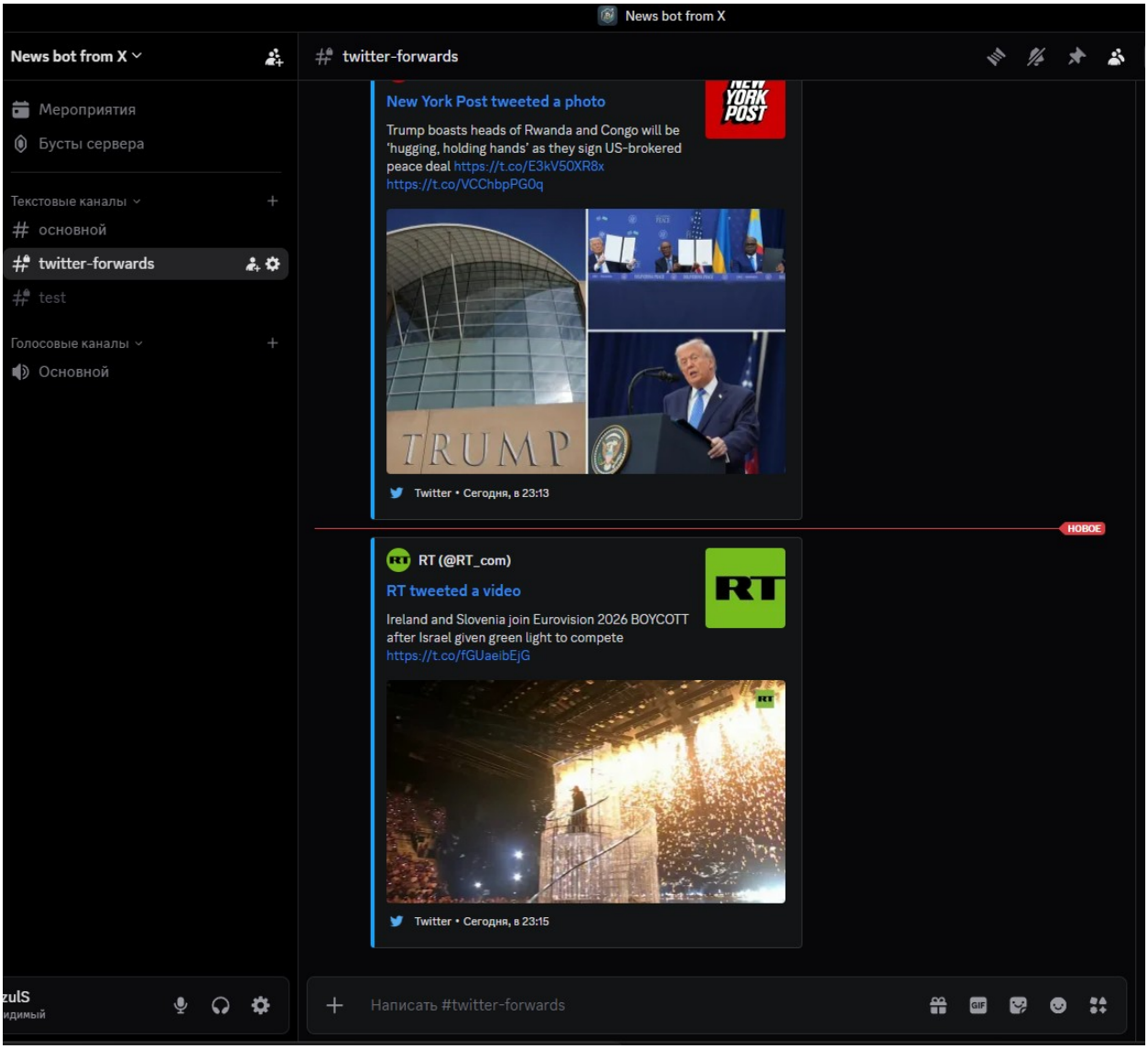

Рис. 4.6 – Доставка нової новини у режимі реального часу

Командний інтерфейс керування ботом

Окремим етапом тестування було перевірено роботу системи керування ботом через Discord-інтерфейс. Реалізований набір slash-команд дозволяє адміністраторам серверів повністю контролювати процес моніторингу без доступу до серверної частини програми [7, 8, 25, 33, 34].

