ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**Система моніторингу великих рухів капіталу на блокчейні**»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
(код,найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Назар ДЯЧУК
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти група ІСД-41
Назар ДЯЧУК
(ім'я, ПРІЗВИЩЕ)

Керівник Ольга ПОЛОНЕВИЧ
к.т.н.,доцент *(ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
к.т.н.,доцент *(ім'я, ПРІЗВИЩЕ)*

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ
_____ Каміла СТОРЧАК

“____” _____ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дячуку Назару Ігоровичу
(прізвище, ім’я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Система моніторингу великих рухів капіталу на блокчейні
керівник кваліфікаційної роботи: Ольга ПОЛОНЕВИЧ к.т.н., доцент
(ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від “20 ” лютого 2026 р. № 51

2. Строк подання кваліфікаційної роботи «01» червня 2026 р.

3. Вихідні дані кваліфікаційної роботи:
1. Документація платформи Ethereum та протоколу WebSocket.
2. Бібліотеки web3.py, FastAPI, aiogram, Motor, Chart.js, D3.js.
3. Науково-технічна література з блокчейн-технологій

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Теоретичні основи моніторингу блокчейн-транзакцій.
2. Архітектура та реалізація системи моніторингу.
3. Тестування та оцінка ефективності системи.

5.Перелік ілюстраційного матеріалу: *презентація*

6. Дата видачі завдання «20» лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Збір та аналіз літературних джерел.	25.02-10.03.26	
2.	Аналіз існуючих рішень моніторингу	11.03-20.03.26	
3.	Проєктування архітектури системи.	21.03-01.04.26	
4.	Розробка Whale Tracker та API.	02.04-15.04.26	
5.	Розробка Telegram-бота.	16.04-22.04.26	
6.	Розробка вебдашборду.	23.04-05.05.26	
7.	Тестування, розгортання на сервері.	06.05-14.05.26	
8.	Оформлення бакалаврської роботи	15.05-30.05.26	

Здобувач вищої освіти _____ Назар ДЯЧУК
(підпис) *(ім'я, ПРІЗВИЩЕ)*

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

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 12 рис., 5 табл., 22 джерела.

Мета роботи - покращення аналізу крипторинку шляхом розробки системи моніторингу з можливістю налаштування порогових значень та отримання сповіщень у реальному часі.

Об'єкт дослідження - процеси виявлення та аналізу великих транзакцій у публічних блокчейн-мережах.

Предмет дослідження - методи та інструменти аналізу блокчейн-транзакцій на основі відкритих API та децентралізованих даних.

Короткий зміст роботи. У роботі проведено аналіз принципів функціонування мережі Ethereum та механізмів доступу до даних блокчейну. Досліджено існуючі рішення моніторингу (Etherscan, Nansen, Whale Alert) та визначено їхні обмеження. Розроблено трикомпонентну систему: Whale Tracker для відстеження транзакцій через WebSocket, Telegram-бот для персоналізованих сповіщень з налаштовуваними порогами, вебдашборд з інтерактивними графіками та аналітикою. Реалізовано асинхронну архітектуру на базі Python/FastAPI, збереження в MongoDB, систему розгорнуто на VPS-сервері з HTTPS доступом за адресою whale-tracker.website.

КЛЮЧОВІ СЛОВА: БЛОКЧЕЙН, ВЕЛИКІ ТРАНЗАКЦІЇ, КРИПТОАНАЛІТИКА, АНАЛІТИКА, БЛОКЧЕЙН-ТРАНЗАКЦІЇ, АНАЛІЗ, КРИПТОРИНОК, ETHEREUM, WHALE TRACKER

Зміст

ВСТУП

Актуальність теми. Технологія блокчейн та децентралізовані фінанси (DeFi) стрімко розвиваються, залучаючи мільярди доларів щоденного обороту. За даними аналітичних платформ, щоденний обсяг транзакцій у мережі Ethereum у 2024 році перевищував 3 млрд доларів США. Серед цих транзакцій особливий інтерес представляють так звані «whale» (кит) транзакції - переміщення значних обсягів криптовалюти, які можуть суттєво впливати на ринок. Моніторинг великих рухів капіталу в блокчейні є важливим інструментом для трейдерів, аналітиків та дослідників ринку. Своєчасне виявлення активності великих гравців дозволяє приймати більш обґрунтовані інвестиційні рішення, передбачати цінові коливання та розуміти загальні тенденції ринку. Дослідження показують, що великі переміщення коштів між гаманцями та біржами часто передують значним ціновим коливанням на 15-30 хвилин.

Незважаючи на існування комерційних рішень (Etherscan, Nansen, Whale Alert), вони мають суттєві обмеження: платний доступ до розширених функцій, відсутність персоналізованих сповіщень, обмежені можливості для налаштування порогів та інтеграції з месенджерами. Whale Alert, наприклад, не дозволяє налаштовувати індивідуальні порогові значення для кожного користувача. Це зумовлює актуальність розробки відкритої системи з гнучким налаштуванням.

Мета роботи - покращення аналізу крипторинку шляхом розробки системи моніторингу з можливістю налаштування порогових значень та отримання сповіщень у реальному часі. Для досягнення мети, у кваліфікаційній роботі поставлені наступні завдання:

1. Дослідити принципи функціонування мережі Ethereum та механізмів доступу до даних блокчейну;
2. Оглянути існуючі рішення моніторингу та визначення їхніх переваг і недоліків;
3. Спроектувати архітектуру системи з урахуванням вимог до надійності та масштабованості;

4. Реалізувати модуль відстеження транзакцій через WebSocket зєднання з вузлами Ethereum;

5. Розробити Telegram-бот для надсилання персоналізованих сповіщень та створити вебдашборд з інтерактивними графіками та аналітичними функціями;

6. Протестувати та оцінити ефективність.

Об'єкт дослідження - процеси виявлення та аналізу великих транзакцій у публічних блокчейн-мережах.

Предмет дослідження - методи та інструменти аналізу блокчейн-транзакцій на основі відкритих API та децентралізованих даних.

Методи дослідження. Під час написання кваліфікаційної роботи були використані методи системного аналізу, асинхронного програмування, порівняльного аналізу існуючих рішень.

Практична значущість одержаних результатів. Розроблена система надає трейдерам, криптоаналітикам інструмент для оперативного виявлення великих рухів капіталу в мережі Ethereum, що дозволяє приймати більш обґрунтовані інвестиційні рішення та своєчасно реагувати на зміни ринкової кон'юнктури. Система Whale Tracker функціонує за адресою whale-tracker.website, відстежує транзакції Ethereum у реальному часі та надає аналітичні дані через веб-інтерфейс і Telegram-бот (@Eth_WhaleTracker_Bot).

Апробація результатів та публікації. Основні положення і результати кваліфікаційної роботи доповідались на науково практичних конференціях, що проходили на базі Державного університету інформаційно-комунікаційних технологій. Тезу на тему “Ознаки ринкових маніпуляцій через призму аналізу великих транзакцій у блокчейні” було опубліковано на VII Міжнародній науково-технічній конференції 20 квітня 2026 року, та тезу на тему “Вплив великих транзакцій у блокчейні на динаміку криптовалютного ринку: аналітичне дослідження” було опубліковано на III Всеукраїнській науково-технічній конференції 18 листопада 2025 року.

РОЗДІЛ 1. ОСНОВИ МОНІТОРИНГУ БЛОКЧЕЙН-ТРАНЗАКЦІЙ

1.1 Технологія блокчейн та принципи децентралізації

Блокчейн (blockchain) - це розподілена база даних або реєстр, що складається з послідовного ланцюжка блоків, кожен з яких містить набір транзакцій та пов'язаний з попереднім через криптографічний хеш. Технологія забезпечує незмінність та прозорість зберігання даних без потреби в централізованому органі управління. Концепція блокчейну була вперше описана Сатоші Накамото у 2008 році у «Bitcoin Whitepaper»[1] як технологічна основа для першої децентралізованої криптовалюти. З того часу технологія суттєво еволюціонувала та знайшла застосування далеко за межами фінансових транзакцій: управління ланцюгами постачання, охорона здоров'я, цифрова ідентичність, децентралізоване управління.

Принцип роботи блокчейну базується на криптографічному зв'язку блоків. Кожен новий блок містить хеш попереднього блоку у своєму заголовку. Будь-яка спроба змінити дані в одному блоці змінить його хеш та зробить недійсними всі наступні блоки. Для підробки необхідно було б переписати весь ланцюжок та перевершити обчислювальну потужність решти вузлів мережі. Ключові властивості технології блокчейн, що забезпечують її надійність та безпеку:

1. Децентралізація - відсутність єдиного контролюючого органу; мережа функціонує на основі консенсусу розподілених учасників;
2. Незмінність (immutability) - після запису транзакції у блокчейн її неможливо змінити або видалити без перерахунку всіх наступних блоків, що потребує неможливо великих обчислювальних ресурсів;
3. Прозорість (transparency) - усі транзакції є публічно доступними та верифікованими будь-яким учасником мережі без спеціальних дозволів;

4. Псевдонімність (pseudonymity) - учасники ідентифікуються криптографічними адресами, а не особистими даними, що забезпечує певний рівень анонімності;
5. Відмовостійкість (fault tolerance) - відсутність єдиної точки відмови; мережа продовжує функціонувати навіть при виході з ладу частини вузлів.

Структура блоку складається з двох основних частин. Заголовок блоку (block header) містить: хеш попереднього блоку (previous block hash) для зв'язку в ланцюжку, мітку часу (timestamp), значення складності (difficulty target), нонс (nonce) для PoW або дані валідатора для PoS, корінь дерева Меркле (Merkle root) усіх транзакцій блоку. Тіло блоку містить перелік транзакцій, що підтверджуються даним блоком. Дерево Меркле (Merkle Tree) (Рис. 1.1) - це бінарне хеш-дерево, що дозволяє ефективно верифікувати вміст великих наборів даних. Кожен листовий вузол дерева є хешем транзакції, а кожен не-листовий вузол - хешем своїх дочірніх вузлів. Корінь Меркле є єдиним хешем, що криптографічно представляє всі транзакції блоку. Це дозволяє перевірити наявність конкретної транзакції в блоці без завантаження та перевірки всього блоку, що є важливим для легких клієнтів.