На рисунку 4.7 представлено список доступних команд Discord-бота, серед яких:

- `/add notifier` - додавання нового акаунта X для моніторингу та прив’язка його до конкретного Discord-каналу;

- `/remove notifier` - видалення відстежуваного акаунта;
- `/list users` - перегляд усіх активних джерел новин;
- `/customize message` - налаштування шаблону повідомлень;
- `/sync` - синхронізація стану бота з базою даних;
- `/migrate` - перенесення підписок між токенами або клієнтами [7, 8, 25, 33, 34].

Виконання кожної команди реалізовано асинхронно та не впливає на загальну стабільність системи [2, 3, 27, 29, 30].

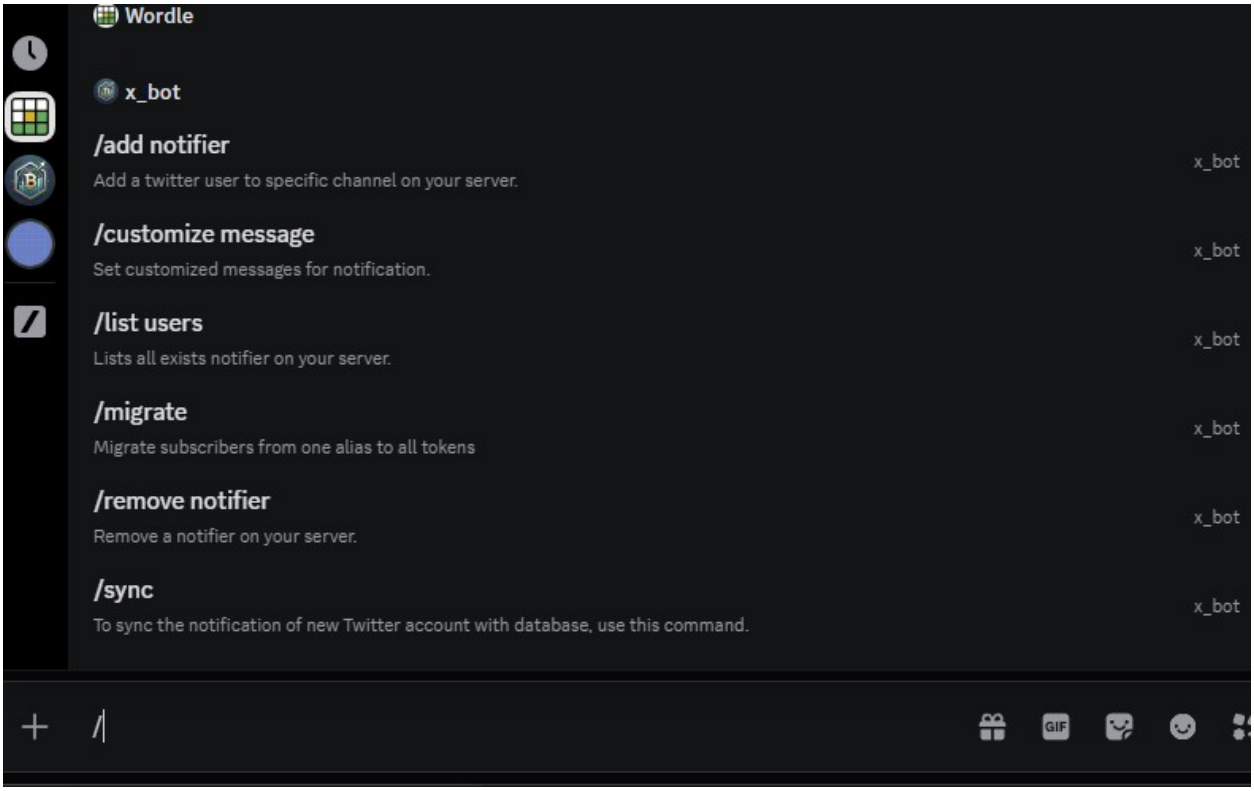

Рис. 4.7 – Командний інтерфейс керування Discord-ботом

Загальні результати тестування

Проведене тестування показало, що система:

- стабільно працює в режимі 24/7;
- коректно обробляє велику кількість джерел новин;
- не допускає дублювання повідомлень;
- забезпечує гнучке керування через Discord;

- масштабована за рахунок асинхронної архітектури [2, 3, 6, 12, 13, 14, 16, 17, 27, 29].

Реалізований підхід підтверджує ефективність використання подійно-орієнтованої архітектури в поєднанні з асинхронними механізмами обробки даних та сучасними каналами доставки інформації [12, 13, 14, 17].

4.5. Можливі напрями вдосконалення системи моніторингу новин

Розроблена система моніторингу глобальних новин продемонструвала стабільну роботу та відповідність основним вимогам систем реального часу. Водночас подальший розвиток програмного забезпечення дозволяє підвищити його функціональні можливості, продуктивність і зручність використання, а також адаптувати систему до зростаючих інформаційних потоків і розширення аудиторії користувачів [2, 3, 12, 13, 14, 17, 27, 29].

Одним із можливих напрямів удосконалення є розширення переліку джерел новинної інформації. У поточній реалізації система зосереджена на моніторингу соціальної мережі X (Twitter), однак перспективним є підключення інших джерел, таких як RSS-стрічки новинних порталів, тематичні форуми або платформи обміну повідомленнями. Застосування модульного підходу дозволяє інтегрувати нові джерела без суттєвих змін в існуючій архітектурі [2, 3, 12, 14, 17, 27, 29].