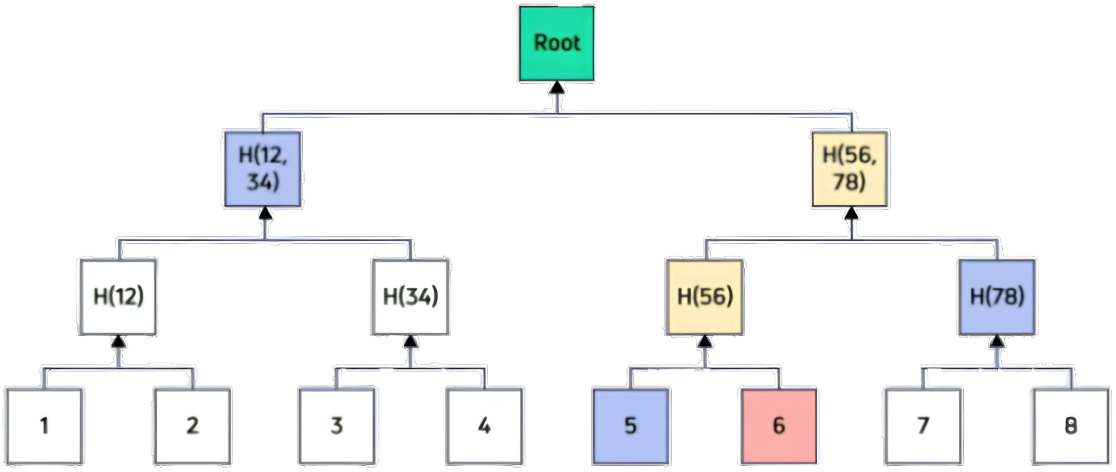


Рис. 1.1 Структура блоку Ethereum: заголовок та дерево Меркле

Відкритість блокчейну Ethereum відіграє ключову роль у розвитку аналітичних інструментів. Оскільки всі транзакції є публічно доступними, будь-який учасник мережі може самостійно перевірити будь-яку операцію або розробити власний інструмент для аналізу даних. Саме ця прозорість стала

основою для появи цілої індустрії блокчейн-аналітики, до якої відноситься і система, розроблена в рамках даної роботи. Механізми консенсусу забезпечують узгодженість стану розподіленого реєстру між усіма вузлами мережі. Proof of Work (PoW) вимагає від вузлів вирішення обчислювально складного завдання - знаходження такого значення nonce, щоб хеш блоку був меншим за задане цільове значення. Складність задачі автоматично коригується мережею для підтримки постійної частоти блоків. Цей механізм потребує значних енергетичних витрат: у 2021 році мережа Ethereum споживала ~78 ТВт/год на рік. Proof of Stake (PoS), що застосовується в Ethereum після The Merge (вересень 2022 р.), обирає валідатора для пропозиції нового блоку на основі суми заблокованих ETH (stake). Для участі у валідації необхідно заблокувати мінімум 32 ETH. Відповідальні валідатори отримують винагороду, а зловмисні штрафуються списанням частини застави (slashing). Перехід на PoS знизив енергоспоживання мережі приблизно на 99.95%[2]. Смарт-контракти - це самовиконувані програми, що зберігаються в блокчейні та автоматично виконуються при настанні визначених умов без участі третіх осіб. Код смарт-контракту є незмінним після деплойменту та виконується детерміновано на всіх вузлах мережі. Смарт-контракти є ключовою інновацією Ethereum, що відрізняє його від Bitcoin та дозволяє реалізовувати складні децентралізовані протоколи (DeFi, NFT, DAO).

Ethereum Virtual Machine (EVM) є ізольованим середовищем виконання, що функціонує однаково на кожному вузлі мережі незалежно від апаратної платформи. Ізольованість EVM означає, що код смарт-контракту не має доступу до файлової системи, мережі або інших процесів вузла. Виконання коду є детермінованим: для однакових вхідних даних та стану мережі результат завжди однаковий на будь-якому вузлі - це є фундаментальною вимогою для досягнення консенсусу в розподіленій системі. Lifecycle смарт-контракту складається з чотирьох етапів. На першому етапі розробник пише код контракту мовою Solidity або Vyper. На другому етапі код компілюється у байт-код EVM - низькорівневу послідовність інструкцій (opcodes). На третьому етапі відбувається деплоймент: транзакція з порожнім полем to та байт-кодом у полі data надсилається в мережу,

EVM виконує ініціалізаційний код та записує байт-код контракту за новою адресою. На четвертому етапі користувачі взаємодіють з контрактом через транзакції виклику (call transactions), що містять ABI-закодований селектор функції та аргументи у полі data.

Механізм газу зазнав суттєвих змін після впровадження EIP-1559 у серпні 2021 року. До EIP-1559 ціна газу визначалась аукціоном: користувачі вказували gasPrice, і майнери включали транзакції з вищою ціною. Це призводило до значних коливань комісій та складності прогнозування витрат. Після EIP-1559 введено дворівневу структуру комісій. Базова комісія (base fee) визначається алгоритмічно мережею залежно від завантаженості попереднього блоку та повністю спалюється - вилучається з обігу. Чайові (priority fee або maxPriorityFeePerGas) сплачуються безпосередньо валідатору як стимул для включення транзакції в блок. Максимальна комісія (maxFeePerGas) встановлює верхню межу загальних витрат користувача: якщо base fee + priority fee менше maxFeePerGas, різниця повертається відправнику. Механізм спалювання базової комісії має дефляційний ефект на загальну пропозицію ETH. У періоди високого попиту на мережу обсяг спалення може перевищувати емісію нових ETH для валідаторів, що робить Ethereum дефляційним активом. Це явище отримало назву «ultra sound money» у спільноті Ethereum та є одним з факторів, що визначає поведінку великих гравців ринку при прийнятті рішень про переміщення капіталу - що безпосередньо пов'язано з предметом дослідження даної роботи.

1.2 Мережа Ethereum: архітектура та механізм транзакцій

Ethereum - публічна блокчейн-платформа з відкритим кодом, що була запущена у липні 2015 року Віталіком Бутеріним, Гевіном Вудом та їх співзасновниками. Ethereum є другою за ринковою капіталізацією криптовалютою та найбільшою платформою для смарт-контрактів у світі[3]. Графовий аналіз транзакцій мережі Ethereum демонструє складну структуру взаємодій між адресами, що охоплює мільйони вузлів та ребер [21]. Архітектура мережі

Ethereum після The Merge (вересень 2022) складається з двох рівнів. Execution Layer (рівень виконання) обробляє транзакції та виконує смарт-контракти через EVM. Consensus Layer (рівень консенсусу) координує вузли мережі через протокол Gasper (комбінація Casper FFG та LMD-GHOST). Два рівні взаємодіють через Engine API.

Типи вузлів мережі Ethereum:

- Full Node (повний вузол) - зберігає останні N блоків, самостійно перевіряє всі транзакції та стан мережі. Розмір ~1-2 ТБ;

- Archive Node (архівний вузол) - зберігає повну історію всіх станів з початку мережі (> 12 ТБ). Використовується для аналітики;

- Light Node (легкий вузол) - зберігає лише заголовки блоків, довіряє повним вузлам для верифікації транзакцій;

- Validator Node (вузол-валідатор) - бере участь у консенсусі PoS, потребує щонайменше 32 ETH у заставі.

Акаунти в Ethereum бувають двох типів. Зовнішні акаунти (Externally Owned Accounts, EOA) керуються парою приватний/публічний ключ та можуть ініціювати транзакції. Акаунти контрактів (Contract Accounts) містять код смарт-контракту та можуть виконувати транзакції лише у відповідь на виклики. Стан мережі Ethereum (world state) представляє відображення адрес (20-байтних публічних ключів) на стани акаунтів. Стан акаунту включає: nonce, balance (баланс в wei), storageRoot (хеш сховища контракту у вигляді Patricia Merkle Trie), codeHash (хеш коду контракту, порожній для EOA).

Структура транзакції Ethereum включає обов'язкові поля, що повністю описують передачу цінності або виклик контракту:

1. nonce - порядковий номер транзакції відправника (починаючи з 0); запобігає повторним атакам (replay attacks);
2. maxFeePerGas та maxPriorityFeePerGas - максимальна ціна газу та чайові (після EIP-1559, з серпня 2021);
3. gasLimit - максимальна кількість газу; проста передача ETH вимагає рівно 21,000 одиниць;

4. to - адреса отримувача або смарт-контракту; значення null означає деплоймент нового контракту;
5. value - кількість ETH у wei (1 ETH = 10^18 wei; 1 gwei = 10^9 wei);
6. data - ABI-закодовані вхідні дані для виклику смарт-контракту (порожні для простих переказів ETH);
7. v, r, s - компоненти ECDSA-підпису відправника на кривій secp256k1.

Транзакції в мережі Ethereum поділяються на кілька категорій залежно від їх призначення та механізму виконання. Розуміння цієї класифікації є критично важливим для коректної фільтрації whale-транзакцій у розробленій системі.

Прості перекази ETH (EOA-to-EOA transfers) є найпростішим типом транзакцій: відправник переказує певну кількість ETH на адресу отримувача без виконання коду. Характерні ознаки: поле data є порожнім (0x), gasLimit дорівнює рівно 21 000 одиниць газу, поле to містить адресу зовнішнього акаунту (EOA), а не контракту. Саме цей тип транзакцій є основним об'єктом моніторингу розробленої системи, оскільки відображає прямі переміщення капіталу між гравцями ринку.

Виклики смарт-контрактів (contract calls) виникають коли EOA взаємодіє з розгорнутим контрактом. Поле to містить адресу контракту, поле data містить ABI-закодований селектор функції (перші 4 байти Keccak-256 хешу сигнатури функції) та аргументи виклику. Значення поля value може бути як нульовим (виклик без передачі ETH, наприклад approve токену), так і ненульовим (виклик з одночасним переказом ETH, наприклад додавання ліквідності у DeFi-протокол). Розроблена система фільтрує транзакції виключно за полем value, тому виклики контрактів з передачею ETH $\geq$ 10 ETH також потрапляють до моніторингу - це охоплює великі DeFi-операції таких протоколів як Uniswap, Aave та Lido.

Деплоймент контрактів (contract creation) є особливим типом транзакцій, де поле to має значення null (None в Python). EVM інтерпретує це як команду розгорнути новий контракт: виконується ініціалізаційний код з поля data, після чого байт-код контракту записується за новою адресою, що обчислюється як Keccak-256(sender_address + nonce). Розроблена система явно виключає цей тип

через перевірку to_addr is not None, оскільки такі транзакції не відображають переміщення капіталу між гравцями.

Internal transactions (внутрішні транзакції) є принципово відмінним явищем - це виклики між смарт-контрактами, що відбуваються всередині виконання основної транзакції. Наприклад, коли користувач викликає агрегатор Uniswap, той в свою чергу викликає кілька пулів ліквідності - кожен такий виклик є internal transaction. Вони не є самостійними транзакціями в блокчейні та не мають власного хешу - існує лише хеш батьківської транзакції. Internal transactions не доступні через стандартний JSON-RPC метод eth_getBlockByNumber та вимагають окремого виклику debug_traceTransaction або використання спеціалізованих індексуючих сервісів на кшталт Etherscan Internal Transactions API. Розроблена система не відстежує internal transactions у поточній реалізації, що є одним з напрямків подальшого розвитку. Практичний наслідок цієї класифікації для системи моніторингу полягає в тому, що реальний обсяг переміщення капіталу великим гравцем може суттєво відрізнятись від значення поля value основної транзакції. Наприклад, транзакція виклику агрегатора з value=0 ETH може ініціювати internal transactions на мільйони доларів через серію викликів DeFi-протоколів. Це обмеження є характерним для всіх систем моніторингу на базі публічного JSON-RPC API та підкреслює важливість розуміння архітектури Ethereum при інтерпретації аналітичних даних дашборду.

Ідентифікація транзакцій здійснюється через хеш (txHash) - результат застосування алгоритму Keccak-256 до RLP-закодованих полів транзакції. Хеш є унікальним 32-байтним ідентифікатором, що записується у шістнадцятковому форматі з префіксом 0x. Саме він є первинним ключем у базі даних системи моніторингу.

Для доступу до даних блокчейну застосовуються вузлові провайдери - сервіси, що надають API для взаємодії з Ethereum. Серед найпоширеніших: Infura, Alchemy, QuickNode та публічний безкоштовний вузол publicnode.com, що використовується в даній роботі. Перевагою publicnode.com є відсутність необхідності реєстрації та API-ключів.

Ethereum JSON-RPC API (Рис. 1.2) надає стандартний набір методів для взаємодії з мережею. Для системи моніторингу ключовими є: eth_subscribe newHeads - підписка на нові блоки через WebSocket; eth_getBlockByNumber - отримання повного вмісту блоку з усіма транзакціями; eth_getTransactionByHash - отримання деталей транзакції за хешем.