Важливим напрямом розвитку є оптимізація роботи з базою даних. За умов зростання кількості відстежуваних джерел і користувачів доцільним є впровадження механізмів кешування, індексації таблиць, а також перехід до більш продуктивних систем керування базами даних. Це дозволить зменшити час доступу до даних, підвищити швидкість перевірки на дублювання повідомлень і забезпечити стабільну роботу системи під час високих навантажень [16, 20, 24].

Наступним можливим напрямом є підвищення масштабованості системи. Використання розподілених черг повідомлень або контейнеризації компонентів дозволить розгорнути декілька екземплярів бота та рівномірно розподіляти навантаження між ними. Такий підхід особливо актуальний для великих серверів, які одночасно відстежують значну кількість новинних джерел [2, 3, 12, 13, 14, 17, 27, 29, 30].

Перспективним є також впровадження інтелектуальних механізмів фільтрації та аналізу контенту. Застосування методів обробки природної мови дасть можливість автоматично визначати тематику новин, рівень їх важливості або емоційне забарвлення. Це дозволить користувачам отримувати лише найбільш релевантну інформацію та зменшити інформаційне навантаження [1, 26, 35].

Окремої уваги заслуговує розширення каналів доставки новинної інформації. Хоча платформа Discord показала високу ефективність завдяки підтримці ролей, каналів і командного інтерфейсу, доцільним є додавання інтеграції з месенджером Telegram як альтернативним або додатковим каналом інформування. Це дозволить охопити ширшу аудиторію користувачів та підвищити універсальність системи [7, 8, 9, 10, 22, 28].

Таким чином, запропоновані напрями вдосконалення відкривають широкі можливості для подальшого розвитку системи моніторингу глобальних новин. Реалізація зазначених підходів дозволить підвищити ефективність, надійність і масштабованість програмного рішення та адаптувати його до сучасних викликів інформаційного середовища [2, 3, 12, 13, 14, 17, 27, 29].