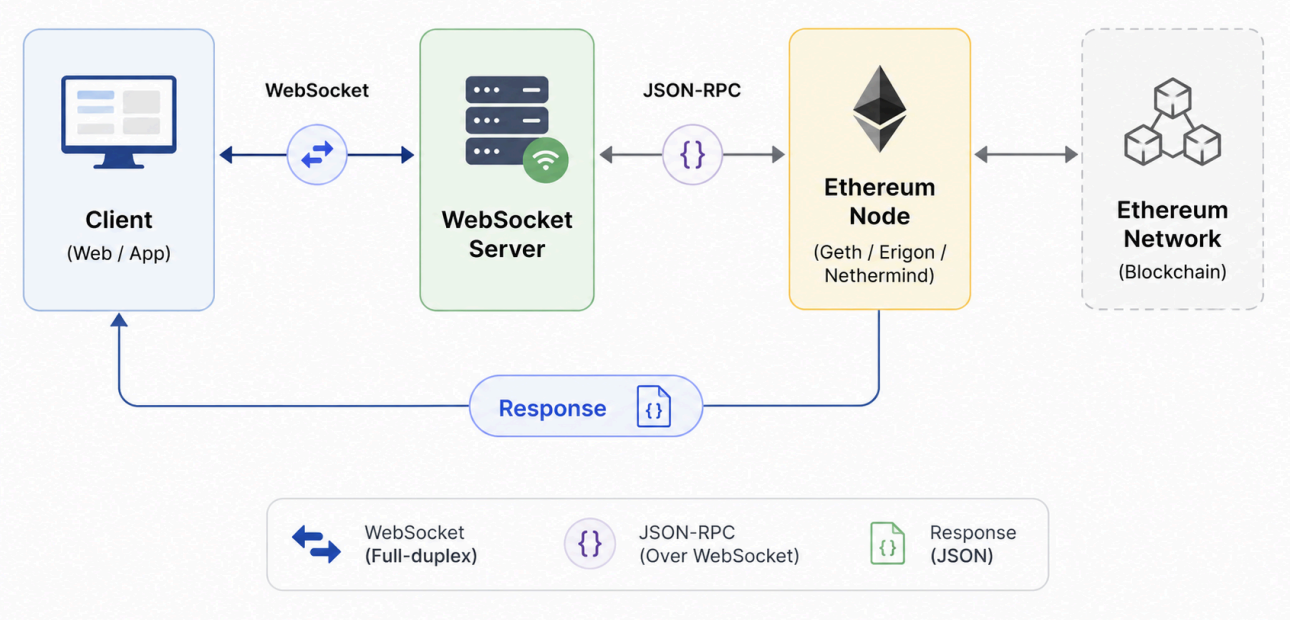


Рис. 1.2 Взаємодія компонентів системи з мережею Ethereum через JSON-RPC

WebSocket-підписка eth_subscribe newHeads є ключовим механізмом для систем моніторингу в реальному часі. На відміну від HTTP polling (постійного опитування сервера), WebSocket встановлює постійне двостороннє з'єднання та миттєво доставляє повідомлення про нові блоки. Нові блоки в мережі Ethereum після The Merge з'являються кожні 12 секунд (один slot), що є постійним та передбачуваним інтервалом. Кожен блок Ethereum може містити до декількох сотень транзакцій. Газовий ліміт блоку (зазвичай 15-30 мільйонів одиниць газу) обмежує кількість транзакцій. Проста транзакція переказу ETH вартує 21,000 одиниць газу, тому теоретично в один блок може поміститись ~700-1,400 простих переказів. На практиці блок містить суміш простих переказів та складних DeFi-транзакцій, тому в середньому він включає 100-300 транзакцій.

1.3 Поняття «кит» (whale) у криптовалютних мережах

Термін «кит» (whale) у контексті криптовалютних ринків позначає учасників, що переміщують значні обсяги криптовалюти. Великі транзакції можуть здійснюватись криптовалютними біржами, інституційними інвесторами, хедж-фондами, протоколами DeFi, а також анонімними великими гравцями. Класифікація учасників крипторинку за розміром позиції (на прикладі Ethereum):

- дрібні інвестори (retail): менше 1 ETH - найбільша за кількістю, але найменша за обсягом група;
- малі інвестори: 1-10 ETH - типові роздрібні учасники ринку;
- середні гравці: 10-100 ETH - досвідчені трейдери та невеликі фонди;
- «дельфіни»: 100-1000 ETH - значні тримачі, малі інституції;
- «кити»: 1000-10000 ETH - великі інституції та фонди (~25,000-250,000 USD при ETH $25);
- «мегакити»: більше 10000 ETH - криптовалютні біржі, венчурні фонди.

Відстеження активності китів є важливим інструментом ринкового аналізу з кількох причин. Великі переміщення коштів часто передують значним ціновим коливанням, оскільки продаж або купівля великих обсягів впливає на ліквідність ринку. Аналіз адрес та патернів транзакцій дозволяє виявляти концентрацію коштів та потенційні маніпуляції ринком. Дослідження мережевої структури транзакцій Ethereum показують, що більшість токенових мереж концентруються навколо вузлових адрес за принципом «зірки», що є характерною ознакою активності великих гравців ринку [22]. Для ідентифікації whale-адрес аналітики застосовують методи кластеризації - об'єднання адрес, що ймовірно належать одній організації, на основі спільних патернів витрат або спільного джерела фінансування. Додатково використовується аналіз часових патернів: адреси, що здійснюють транзакції з характерними інтервалами, можуть належати автоматизованим торговим системам одного гравця. Публічно відомі адреси великих бірж (Binance, Coinbase, Kraken) дозволяють одразу класифікувати значну

частину великих транзакцій. Дослідники відзначають кілька характерних патернів whale-активності та їх ринкові наслідки:

1. Переміщення ETH з холодних гаманців на гарячі (біржові) адреси традиційно розглядається як ведмежий сигнал - whale готується до продажу;
2. Переміщення з біржових адрес на холодні гаманці або стейкінг-контракти - бичачий сигнал, whale виводить кошти з обороту;
3. Великі транзакції між невідомими адресами можуть означати позабіржові (OTC) угоди між великими гравцями;
4. Серія транзакцій з однієї адреси за короткий проміжок часу може вказувати на «розбиття» великої позиції для мінімізації ринкового впливу.

У реальних ринкових умовах кореляція між whale-активністю та ціновими рухами підтверджується дослідженнями. Перед різким падінням ціни ETH у травні 2021 року спостерігалась значна активність великих адрес з переміщення коштів на біржі. Аналогічна картина передувала обвалу листопада 2022 року. Водночас, важливо розуміти, що кореляція не означає причинно-наслідкового зв'язку[4].

В рамках даної роботи мінімальним порогом whale-транзакції для збереження в базу даних встановлено 10 ETH. При поточній ціні ETH ~2500 USD транзакція на 10 ETH еквівалентна ~25,000 USD, що суттєво перевищує середній обсяг роздрібної транзакції (~0.05 ETH ≈ 125 USD). Система дозволяє кожному користувачу налаштовувати індивідуальний поріг від 1 до 1,000,000 ETH. Ідентифікація відомих whale-адрес є окремим напрямком аналітики. Великі криптовалютні біржі (Binance, Coinbase, Kraken) мають публічно відомі адреси, транзакції з яких легко ідентифікувати. Більш складним є відстеження гаманців анонімних великих гравців, для чого використовуються методи кластеризації адрес та аналіз патернів транзакцій - ці можливості реалізовані у вигляді мережевого графу зв'язків у вебдашборді розробленої системи. Ідентифікація whale-адрес є окремим напрямком on-chain аналітики, що використовує кілька методологічних підходів. Найпоширенішим є метод кластеризації адрес -

об'єднання множини адрес, що з високою ймовірністю належать одному учаснику, на основі аналізу патернів транзакцій.

Евристика спільного входу (Common Input Ownership Heuristic, CIOH) базується на припущенні, що всі адреси-відправники однієї транзакції контролюються одним гравцем, оскільки для підписання транзакції необхідний доступ до всіх приватних ключів одночасно. В мережі Ethereum аналогічна логіка застосовується до адрес, що систематично фінансують одні й ті самі смарт-контракти або отримують кошти з одного джерела. Аналіз часових патернів дозволяє ідентифікувати автоматизовані торгові системи - адреси, що здійснюють транзакції з характерними часовими інтервалами або реагують на певні ринкові події із сталою затримкою. Такі патерни свідчать про роботу алгоритмічних трейдерів або арбітражних ботів, що є окремою категорією великих гравців ринку. Метод аналізу пилових транзакцій (dust analysis) використовується у зворотному напрямку: відправляючи мікроскопічні суми на велику кількість адрес та відстежуючи куди ці кошти переміщуються далі, аналітики можуть встановити зв'язки між адресами, що зовні виглядають непов'язаними. Хоча цей метод є більш характерним для Bitcoin, в мережі Ethereum він також застосовується для деанонімізації великих гравців.

Публічно відомі адреси великих криптовалютних бірж (Binance, Coinbase, Kraken) формують окрему категорію - так звані labeled addresses. Організації Chainalysis, Elliptic та Nansen формують бази даних таких адрес шляхом комбінування on-chain аналізу та off-chain даних: реєстраційних документів бірж, публічних заяв компаній, інформації з KYC-процедур. Розроблена в даній роботі система використовує базовий підхід до ідентифікації - мережевий граф зв'язків між адресами, що дозволяє візуально виявляти кластери пов'язаних адрес без необхідності доступу до комерційних баз даних [22].

Щоденна статистика whale-транзакцій (Рис. 1.3) у мережі Ethereum, зафіксована системою моніторингу за 2 місяці спостереження:

- транзакцій >= 10 ETH: від 2,000 до 5,000 на день (близько 150-200 на блок);

- транзакцій >= 100 ETH: від 400 до 1,000 на день;
- транзакцій >= 1,000 ETH: від 10 до 200 на день;
- транзакцій >= 10,000 ETH: від 3 до 20 на день (еквівалент від 25 млн USD).