4.6. Висновки до розділу 4

У четвертому розділі було здійснено проєктування та практичну реалізацію системи моніторингу глобальних новин із використанням асинхронної, подійно-орієнтованої архітектури. Розглянуто структурну організацію програмного забезпечення, основні модулі системи та їх взаємодію між собою в процесі збору, обробки та доставки новинної інформації кінцевим користувачам.

У межах розділу детально описано реалізацію модуля моніторингу новин, який забезпечує безперервне відстеження нових публікацій, перевірку отриманих даних на дублювання та їх подальшу обробку з використанням бази даних. Застосування асинхронних механізмів обробки дозволило забезпечити стабільну роботу системи в режимі реального часу та ефективну взаємодію між її компонентами.

Також було проведено аналіз роботи розробленої системи з використанням реальних прикладів функціонування програмного забезпечення. Продемонстровано процес запуску бота, отримання нових новинних повідомлень, їх відображення в середовищі Discord та використання командного інтерфейсу для керування системою. Результати тестування підтвердили коректність роботи основних алгоритмів та надійність обраних технічних рішень.

Окрему увагу було приділено можливим напрямам вдосконалення системи, серед яких розширення переліку джерел новин, підвищення масштабованості, оптимізація роботи з базою даних та впровадження інтелектуальних механізмів аналізу контенту. Запропоновані напрями розвитку дозволяють розглядати створену систему як гнучку та перспективну платформу для подальших досліджень і практичного застосування.

Таким чином, у четвертому розділі досягнуто поставленої мети щодо побудови, реалізації та аналізу роботи системи моніторингу глобальних новин, що підтверджує доцільність використання обраних архітектурних підходів і технологій.

ВИСНОВКИ

У даній кваліфікаційній роботі було досягнуто поставленої мети - розроблено систему моніторингу глобальних новин у режимі реального часу з використанням асинхронної обробки повідомлень та сучасних каналів комунікації. Створене програмне рішення забезпечує оперативний збір, обробку та доставку новинної інформації з мінімальними затримками, зберігаючи її актуальність та цілісність.

У процесі виконання роботи було вивчено особливості побудови систем реального часу, зокрема вимоги до швидкодії, надійності та безперервності обробки подій. Досліджено методи та алгоритми збору, обробки і фільтрації новинної інформації, а також подійно-орієнтовані підходи, що дозволяють ефективно працювати з великими інформаційними потоками в асинхронному середовищі.

Проведено аналіз існуючих систем і сервісів моніторингу новин, що дало змогу визначити їх основні переваги та обмеження, а також обґрунтувати доцільність розробки власного програмного рішення. Окрему увагу приділено дослідженню технологій асинхронної обробки повідомлень, які стали основою архітектури розробленої системи.

У межах роботи було спроєктовано архітектуру системи моніторингу глобальних новин, що включає модулі збору даних з інформаційних джерел, асинхронної обробки подій, збереження стану у базі даних та доставки повідомлень кінцевим користувачам. Реалізовано механізм збору та обробки новин із зовнішніх інформаційних джерел із застосуванням фонових асинхронних задач та механізмів уникнення дублювання повідомлень.

Для забезпечення оперативної доставки новинної інформації було реалізовано Telegram- та Discord-ботів, які підтримують надсилання повідомлень у реальному часі з урахуванням налаштувань користувачів, типів подій та каналів доставки. Використання бази даних дозволило зберігати стан системи, контролювати послідовність подій та забезпечувати узгодженість обробки повідомлень у багатокористувацькому середовищі.

Проведено тестування розробленої системи в реальних умовах експлуатації та проаналізовано отримані результати. За результатами тестування підтверджено коректність роботи основних алгоритмів, стабільність асинхронних процесів та здатність системи функціонувати безперервно протягом тривалого часу без втрати даних і повторного надсилання повідомлень.

У ході роботи також визначено можливі напрями подальшого вдосконалення системи, зокрема розширення переліку джерел новин, підтримку додаткових платформ доставки повідомлень, оптимізацію взаємодії з базою даних, а також впровадження інтелектуальних методів аналізу та фільтрації новинного контенту.

Отже, у результаті виконання кваліфікаційної роботи було розроблено працездатну та масштабовану систему моніторингу глобальних новин у режимі реального часу, яка поєднує асинхронну архітектуру, гнучкі механізми керування та інтеграцію з сучасними каналами комунікації. Отримані результати можуть бути використані для подальшого розвитку інформаційно-аналітичних систем і мають практичну цінність у сферах аналітики, медіамоніторингу, фінансів та інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Beginner Guide to Telegram Bot API [Електронний ресурс]. – Режим доступу: https://apidog.com/blog/beginners-guide-to-telegram-bot-api/
2. A Conceptual Overview of asyncio [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/howto/a-conceptual-overview-of-asyncio.html Python documentation
3. asyncio - асинхронне введення/виведення (укр. версія документації Python) [Електронний ресурс]. – Режим доступу: https://docs.python.org/uk/3.13/library/asyncio.html Python documentation
4. aiosqlite Documentation [Електронний ресурс]. – Режим доступу: https://github.com/omnilib/aiosqlite
5. BBC R&D. Python Asyncio Part 1 – Basic Concepts and Patterns [Електронний ресурс]. – Режим доступу: https://bbc.github.io/cloudfit-public-docs/asyncio/asyncio-part-1.html
6. Buttazzo G. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. – Springer, 2011. – 524 p.
7. Discord Developer Portal. API Reference [Електронний ресурс]. – Режим доступу: https://discord.com/developers/docs/reference
8. Discord Developer Portal. Intro [Електронний ресурс]. – Режим доступу: https://discord.com/developers/docs/intro
9. Discord Developer Portal. Messages Resource [Електронний ресурс]. – Режим доступу: https://discord.com/developers/docs/resources/message
10. discord.py – API wrapper for Discord [Електронний ресурс]. – Режим доступу: https://discordpy.readthedocs.io/
11. Estuary. Event-Driven Architecture in Practice for Real-Time Data Flows [Електронний ресурс]. – Режим доступу: https://estuary.dev/blog/event-driven-architecture-examples

12. Event-Driven Architecture (EDA): A Complete Guide to Real-Time Systems [Електронний ресурс]. – Режим доступу: https://medium.com/the-software-frontier/event-driven-architecture-eda-a-complete-guide-to-real-time-systems-974f612dc6b5
13. Event-Driven Architecture Examples: Real-World Use Cases [Електронний ресурс]. – Режим доступу: https://estuary.dev/blog/event-driven-architecture-examples/
14. Event-driven architecture (EDA): A Complete Introduction [Електронний ресурс]. – Режим доступу: https://www.confluent.io/learn/event-driven-architecture/
15. FlowWright. How Event-Driven Architecture Powers Real-Time Systems [Електронний ресурс]. – Режим доступу: https://www.flowwright.com/how-event-driven-architecture-powers-real-time-systems
16. Fowler M. Patterns of Enterprise Application Architecture. – Addison-Wesley, 2002.
17. How to Implement Event-Driven Architecture for Real-Time Apps [Електронний ресурс]. – Режим доступу: https://marutitech.com/event-driven-architecture-real-time-apps/
18. Kleppmann M. Designing Data-Intensive Applications. – O'Reilly Media, 2017. – 616 p.
19. Kopetz H. Real-Time Systems: Design Principles for Distributed Embedded Applications. – Springer, 2011. – 378 p.
20. Liu J.W.S. Real-Time Systems. – Prentice Hall, 2000. – 560 p.
21. Logging HOWTO – Python 3 [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/howto/logging.html
22. Newman S. Building Microservices: Designing Fine-Grained Systems. – O'Reilly Media, 2015.
23. Official Discord API Rate Limits Guide [Електронний ресурс]. – Режим доступу: https://discord.com/developers/docs/topics/rate-limits