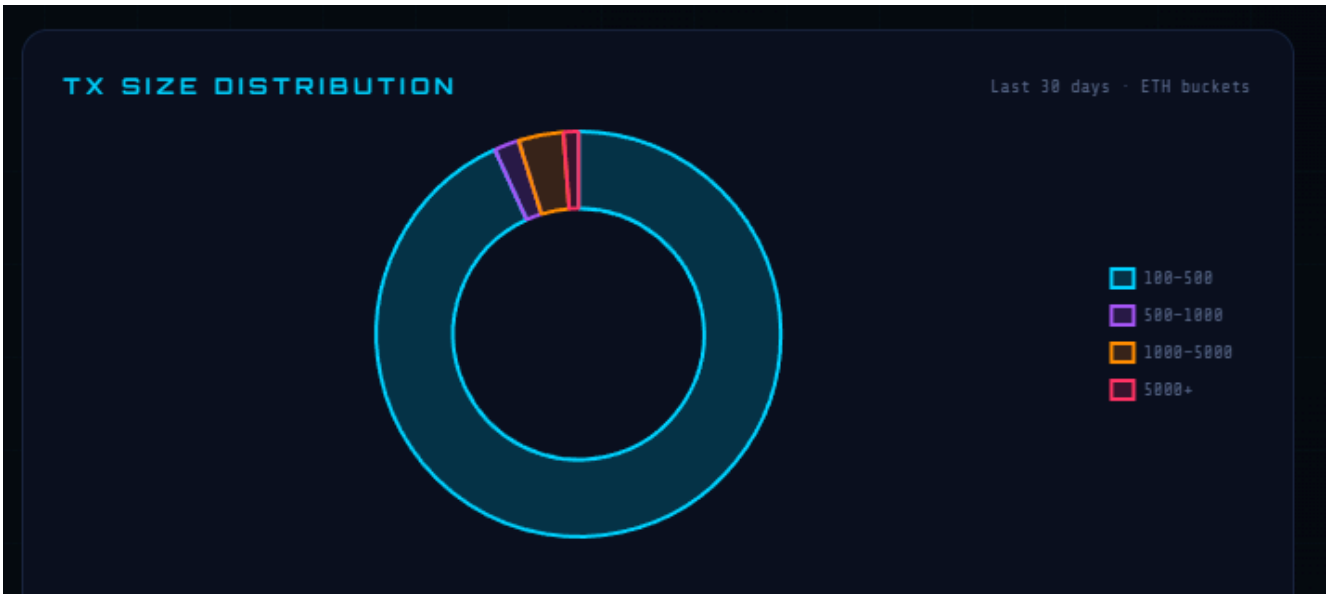


Рис. 1.3 Щоденна кількість whale-транзакцій Ethereum за розміром

1.4 Аналіз існуючих систем моніторингу

Перед розробкою системи проведено детальний аналіз існуючих рішень у сфері моніторингу блокчейн-транзакцій. Розглянуто три найбільш поширені інструменти: Etherscan, Nansen та Whale Alert. Аналіз проводився за критеріями: вартість використання, наявність push-сповіщень, можливість персоналізації, функціональність дашборду та доступність API.

Etherscan (etherscan.io) - найпопулярніший провідник блокчейну Ethereum, запущений у 2015 році. Щоденна кількість запитів до сервісу перевищує 1 млрд. Etherscan є фактичним стандартним інструментом для перегляду даних Ethereum у галузі[5]. Функціональні можливості Etherscan включають: детальний перегляд транзакцій, блоків та адрес; аналітику газових цін в реальному часі; верифікацію вихідного коду смарт-контрактів; DEX-трекер для відстеження угод на децентралізованих біржах; токен-аналітику для ERC-20 та NFT. Інструмент також

надає базовий безкоштовний API для розробників з обмеженням 3 запитів/секунду.

Ключові обмеження Etherscan для завдань системи сповіщень:

1. Відсутність push-сповіщень у месенджери: користувач повинен самостійно відкривати сайт та перевіряти транзакції;
2. Платний API: для більшого за 3 запитів/сек доступу необхідна підписка Pro від $49/місяць;
3. Відсутність фільтрації за розміром транзакції: неможливо побачити тільки whale-транзакції;
4. Відсутність інтеграції з Telegram або іншими месенджерами;
5. Відсутність агрегованої аналітики: немає графіків whale-активності або leaderboard відправників.

Nansen (nansen.ai) - платформа блокчейн-аналітики з акцентом на «розумні гроші» (smart money). Заснована у 2020 році, залучила $75M інвестицій у серії B у 2022 році при оцінці $750M. Маркує понад 250 мільйонів адрес у 30+ блокчейнах[6]. Ключові особливості Nansen: маркування «розумних грошей» (smart money labeling) для виявлення адрес з доведеним успіхом у торгівлі; Token God Mode - аналіз власників будь-якого токена; Wallet Profiler - детальний аналіз будь-якої адреси; NFT analytics; сигнали про whale-активність з автоматичними тегами відомих адрес. Обмеження Nansen: вартість підписки від $150/місяць (Starter) до $1,500+/місяць (Pro), що робить платформу недоступною для більшості роздрібних інвесторів. Відсутня інтеграція з Telegram для автоматичних push-сповіщень про транзакції в реальному часі. Складний інтерфейс потребує навчання.

Whale Alert (whale-alert.io) - спеціалізований сервіс відстеження великих криптовалютних транзакцій, заснований у 2019 році. Має понад 1.5 мільйони підписників у Twitter та Telegram-канал з 500,000+ учасників[7]. Функціональність Whale Alert: відстеження транзакцій у Bitcoin, Ethereum, Ripple, Tron та інших мережах; ідентифікація відомих адрес (Binance, Coinbase, Kraken та ін.); автоматичні публікації у Twitter та Telegram-каналі; базовий веб-інтерфейс з

переліком останніх транзакцій; платний API від $99/місяць. Ключовим обмеженням Whale Alert є відсутність персоналізації. Всі підписники Telegram-каналу отримують абсолютно однакові сповіщення незалежно від їх інтересів. Якщо користувача цікавлять тільки транзакції > 1,000 ETH, він все одно отримує сповіщення про транзакції від 500 ETH. Це призводить до «інформаційного шуму» та відмови від підписки. Порівняльний аналіз розглянутих систем моніторингу за ключовими критеріями представлено в таблиці 1.1.

Таблиця 1.1

Порівняльний аналіз систем моніторингу блокчейн-транзакцій

Критерій	**Etherscan**	**Nansen**	**Whale Alert**	**Whale Tracker**
Вартість	Безкоштовно / $199+/міс	$150-$1500+/міс	$5-99+/міс	Безкоштовно
Telegram сповіщення	Ні	Ні	Так (без персон.)	Так (персональні)
Індивід. поріг	Ні	Частково	Ні	Так (1-1M ETH)
Вебдашборд	Частково	Так	Ні	Так (повний)
Historical аналітика	Так	Так	Ні	Так (MongoDB)
Реальний час	Частково	Так	Так	Так
Профілювання адрес	Так	Так	Частково	Так (базове)

Таким чином, жодне з розглянутих рішень не поєднує безкоштовність, персоналізацію порогів для кожного користувача та повноцінний інтерактивний вебдашборд з historical аналітикою. Це підтверджує актуальність розробкисистеми з даним набором функцій.

1.5 Огляд технологій для розробки системи

Для реалізації системи обрано технологічний стек, що забезпечує асинхронну обробку даних, масштабованість та зручність розгортання. Вибір

кожного компонента ґрунтувався на порівняльному аналізі альтернатив з урахуванням специфіки задачі.

Обрану основну мову програмування Python, завдяки широкій екосистемі бібліотек для роботи з блокчейном (Web3), підтримці asyncio для асинхронного програмування, та активній спільноті. Альтернативно розглядався NodeJS з бібліотекою ethers.js: він має схожу асинхронну модель та хорошу підтримку Ethereum. Перевагу Python забезпечили зрілість бібліотеки Web3 для нашої версії Ethereum API та більший досвід з цією мовою.

Бібліотека Web3 надає повний набір інструментів для взаємодії з Ethereum: підключення через HTTP та WebSocket провайдери, читання даних блокчейну, конвертація між форматами адрес та одиниць (wei, gwei, ether), підпис транзакцій.[8].

asyncio - вбудована Python-бібліотека для асинхронного програмування через coroutines та event loop. Вона дозволяє виконувати кілька I/O операцій одночасно без блокування потоку виконання. Це критично важливо для системи, що одночасно слухає WebSocket з’єднання, відповідає на HTTP API запити та виконує запити до MongoDB[13].

websockets - асинхронна бібліотека для WebSocket протоколу. Забезпечує встановлення та підтримку WebSocket з'єднання з вузлом Ethereum, автоматичне надсилання ping-повідомлень для виявлення розривів з’єднання, обробку помилок з’єднання[12].

FastAPI - сучасний асинхронний Python-веб-фреймворк, що базується на стандартних анотаціях типів Python 3.6+. Автоматично генерує OpenAPI (Swagger) та ReDoc документацію, валідує вхідні та вихідні дані через Pydantic, підтримує Dependency Injection. За бенчмарками TechEmpower FastAPI входить до топ-5 найшвидших Python-фреймворків[9].

Ключові переваги FastAPI для даного проєкту:

1. Нативна підтримка asyncio - не блокує event loop при обробці HTTP-запитів;
2. Автоматична Swagger UI документація за адресою /docs;

3. Вбудована підтримка CORS (Cross-Origin Resource Sharing) для вебдашборду;
4. Вбудована підтримка StaticFiles для роздачі HTML/CSS/JS файлів дашборду.

Uvicorn - ASGI (Asynchronous Server Gateway Interface) веб-сервер. Забезпечує виконання FastAPI застосунку та підтримує HTTP/1.1, WebSocket та HTTP/2. В даній роботі Uvicorn запускається програмно через uvicorn.Server всередині основного asyncio event loop разом з Telegram-ботом та Whale Tracker.

Telegram Bot API - офіційний HTTP API для роботи з Telegram-ботами. Надає можливість надсилати та отримувати повідомлення, управляти клавіатурами, вебхуками та inline запитами. Telegram забезпечує безкоштовний та надійний канал для push-сповіщень з аудиторією понад 900 мільйонів активних користувачів станом на 2024 рік.

aiogram - асинхронний фреймворк для Telegram-ботів на Python. Реалізує Router-архітектуру для маршрутизації повідомлень, FSM (Finite State Machine) для управління станами розмови через MemoryStorage або Redis, Middleware для перехоплення та обробки запитів. Повністю асинхронний, всі операції виконуються через asyncio без блокування event loop[10].

Порівняння з альтернативами: python-telegram-bot (використовує threading замість asyncio, складніший для інтеграції з нашою архітектурою), telebot (простіший API, але менш функціональний для FSM). aiogram обрано як найбільш зрілий та активно підтримуваний асинхронний фреймворк для Telegram.

MongoDB - документно-орієнтована NoSQL база даних, що зберігає дані у форматі BSON (Binary JSON). Обрана для зберігання транзакцій завдяки гнучкій схемі документів (можна зберігати додаткові поля без міграцій), потужному aggregation pipeline для складних аналітичних запитів та горизонтальному масштабуванню через sharding[11]. Порівняння з PostgreSQL: PostgreSQL забезпечує ACID-гарантії та складні JOIN-запити, але для даного проєкту JOIN не потрібні (всі дані транзакції зберігаються в одному документі). MongoDB виграє в продуктивності aggregation pipeline для time-series аналітики та у гнучкості схеми.

Для майбутньої підтримки схеми транзакцій різних токенів (ERC-20) гнучкість MongoDB є особливо важливою.

Motor - асинхронний Python-драйвер для MongoDB. Дозволяє виконувати будь-які MongoDB операції (CRUD, aggregation pipeline, індекси) без блокування asyncio event loop. Підтримує connection pooling та автоматичне перепідключення при розриві[18].

Chart.js (Рис. 1.4) - JavaScript-бібліотека з відкритим кодом для побудови інтерактивних графіків на HTML5 Canvas. Підтримує 8 вбудованих типів графіків (bar, line, doughnut, scatter тощо), адаптивний дизайн, анімації, плагіни та налаштування tooltips. Обрана завдяки простоті налаштування, активній підтримці та відмінній документації[14].