24. Practical Guide to Event-Driven Architecture [Електронний ресурс]. – Режим доступу: https://www.createq.com/en/insights/practical-guide-to-event-driven-architecture
25. Python Discord. Discord.py Learning Guide [Електронний ресурс]. – Режим доступу: https://www.pythondiscord.com/pages/guides/python-guides/discordpy/
26. python-telegram-bot. Офіційна документація бібліотеки [Електронний ресурс]. – Режим доступу: https://docs.python-telegram-bot.org/
27. Python Software Foundation. asyncio - Asynchronous I/O [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/asyncio.html
28. Rapptz. discord.py: An API wrapper for Discord written in Python [Електронний ресурс]. – Режим доступу: https://github.com/Rapptz/discord.py
29. Real Python. Python’s asyncio: A Hands-On Walkthrough [Електронний ресурс]. – Режим доступу: https://realpython.com/async-io-python/
30. SQLite Documentation [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/docs.html
31. Tanenbaum A.S., Van Steen M. Distributed Systems: Principles and Paradigms. – 2nd ed. – Prentice Hall, 2007.
32. tdlib/telegram-bot-api. Telegram Bot API server implementation [Електронний ресурс]. – Режим доступу: https://github.com/tdlib/telegram-bot-api
33. Telegram APIs. Офіційна документація платформи Telegram [Електронний ресурс]. – Режим доступу: https://core.telegram.org/
34. Telegram Bot API [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api
35. Telegram Bot API. Bot API Library Examples [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/samples
36. Telegram: Bots. An introduction for developers [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots

37. THE POWER OF EVENT-DRIVEN ARCHITECTURE: ENABLING REAL-TIME SYSTEMS AND SCALABLE SOLUTIONS // ResearchGate [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/385668247
38. tlgrm.ru. Справочник по Telegram Bot API [Електронний ресурс]. – Режим доступу: https://tlgrm.ru/docs/bots/api

Додаток А

Ініціалізація Discord-бота та запуск системи (фрагмент файла `bot.py`)

```
import asyncio
import os
import sys
import discord
from discord.ext import commands
from dotenv import load_dotenv
from configs.load_configs import configs
from src.checker import check_configs, check_env, check_db, check_upgrade
from src.db_function.init_db import init_db
from src.db_function.repair_db import auto_repair_mismatched_clients
from src.presence_updater import update_presence
from src.log import setup_logger
from src.notification.account_tracker import AccountTracker

log = setup_logger(__name__)
intents = discord.Intents.default()
intents.message_content = True
intents.guilds = True
bot = commands.Bot(command_prefix=configs['prefix'], intents=intents)
@bot.event
async def on_ready():
    log.info(f'Logged in as {bot.user} (ID: {bot.user.id})')
    await update_presence(bot)
                    if not (os.path.isfile(os.path.join(os.getenv('DATA_PATH'),
'tracked_accounts.db'))):
        await init_db()
```

```
    await auto_repair_mismatched_clients()
    check_upgrade()
    if not check_env():
        log.warning('incomplete environment variables detected, will retry in 30 seconds')
        await asyncio.sleep(30)
        load_dotenv()
    if not check_configs(configs):
        log.warning('incomplete configs file detected, will retry in 30 seconds')
        await asyncio.sleep(30)
        os.execv(sys.executable, ['python'] + sys.argv)
    AccountTracker(bot)
    log.info('AccountTracker initialized and background tasks started')
if __name__ == '__main__':
    bot.run(os.getenv('BOT_TOKEN'))
```

Фрагмент демонструє точку входу програмного застосунку та процес запуску системи моніторингу.