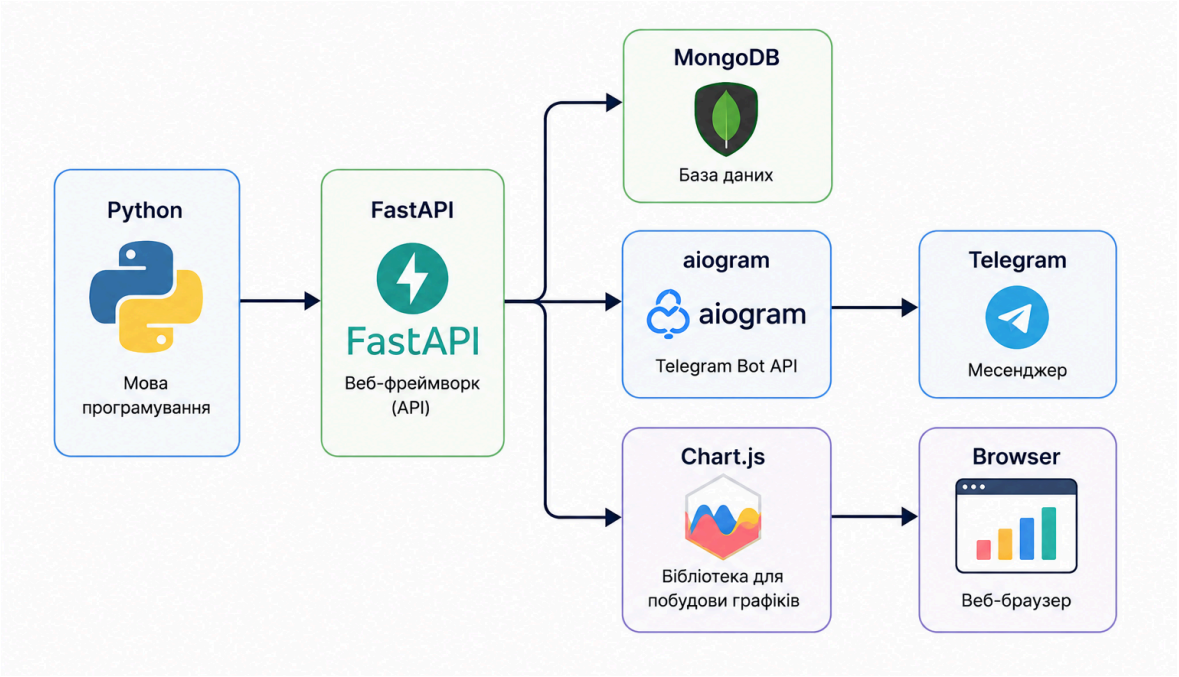


Рис. 1.4 Технологічний стек та компоненти взаємодії системи Whale Tracker

D3.js - потужна JavaScript-бібліотека для маніпулювання DOM на основі даних. Забезпечує повний контроль над SVG-елементами, підтримує фізичну симуляцію для мережевих графів (Рис. 1.5) (d3.forceSimulation), складні переходи та анімації. D3.js використовується для двох компонентів: теплової карти активності за GitHub-стилем та мережевого графу зв'язків між whale-адресами[15].

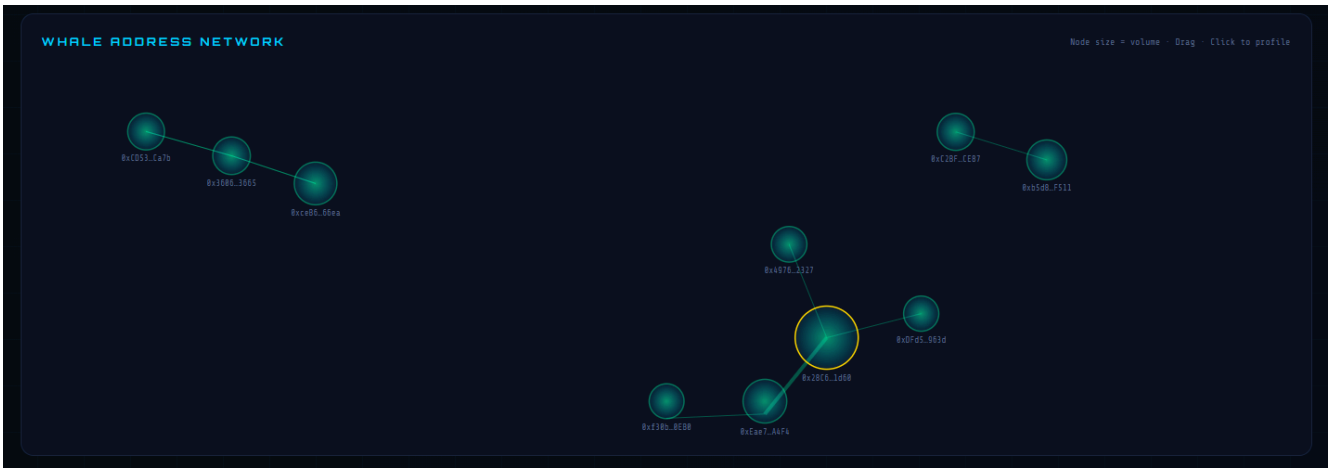


Рис. 1.5 Скріншот мережевого графу зв'язків між whale-адресами у веб-дашборді

Nginx - високопродуктивний веб-сервер та зворотній проксі-сервер з відкритим кодом. Використовується для SSL термінації (розшифрування HTTPS трафіку) та проксіювання HTTP-запитів до FastAPI на порту 8000. Nginx ефективно обробляє до 10,000 одночасних з'єднань, що забезпечує запас для майбутнього масштабування системи[16].

Let's Encrypt - безкоштовний, автоматизований центр сертифікації SSL/TLS, заснований у 2016 році як частина Internet Security Research Group (ISRG). Надає X.509 SSL-сертифікати безкоштовно через протокол ACME. Certbot - офіційний клієнт для автоматичного отримання та оновлення сертифікатів кожні 60-90 днів[17].

Ubuntu обрано як операційну систему VPS-сервера завдяки тривалій підтримці до квітня 2027 року, широкій доступності документації, стабільним офіційним пакетам MongoDB 7.0 та Python 3.11. Systemd забезпечує автоматичний запуск та перезапуск сервісів, що є критично важливим для забезпечення безперервної роботи системи моніторингу[20].

Ключовою архітектурною перевагою обраного стеку є спільний asyncio event loop, в якому одночасно працюють три незалежні компоненти: Whale Tracker, FastAPI-сервер та Telegram-бот. Це усуває необхідність міжпроцесної комунікації та зменшує затримку між виявленням whale-транзакції та надсиланням сповіщення користувачу. Порівняно з багатопотоковою або

мультипроцесною архітектурою, асинхронний підхід забезпечує нижче споживання пам'яті та вищу стабільність під навантаженням.

Таким чином, обраний технологічний стек на основі Python, FastAPI, aiogram, MongoDB та Nginx забезпечує повністю асинхронну, масштабовану та економічно ефективну архітектуру для реалізації системи моніторингу транзакцій на блокчейні Ethereum.

РОЗДІЛ 2. АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ СИСТЕМИ МОНІТОРИНГУ

2.1 Загальна архітектура та проєктні рішення

Whale Tracker (Рис. 2.1) являє собою трикомпонентну асинхронну архітектуру, компоненти якої працюють паралельно в рамках єдиного Python-процесу з використанням asyncio event loop. Таке проєктне рішення дозволяє уникнути накладних витрат на міжпроцесну комунікацію та спрощує розгортання системи на одному сервері.

Три основних компоненти системи та їх функції:

1. Whale Tracker - відстежує нові блоки Ethereum через WebSocket, фільтрує whale-транзакції, зберігає в MongoDB та надсилає сповіщення через Telegram API;
2. Telegram Bot - реєструє користувачів, приймає команди налаштування порогу, надсилає персоналізовані сповіщення про whale-транзакції;
3. FastAPI Dashboard - надає REST API для вебдашборду, виконує агрегаційні запити до MongoDB, роздає статичні файли HTML/CSS/JS.

Усі три компоненти взаємодіють через спільну базу даних MongoDB. Whale Tracker записує транзакції в колекцію transactions та читає профілі користувачів з колекції user_profiles для надсилання сповіщень. Telegram Bot читає та оновлює колекцію user_profiles. FastAPI читає колекцію transactions для аналітичних запитів. Точка входу системи - файл main.py, що одночасно запускає три асинхронні задачі: asyncio.create_task(watch_blocks(bot)) для Whale Tracker, asyncio.create_task(server.serve()) для FastAPI/Uvicorn та await dp.start_polling(bot) для Telegram polling. Функція start_polling блокує основний потік виконання, підтримуючи все в активному стані.

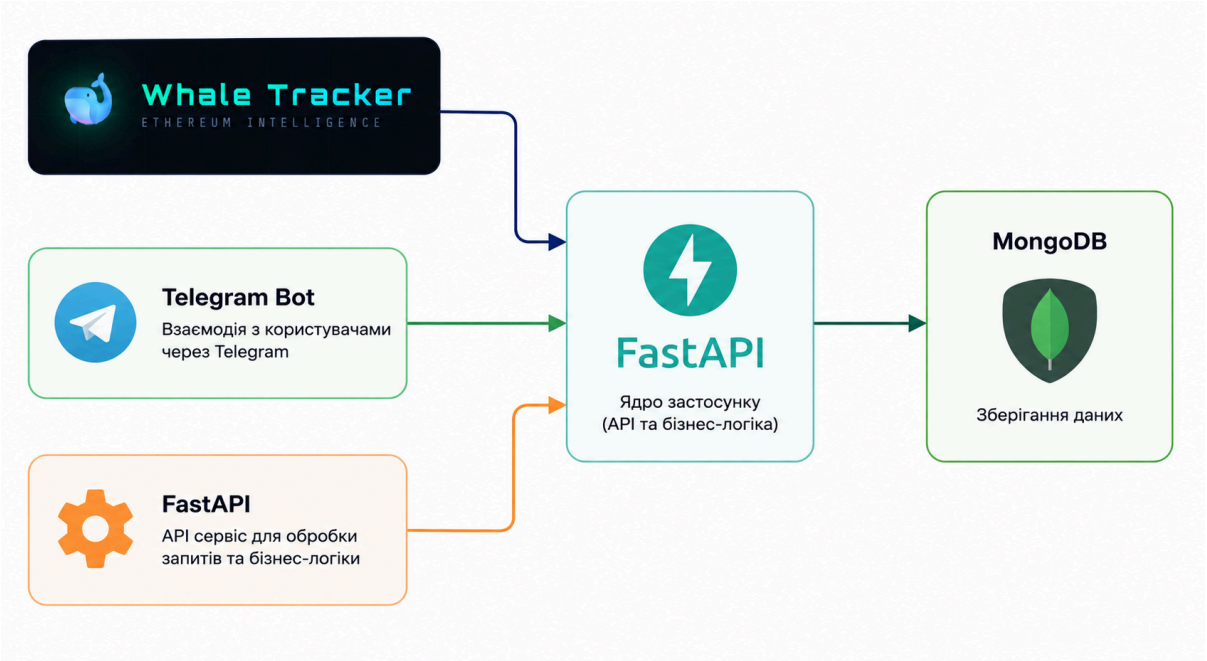


Рис. 2.1 Загальна архітектура системи моніторингу Whale Tracker

Структура файлів проєкту організована за принципом модульності:

- main.py - точка входу, ініціалізація всіх компонентів;
- config.py - конфігурація з ENV-змінних (токен бота, URI MongoDB);
- logger.py - логування для всіх модулів;
- blockchain/watcher.py - WebSocket слухач, обробка блоків та транзакцій;
- bot/handlers.py - aiogram обробники команд (/start, /limit) та FSM;
- bot/keyboards.py - ReplyKeyboard та InlineKeyboard клавіатури;
- bot/etherscan.py - генератор InlineKeyboard з посиланнями на Etherscan;
- db/mongo.py - підключення до MongoDB та ініціалізація колекцій;
- api.py - FastAPI застосунок з 13 REST API ендпоінтами;
- static/index.html - SPA вебдашборд;
- static/css/styles.css - стилі дашборду;
- static/js/ - ES-модулі: main.js, charts.js, data.js, utils.js та ін.

Ключові проєктні рішення, що забезпечують надійність системи.

Перше рішення - єдиний asyncio event loop: Whale Tracker, Telegram polling та Uvicorn server виконуються кооперативно без блокування один одного. Це

означає, що затримка в одному компоненті (наприклад, повільна MongoDB-операція) не блокує інші компоненти.

Друге рішення - upsert замість insert при збереженні транзакцій. Операція upsert з $setOnInsert записує транзакцію тільки якщо вона ще не існує в БД. Це захищає від дублікатів при повторній обробці блоків, що може статися при перепідключенні WebSocket або перезапуску системи. Гарантується ідемпотентність збереження.

Третє рішення - Systemd як менеджер процесів. Конфігурація Restart=always та RestartSec=5 забезпечує автоматичний перезапуск при будь-якій помилці. WantedBy=multi-user.target гарантує автоматичний старт після перезавантаження сервера. Requires=mongod.service гарантує, що MongoDB запущений перед стартом системи.

Четверте рішення - розділення WebSocket (WSS) та HTTP провайдерів. WebSocket використовується тільки для підписки на нові блоки (eth_subscribe), а завантаження повного вмісту блоку виконується через HTTP (eth_getBlockByNumber). Це оптимізує навантаження та спрощує обробку помилок.

Архітектурні рішення, реалізовані в системі Whale Tracker, забезпечують оптимальний баланс між продуктивністю, надійністю та простотою розгортання. Єдиний asyncio event loop усуває накладні витрати на міжпроцесну комунікацію та гарантує мінімальну затримку між виявленням whale-транзакції та надсиланням сповіщення користувачу. Модульна структура проєкту - з чітким розділенням відповідальності між Whale Tracker, Telegram-ботом та FastAPIдашбордом - спрощує супровід і майбутнє масштабування системи. Використання операції upsert замість звичайного insert забезпечує ідемпотентність збереження даних і захищає від дублікатів при нештатних ситуаціях, таких як обрив WebSocket-з'єднання або перезапуск процесу. Розділення WebSocket та HTTP-провайдерів оптимізує мережеве навантаження: підписка на нові блоки здійснюється через постійне WSS-з'єднання, тоді як ресурсомісткі запити вмісту блоків виконуються через HTTP. Інтеграція із systemd з параметрами

Restart=always та Requires=mongod.service перетворює систему на повноцінний керований сервіс, що автоматично відновлюється після збоїв і коректно запускається разом з операційною системою. Сукупність цих рішень дозволяє розгорнути всі три компоненти системи в рамках одного Python-процесу на бюджетному VPS-сервері, зберігаючи при цьому продуктивність та стійкість, порівнянну з комерційними аналогами.

2.2 Модуль відстеження блокчейн-транзакцій

Модуль відстеження транзакцій реалізовано у файлі blockchain/watcher.py та є ключовим компонентом системи. Алгоритм роботи модуля складається з семи послідовних етапів, що відображені на блок-схемі (Рис. 2.2).

Етап 1. Підписка на нові блоки. Після старту системи модуль встановлює постійне WebSocket-з'єднання з вузлом Ethereum та надсилає JSON-RPC запит методом eth_subscribe з параметром newHeads. Підписка забезпечує отримання push-повідомлень про кожен новий блок без необхідності циклічного опитування вузла. Для забезпечення стійкості роботи реалізовано зовнішній цикл автоматичного відновлення з'єднання: при будь-якій мережевій помилці система логує подію та через 5 секунд повторює спробу підключення.

Етап 2. Отримання нового блоку. Кожне вхідне WebSocket-повідомлення розбирається як JSON-об'єкт. З поля params.result.number витягується номер нового блоку у шістнадцятковому форматі та конвертується у десяткове ціле число для подальшої обробки.

Етап 3. Завантаження вмісту блоку. Функція process_block завантажує повний вміст блоку разом з усіма транзакціями через окремий HTTP-запит eth_getBlockByNumber з параметром full_transactions=True. Використання окремого AsyncHTTPProvider для цього кроку є свідомим архітектурним рішенням: WebSocket оптимальний для підписки на події, тоді як HTTP надійніший для завантаження великих об'єктів даних. Кількість транзакцій у блоці в середньому становить 100–300.

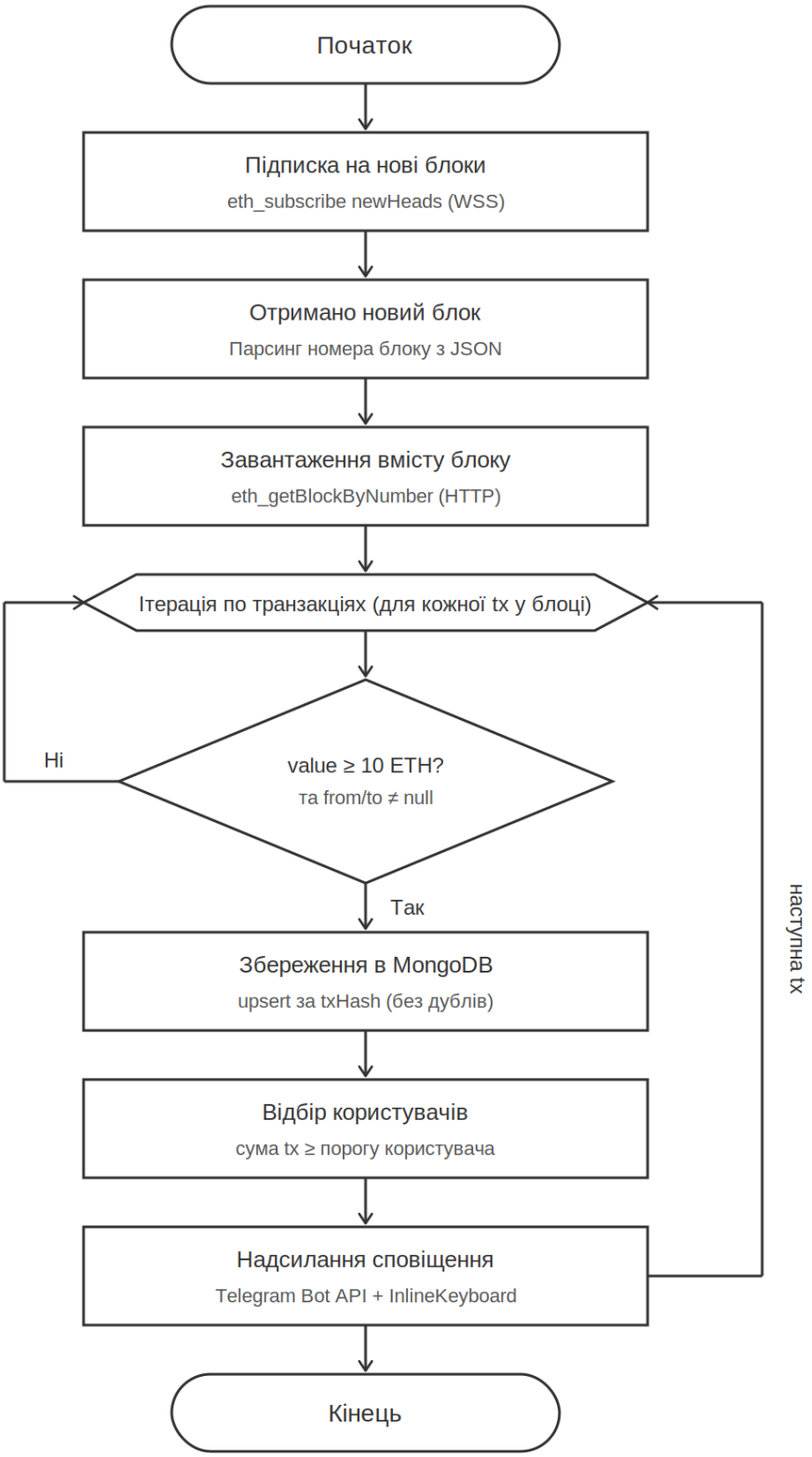


Рис. 2.2 Блок-схема алгоритму роботи модуля

Етап 4. Фільтрація транзакцій. Для кожної транзакції блоку перевіряються три умови: сума переказу не менша за мінімальний поріг у 10 ETH (у wei), поле

отримувача не є порожнім (виключення деплойменту смарт-контрактів), поле відправника не є порожнім. Транзакції, що не відповідають хоча б одній умові, пропускаються і цикл переходить до наступної транзакції.

Етап 5. Збереження в MongoDB. Відфільтровані транзакції зберігаються в колекцію transactions (Рис. 2.3) через асинхронну операцію upsert з ключем txHash. Перед збереженням хеш транзакції нормалізується функцією extract_hash до стандартного формату 0x-prefixed lowercase hex - це усуває проблему дублікатів, що виникала через непослідовний формат хешів від різних RPC-провайдерів. Помилки збереження логуються, але не переривають обробку наступних транзакцій.

Рис. 2.3 Відфільтровані транзакції в колекції transactions

Під час розробки модуля Whale Tracker було виявлено практичну проблему, характерну для систем, що працюють з публічними RPC-вузлами Ethereum: непослідовність формату хешів транзакцій від різних провайдерів. Функція extract_hash була розроблена для вирішення цієї проблеми та є прикладом захисного програмування в умовах ненадійних зовнішніх джерел даних. Стандарт Ethereum визначає хеш транзакції як 32-байтне число у шістнадцятковому форматі з префіксом 0x, наприклад: 0xabc123...def456. Однак на практиці різні RPC-провайдери повертають хеш у різних форматах залежно від реалізації вузла та версії клієнта. Виявлено три основних варіанти: по-перше, bytes-об'єкт Python (b'\xab\xc1\x23...') - характерний для старіших версій web3.py; по-друге, рядок з одним префіксом 0x ('0xabc123...') - стандартний очікуваний формат; по-третє, рядок з подвійним префіксом '0x0xabc123...' - артефакт серіалізації bytes-об'єкта в рядок через str().

Без нормалізації один і той самий хеш транзакції зберігався б у базі даних MongoDB у різних форматах залежно від того, який провайдер його повернув. Це порушувало унікальний індекс колекції transactions та призводило до появи дублікатів. У виробничому середовищі до виправлення баг призводив до того, що окремі транзакції зберігались двічі або тричі, що спотворювало аналітичні дані дашборду - зокрема загальний обсяг ETH та кількість унікальних транзакцій.

Алгоритм функції extract_hash виконує нормалізацію у три кроки. На першому кроці перевіряється тип вхідного значення: якщо це bytes-об'єкт, він конвертується у шістнадцятковий рядок через метод hex(). На другому кроці рядок наводиться до нижнього регістру через lower() для унеможливлення дублікатів через різний регістр символів (0xAbC та 0xabc є одним і тим самим хешем). На третьому кроці виконується видалення подвійного префіксу: якщо рядок починається з '0x0x', зайвий префікс видаляється. Результатом є завжди коректний 0x-prefixed lowercase hex рядок довжиною 66 символів.

Впровадження функції extract_hash повністю усунуло появу дублікатів у виробничому середовищі. За 2 місяці спостереження після виправлення жодного

дублікату в колекції transactions зафіксовано не було, що підтверджується моніторингом унікального індексу через MongoDB Compass.