У даному фрагменті здійснюється:

- створення екземпляра commands.Bot із необхідними параметрами та дозволами (intents);
- перевірка коректності конфігураційних файлів, змінних середовища та стану бази даних;
- ініціалізація бази даних tracked_accounts.db при першому запуску системи;
- автоматичне усунення можливих невідповідностей у структурі бази даних;
- ініціалізація модуля AccountTracker, який запускає асинхронні задачі моніторингу активності облікових записів платформи X (Twitter) та надсилання повідомлень користувачам через Discord.

Після виконання зазначених етапів система переходить у режим безперервного моніторингу глобальних новин у реальному часі.

ДОДАТОК Б

Клас `AccountTracker`: організація асинхронного моніторингу новин (файл `src/notification/account_tracker.py`

```
class AccountTracker():
    def __init__(self, bot: commands.Bot):
        self.bot = bot
        self.accounts_data = get_accounts()
        self.db_path = os.path.join(os.getenv('DATA_PATH'), 'tracked_accounts.db')
                        self.tweets = {account_name: [] for account_name in
self.accounts_data.keys()}
                          self.tasksMonitorLogAt = datetime.now(timezone.utc) -
timedelta(hours=configs['tasks_monitor_log_period'])
        bot.loop.create_task(self.setup_tasks())

    async def setup_tasks(self):
        tasks = []
        async def authenticate_and_track(account_name, account_token):
            try:
                app = Twitter(account_name)
                max_attempts = configs['auth_max_attempts']
                for attempt in range(max_attempts):
                    try:
                        await app.load_auth_token(account_token)
                        break
                    except Exception as e:
                        if attempt < max_attempts - 1:
                            await asyncio.sleep(2)
```

```
                else:
                    raise e
            self.tweets[account_name] = []
            self.bot.loop.create_task(
                self.tweetsUpdater(app)
            ).set_name(f"TweetsUpdater_{account_name}")
                log.info(f"    Account {account_name} authenticated and tracking started")
        except Exception as e:
                log.error(f"    Failed to authenticate {account_name}: {e}")

    for account_name, token in self.accounts_data.items():
            tasks.append(asyncio.create_task(authenticate_and_track(account_name, token)))
    await asyncio.gather(*tasks)
    async with connect_readonly(self.db_path) as db:
            async with db.execute('SELECT username, client_used FROM user WHERE enabled = 1') as cursor:
            usernames_and_clients = {row[0]: row[1] async for row in cursor}
    for username, client_used in usernames_and_clients.items():
                                self.bot.loop.create_task(self.notification(username, client_used)).set_name(username)
    self.bot.loop.create_task(self.tasksMonitor(usernames_and_clients)).set_name('TasksMonitor')
  async def notification(self, username: str, client_used: str):
    while True:
        await asyncio.sleep(configs['tweets_check_period'])
        lastest_tweets = await get_tweets(self.tweets[client_used], username)
```

```
        if lastest_tweets is None:
            continue
        async with aiosqlite.connect(self.db_path) as db:
            db.row_factory = aiosqlite.Row
            async with db.cursor() as cursor:
                await cursor.execute('SELECT * FROM user WHERE username = ?',
(username,))
                user = await cursor.fetchone()
                async with lock:
                    await cursor.execute('UPDATE user SET lastest_tweet = ? WHERE
username = ?',
                                         (str(lastest_tweets[-1].created_on), username))
                    await db.commit()
                for tweet in lastest_tweets:
                    log.info(f'find a new tweet from {username}')
                        await cursor.execute('SELECT * FROM notification WHERE
user_id = ? AND enabled = 1',
                                             (user['id'],))
                    notifications = await cursor.fetchall()
                    for data in notifications:
                        channel = self.bot.get_channel(int(data['channel_id']))
                                if channel is not None and is_match_type(tweet,
data['enable_type']) and \
                                    is_match_media_type(tweet, data['enable_media_type']):
                                try:
                                                                        mention =
f"{channel.guild.get_role(int(data['role_id'])).mention} " if data['role_id'] else ''
                                    author, action = tweet.author.name, get_action(tweet)
                                        url = re.sub('twitter', DOMAIN_NAME, tweet.url) if
EMBED_TYPE == 'fx_twitter' else tweet.url
```

```
                                msg = data['customized_msg'] if data['customized_msg'] else configs['default_message']
                                    msg = msg.format(mention=mention, author=author, action=action, url=url)
                            if EMBED_TYPE != 'built_in':
                                await send_with_throttle(channel.send, msg)
                            else:
                                                img = discord.File('images/twitter.png', filename='twitter.png')
                                embeds = await gen_embed(tweet)
                                  await send_with_throttle(channel.send, None, file=img, embeds=embeds)
                        except Exception as e:
                            if not isinstance(e, discord.errors.Forbidden):
                                log.error(f'an error occurred at {channel.mention} while sending notification: {e}')
    async def tweetsUpdater(self, app: Twitter):
        updater_name = asyncio.current_task().get_name().split('_', 1)[1]
        while True:
            try:
                self.tweets[updater_name] = await app.get_tweet_notifications()
                await asyncio.sleep(configs['tweets_check_period'])
            except Exception as e:
                log.error(f'{e} (task : tweets updater {updater_name})')
                            log.error(f"an unexpected error occurred, try again in {configs['tweets_updater_retry_delay']} minutes")
                await asyncio.sleep(configs['tweets_updater_retry_delay'] * 60)
```