Етап 6. Відбір користувачів. З колекції user_profiles (Рис. 2.4) вибираються всі користувачі, для яких сума транзакції перевищує або дорівнює їхньому індивідуальному порогу, та які не заблокували бота. Запит використовує фільтр is_blocked: {$ne: True}, а вибірка виконується через асинхронний курсор MongoDB без завантаження всіх документів у пам'ять.

Рис. 2.4 Колекція user_profiles в MongoDB

Етап 7. Надсилання сповіщення. Кожному відібраному користувачу надсилається персоналізоване HTML-повідомлення через Telegram Bot API з деталями транзакції: сума в ETH, час UTC+0, номер блоку, адреси відправника та отримувача у тегах <code> для зручного копіювання, хеш транзакції. До кожного повідомлення додається InlineKeyboardMarkup з кнопками-посиланнями на Etherscan для транзакції та відповідних адрес. Якщо Telegram API повертає помилку «bot was blocked», профіль користувача оновлюється полем is_blocked=True і наступні ітерації автоматично пропускають його. При повторному зверненні користувача до бота командою /start поле скидається до False, відновлюючи надсилання сповіщень.

2.3 Адаптація модуля з використанням Telegram-боту та REST API

Telegram-бот реалізовано у файлах bot/handlers.py, bot/keyboards.py та bot/etherscan.py. Бот зареєстрований через @BotFather з ім'ям

@Eth_WhaleTracker_Bot та підтримує взаємодію з необмеженою кількістю користувачів одночасно завдяки асинхронній архітектурі aiogram.

Кожна взаємодія з ботом починається з виклику асинхронної функції touch_user(user_id, username). Ця функція виконує upsert профілю користувача в MongoDB. При першому зверненні ($setOnInsert) створюється новий профіль з полями: user_id (ідентифікатор Telegram), created_at (поточний час UTC), limit (100.0 ETH за замовчуванням). При повторних зверненнях ($set) оновлюється last_seen та username. Критично важливо, що is_blocked скидається до False при кожному /start, відновлюючи сповіщення після розблокування.

Схема документу user_profiles в MongoDB:

- user_id - Int64, первинний ключ, числовий ідентифікатор Telegram;
- username - String або null, @username в Telegram;
- limit - Float64, поріг сповіщень в ETH (за замовчуванням 100.0);
- created_at - ISODate UTC, момент першої реєстрації;
- last_seen - ISODate UTC, час останньої активності;
- is_blocked - Boolean, True якщо користувач заблокував бота.

Обробник /start виконує три дії: скидає поточний FSM-стан (state.clear()), викликає touch_user для реєстрації/оновлення, надсилає привітальне повідомлення та клавіатуру вибору ліміту. Привітальне повідомлення містить ім'я користувача (username або first_name). Обробник callback-кнопок (filter F.data.startswith("limit_")) обробляє як натискання пресетних кнопок (limit_100, limit_500, limit_1000), так і кнопки «Ввести свій ліміт» (limit_custom). При натисканні пресету одразу встановлюється відповідний ліміт. При limit_custom встановлюється FSM-стан LimitState.waiting_custom_limit. FSM (Finite State Machine) через aiogram FSMContext використовується для очікування числового введення від користувача. Стан waiting_custom_limit зберігається в MemoryStorage між повідомленнями одного користувача. Обробник стану виконує комплексну валідацію введеного числа:

- заміна коми на крапку для підтримки різних регіональних форматів;
- конвертація в float через try-except для обробки нечислових введень;

- перевірка на позитивне значення (> 0);

- перевірка на максимальне значення (< 1,000,000 ETH);

- при невалідному введенні - повідомлення про помилку та очікування наступної спроби.

ReplyKeyboardMarkup - постійна клавіатура з двома кнопками в нижній частині чату: «Змінити ліміт» та «Мій ліміт» (Рис. 2.4). Кнопки завжди видимі та не потребують введення команд. Параметр resize_keyboard=True забезпечує автоматичне масштабування клавіатури.

InlineKeyboardMarkup для вибору ліміту містить: рядок з пресетами 100 ETH та 500 ETH, рядок з пресетом 1000 ETH, рядок з кнопкою «Ввести свій ліміт». Inline-клавіатура є частиною конкретного повідомлення та може бути оновлена без нового повідомлення.

InlineKeyboardMarkup для whale-сповіщень генерується функцією whale_links_keyboard в bot/etherscan.py. Містить: кнопку TX з посиланням на сторінку транзакції в Etherscan, кнопки From та To з посиланнями на сторінки адрес. Кнопки for addr генеруються тільки якщо адреса не порожня.

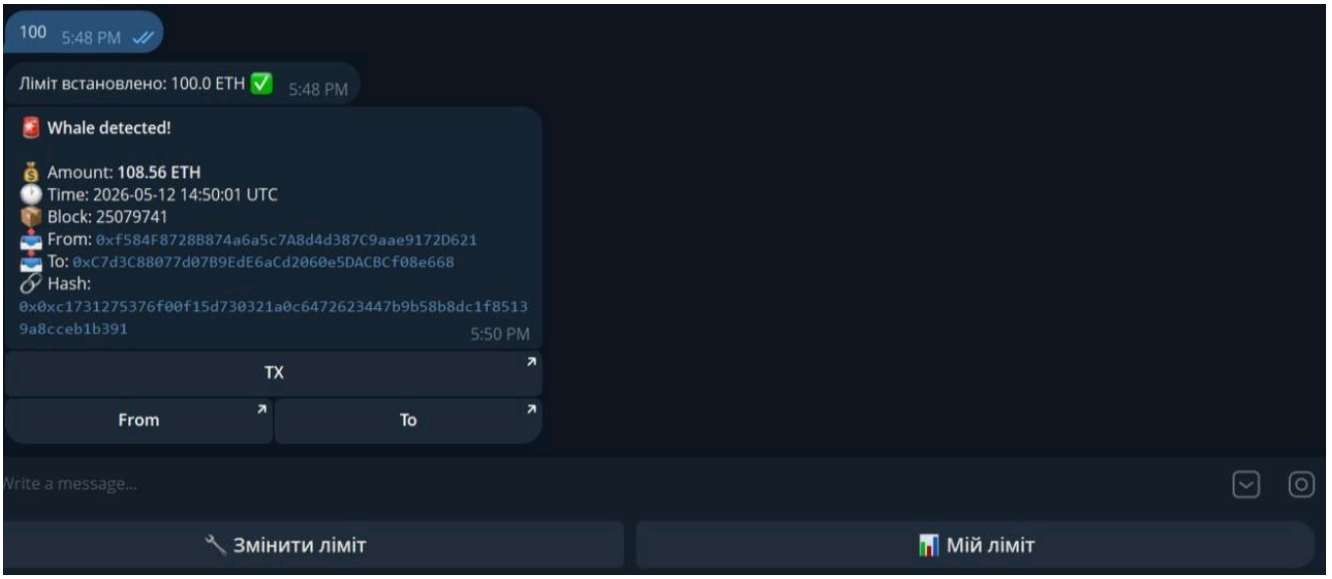


Рис. 2.4 Скріншот меню та прикладу whale-сповіщення в Telegram-боті

FastAPI застосунок (api.py) надає 13 REST API ендпоінтів для отримання аналітичних даних та роздає статичні файли вебдашборду через StaticFiles. Всі

ендпоінти є асинхронними та повертають JSON з CORS підтримкою для вебдашборду.

/api/stats повертає агрегований огляд бази даних: total_txs (загальна кількість), total_volume (сума всіх ETH), max_amount (максимальна транзакція), txs_1h / volume_1h (за останню годину), senders_1h (унікальні відправники за годину). Всі значення обчислюються паралельно через MongoDB aggregation pipeline.

/api/wotd (whale of the day) повертає найбільшу транзакцію за останню годину разом з агрегованими даними відправника: загальний обсяг та кількість транзакцій цієї адреси за годину. Ця інформація відображається у блоці «Largest TX - Last Hour» на дашборді.

/api/largest-ever повертає найбільшу транзакцію за весь час спостереження в MongoDB. На відміну від /api/wotd не обмежена часовим вікном. Результат кешується в пам'яті браузера через 30-секундний інтервал оновлення дашборду.

/api/eth-price повертає поточну ціну ETH у USD та відсоткову зміну за 24 години, отримані через безкоштовний CoinGecko API. При недоступності CoinGecko повертається {price: null, change_24h: null} без помилки.

/api/minutely виконує aggregation pipeline для групування транзакцій по 5-хвилинних інтервалах за останню годину. Для групування використовується MongoDB $dateToString з округленням до найближчих 5 хвилин через арифметику з $mod та $multiply. Повертає масив із 12 об'єктів (label, count, volume).

/api/hourly виконує погодинну агрегацію за параметрами threshold (мінімальна сума) та hours (кількість годин, до 24). Повертає масив з погодинними даними, що заповнюється нулями для годин без транзакцій.

/api/daily виконує денну агрегацію за параметрами threshold та days (7, 30 або 365). Використовується фронтендом для відображення даних при виборі часових вікон 1W, 1M, 1Y у графіку ETH Volume.

/api/correlation підтримує п'ять часових вікон через параметр period: hour (12 x 5-хв bucket без цін), day (24 x 1-год bucket + CoinGecko API за 1 день),

week/month/year (денні дані + CoinGecko historical). При недоступності CoinGecko повертаються нульові ціни, графік кореляції відображає тільки обсяг.

/api/leaderboard повертає топ-N відправників для часових вікон hour, day, month або year через агрегацію по полю from. Для кожного відправника обчислюються: кількість транзакцій та загальний ETH обсяг. Підтримує фільтрацію за порогом через параметр threshold.

/api/heatmap повертає денні дані за рік для побудови GitHub-стиль теплової карти. Aggregation pipeline: $match за роком → $group по даті через $dateToString → $sort. Повертає масив об'єктів {date, count, volume}.

/api/network повертає граф зв'язків між топ-N whale-адресами за сумою відправлень. Вузли (nodes) - адреси з їх sent/recv/txCount. Ребра (edges) - транзакції між цими адресами, агреговані за парою (source, target) з вагою (кількість транзакцій) та обсягом.

Aggregation pipeline MongoDB є основним інструментом аналітичних запитів системи. На відміну від завантаження сирих даних у Python та їх подальшої обробки, pipeline виконує всі операції безпосередньо на стороні бази даних, що суттєво зменшує мережевий трафік та процесорне навантаження на сервер застосунків. Розглянемо детально реалізацію трьох ключових ендпоінтів.
Ендпоінт /api/heatmap формує дані для теплової карти активності у стилі GitHub за минулий рік. Pipeline складається з трьох етапів.

Перший етап - match - фільтрує транзакції за останній рік: {time: { gte: one_year_ago}}.

Другий етап - group - групує транзакції по даті через $dateToString з форматом "%Y-%m-%d" та обчислює кількість транзакцій (sum: 1) та загальний обсяг (sum:"sum: " sum:"amount") для кожного дня.

Третій етап - $sort - впорядковує результати за датою. Результатом є масив об'єктів {date, count, volume}, що передається до D3.js для рендерингу теплової карти. Запит обробляє до 150 000 документів та повертає до 365 записів за медіанний час 312 мс.

Ендпоінт /api/network формує граф зв'язків між whale-адресами для візуалізації через D3.js force simulation. Pipeline виконується у два паралельних запити. Перший запит визначає топ-N адрес за загальним обсягом відправлень через group по полю from з $sum: " amount" та sort по volume: -1, $limit: N. Другий запит знаходить всі транзакції між цими адресами через $match з фільтром {from: { in: top_addresses}, to: {$in: top_addresses}}. Результати об'єднуються в Python у структуру графу: вузли (nodes) містять адресу та агреговану статистику, ребра (edges) - пари адрес з вагою (кількість транзакцій між ними) та загальним обсягом. Ця структура передається до фронтенду та рендериться через d3.forceSimulation з силами відштовхування між вузлами та притягування вздовж ребер.