Пояснення до фрагмента (AccountTracker):

Фрагмент демонструє організацію асинхронного моніторингу та доставки повідомлень. У методах __init__ та setup_tasks() виконується завантаження конфігурації відстежуваних X-акаунтів, їх автентифікація та створення фонових задач TweetsUpdater_<account> для періодичного отримання нових подій. Додатково формується набір задач notification для кожного активного користувача системи на основі даних з БД.

Метод notification(...) реалізує цикл опитування буфера подій, оновлення стану користувача (поле lastest_tweet) та формування повідомлень до Discord-каналів із урахуванням типів подій, медіа та ролей.

Метод tweetsUpdater(...) забезпечує періодичне отримання нових подій через get_tweet_notifications() та включає механізм повторної спроби з затримкою у разі помилок, що підвищує надійність роботи сервісу.

ДОДАТОК В

Функція `get_tweets`: фільтрація нових новин за станом БД (файл `src/notification/get_tweets.py`)

```
import os
from typing import Optional
from src.db_function.readonly_db import connect_readonly
from tweety.types import Tweet
from src.notification.date_comparator import date_comparator

async def get_tweets(tweets: list[Tweet], username: str) -> Optional[list[Tweet]]:
            async with connect_readonly(os.path.join(os.getenv('DATA_PATH'), 'tracked_accounts.db')) as db:
        async with db.execute('SELECT lastest_tweet FROM user WHERE username = ?', (username,)) as cursor:
            row = await cursor.fetchone()
            last_tweet_at = row[0]

    tweets = [
        tweet for tweet in tweets
        if tweet.author.username == username
        and date_comparator(tweet.created_on, last_tweet_at) == 1
    ]
    if tweets != []:
        return sorted(tweets, key=lambda x: x.created_on)
    else:
        return None
```

Функція get_tweets реалізує механізм уникнення дублювання новин шляхом порівняння часових міток.

На основі збереженого у базі даних значення lastest_tweet для конкретного користувача здійснюється відбір лише тих твітів, час створення яких перевищує останній зафіксований. Для цього використовується допоміжна функція date_comparator.

У результаті функція повертає впорядкований за часом список нових подій або None, якщо нових записів не виявлено. Такий підхід забезпечує коректну роботу системи в асинхронному середовищі та збереження узгодженого стану між процесами моніторингу і базою даних.