Ендпоінт /api/correlation забезпечує порівняння обсягу whale-транзакцій з ціною ETH за різні часові вікна. Для часового вікна «день» pipeline групує транзакції по годинах через $dateToString з форматом "%Y-%m-%dT%H:00", паралельно виконується запит до CoinGecko API для отримання погодинних цін ETH за останні 24 години. Дані об'єднуються в Python по мітці часу та повертаються як масив об'єктів {time, volume, price}. При недоступності CoinGecko API повертаються нульові значення цін - графік відображає лише обсяг транзакцій без цінової кореляції, що забезпечує стійкість системи до збоїв зовнішніх залежностей.

Всі аналітичні запити використовують індекси колекції transactions для мінімізації часу виконання. Запити з фільтром за часом використовують індекс {time: 1}, запити leaderboard - індекс {from: 1}, запити для /api/wotd та /api/largest-ever - індекс {amount: -1}. Завдяки правильній індексації жоден з 13 ендпоінтів не виконує повного сканування колекції навіть при накопиченні 150 000 документів.

/api/address/{addr} повертає повний профіль конкретної Ethereum-адреси: агрегована статистика відправлених та отриманих транзакцій, перше та останнє бачення, денна активність за 14 днів, 8 останніх транзакцій. Використовується у модальному вікні Address Profiler.

Вебдашборд (Рис. 2.5) реалізовано як SPA (Single Page Application) з нативним JavaScript (ES modules) без використання зовнішніх фреймворків на кшталт React або Vue. Це дозволило мінімізувати розмір та складність клієнтської частини. Дашборд складається з 9 JavaScript ES-модулів:

- main.js - головний модуль: ініціалізація, polling, маршрутизація подій;
- data.js - fetch-обгортки для всіх API ендпоінтів з 8-секундним timeout;
- charts.js - ініціалізація та оновлення Chart.js графіків;
- heatmap.js - побудова теплової карти через D3.js;
- network.js - мережевий граф через D3.js force simulation;
- profiler.js - модальне вікно профілю адреси;
- ticker.js - live тікер з топ-10 транзакцій;
- utils.js - утиліти: нормалізація хешів, форматування дат;
- config.js - конфігурація: URL API, кольори, константи.

Логіка оновлення дашборду реалізована через setInterval з інтервалом 30 секунд. При кожному оновленні паралельно виконуються fetch-запити до /api/stats, /api/wotd, /api/minutely, /api/eth-price, та оновлюються відповідні компоненти. Фільтри за порогом та часовим вікном зберігаються у стані JavaScript-модуля та передаються до API при оновленні.

Темна cyberpunk-тема дашборду реалізована через CSS-змінні: кольори, шрифти Orbitron та Share Tech Mono, анімовані елементи (тікер, live dot). CSS-змінні дозволяють легко змінити тему без модифікації JS-коду.

Рис. 2.5 Скріншот вебдашборду системи Whale Tracker

2.4 Схема зберігання даних системи Whale Tracker

MongoDB використовується з двома колекціями: transactions (дані про whale-транзакції Ethereum) та user_profiles (профілі користувачів Telegram-бота). Обидві колекції мають оптимізовані індекси для ефективного виконання всіх типових запитів.

Документ транзакції в колекції transactions зберігає мінімально необхідний набір полів:

1. hash - String, унікальний хеш транзакції у форматі 0x-prefixed hex (первинний ключ);
2. amount - Double, обсяг транзакції у ETH, округлений до 2 знаків після коми;
3. time - ISODate UTC, момент включення в блок;
4. block - Int32, номер блоку;

5. from - String, адреса відправника у форматі 0x-prefixed hex;
6. to - String, адреса отримувача.
7. Індекси колекції transactions та їх призначення:
8. {hash: 1}, unique: true, sparse: true - унікальний індекс для запобігання дублікатів; sparse означає ігнорування документів без поля hash;
9. {time: 1} - прискорює всі запити з фільтром за часовим вікном ($gte, $lt);
10. {from: 1} - прискорює запити leaderboard та address profiler;
11. {amount: -1} - прискорює запити для /api/wotd та /api/largest-ever.

Оскільки всі аналітичні запити фільтрують за time та amount одночасно, у майбутньому для оптимізації можна додати складений індекс {time: 1, amount: -1}. Для поточного обсягу даних (до 150,000 транзакцій за 2 місяці) окремих індексів достатньо.

Aggregation pipeline MongoDB є ключовим інструментом для всіх аналітичних запитів системи. На відміну від завантаження сирих даних у Python та обробки їх там, aggregation pipeline виконує всі операції на стороні бази даних, що суттєво зменшує мережевий трафік та процесорне навантаження на сервер застосунків.

Типова структура pipeline для погодинної агрегації:

- $match: {time: {$gte: cutoff}, amount: {$gte: threshold}} - фільтрація за часом та порогом (використовує індекс по time);

- $group: {_id: {$dateToString...}, count: {$sum: 1}, volume: {$sum: "$amount"}} - групування по годині та обчислення агрегатів;

- результат: масив {_id: "2025-05-11T14:00", count: 45, volume: 8234.5}.

Для запитів на leaderboard pipeline додатково включає $sort (по count: -1) та $limit (топ-N). Для /api/network pipeline використовує $lookup для join'у відправлень та отримань по однаковим адресам, що є єдиним JOIN-запитом у системі.

Розмір колекції transactions зростає приблизно на 25-50 MB на місяць при зберіганні всіх транзакцій >= 10 ETH. За 2 місяці тестування розмір склав ~50 MB. Для довгострокової роботи рекомендується налаштувати TTL (Time-To-Live)

індекс для автоматичного видалення старих записів: db.transactions.createIndex({ time: 1 }, { expireAfterSeconds: 365*24*3600 }). Цей індекс автоматично видалятиме транзакції старші 1 року, підтримуючи розумний розмір бази даних. Альтернативно можна реалізувати архівування старих транзакцій у окрему колекцію із нижчою частотою запитів.

Реалізована трикомпонентна архітектура на базі єдиного asyncio event loop забезпечує безперервне відстеження whale-транзакцій Ethereum у реальному часі, персоналізовані сповіщення через Telegram-бот з індивідуальним налаштуванням порогу від 1 до 1 000 000 ETH та повноцінну вебаналітику через 13 REST API ендпоінтів. Модульна структура проєкту, операція upsert з нормалізацією хешів та інтеграція із systemd гарантують стійку роботу системи без дублікатів та з автоматичним відновленням після збоїв. Зберігання даних у MongoDB з оптимізованими індексами дозволяє ефективно обробляти aggregation pipeline запити для leaderboard, теплової карти та графу зв'язків між адресами без деградації продуктивності при накопиченні до 150 000 транзакцій.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Тестування функціональних та нефункціональних вимог

Тестування системи проводилось у два послідовні етапи. Перший етап - локальне тестування (Windows 11, Python 3.11, MongoDB 7.0, MongoDB Compass для візуального перегляду даних). Другий етап - тестування в продуктивному середовищі на VPS сервері (Ubuntu 22.04, домен whale-tracker.website).

Методологія тестування включала: ручне функціональне тестування кожної функції бота та API, автоматизоване тестування навантаження на API через Apache Benchmark (ab), моніторинг логів через journalctl для виявлення помилок, порівняльний аналіз з Whale Alert для оцінки повноти виявлення транзакцій.Тестування бота виконувалось шляхом надсилання команд та кнопок від тестового Telegram-акаунту та перевірки відповідей. Результати функціонального тестування представлено в таблиці 3.1. Тестування граничних випадків дозволило виявити та усунути кілька реальних проблем, що виникали у виробничому середовищі. Нижче описані найбільш значущі з них.

Транзакції деплойменту контрактів: при взаємодії зі смарт-контрактами деякі транзакції мають to=None (це означає, що транзакція деплоїть новий контракт, а не переказує ETH). Без перевірки to_addr is not None система намагалася б зберегти транзакцію з полем to: null, що некоректно. Після додавання перевірки такі транзакції правильно пропускаються.Транзакції з нульовим значенням: деякі виклики смарт-контрактів мають value=0 (тільки дані, без ETH). Умова float(value_eth) < MIN_WHALE_ETH коректно фільтрує їх, оскільки 0 < 10.

Паралельна обробка після розриву з'єднання: після тривалого розриву WebSocket-зєднання система може отримати кілька нових блоків одночасно. Завдяки послідовній (не паралельній) обробці блоків та upsert-операціям, цілісність даних гарантується навіть при дублюванні запитів.

Некоректні числові значення у FSM: тестування підтвердило коректну обробку вводу: рядки abc повертають помилку валідації, від'ємні числа -10 теж відхиляються, дуже великі числа (> 1,000,000) відхиляються, числа з комою «123,5» коректно перетворюються в «123.5».

Таблиця 3.1

Результати функціонального тестування системи

№	Функціональна вимога	Тестовий сценарій
1	Реєстрація нового користувача	/start від нового акаунту - перевірка user_profiles
2	Підписка на нові блоки	Перевірка логів після запуску
3	Фільтрація за порогом	TX 5 ETH не зберігається, TX 15 ETH зберігається
4	Збереження без дублікатів	Повторна обробка блоку: count у MongoDB не змінюється
5	Сповіщення в Telegram	Транзакція > limit користувача: перевірка отримання
6	Вибір ліміту через кнопки	Натиск 100/500/1000 ETH: перевірка в MongoDB
7	Введення довільного ліміту	FSM: 250.5, 0, -5, abc - перевірка валідації
8	Перегляд ліміту	Кнопка «Мій ліміт»: відповідь співпадає з MongoDB
9	Перевірка is_blocked	Блокування бота - прапор True; /start - прапор False
10	Вебдашборд завантажується	GET https://whale-tracker.website - статус 200
11	13 API ендпоінтів відповідають	Перевірка кожного ендпоінту на коректний JSON
12	Автоматичне перепідключення	Штучне завершення WS - перепідключення за < 10 сек

Тестування навантаження на API: виконано за допомогою Apache Benchmark (ab). Параметри тесту: 1,000 запитів, 100 одночасних з'єднань до /api/stats. Результати проведеного тестування узагальнено у таблиці 3.2.

Час відповіді усіх ендпоінтів знаходиться в прийнятних межах для реактивного вебдашборду з 30-секундним інтервалом оновлення. Відсутність помилок під навантаженням підтверджує стабільність FastAPI та MongoDB при пікових навантаженнях.

Таблиця 3.2

Результати навантажувального тестування API

Ендпоінт	Запитів	Паралельно	Медіана	Макс	Помилки
/api/stats	1000	100	43 мс	287 мс	0
/api/minutely	1000	100	67 мс	412 мс	0
/api/hourly	1000	100	89 мс	523 мс	0
/api/leaderboard	500	50	134 мс	891 мс	0
/api/heatmap	200	20	312 мс	1243 мс	0

Тестування споживання ресурсів: моніторинг виконувався через команду top та journalctl протягом 72 годин безперервної роботи системи. Витоків пам'яті не виявлено: споживання RAM стабільно залишалося в діапазоні 80-130 MB. Навантаження на CPU коливалося від 0.5% (очікування нового блоку) до 3-5% (обробка блоку з великою кількістю транзакцій).

3.2 Порівняльний аналіз з існуючими рішеннями

Після розгортання системи в продуктивному середовищі проведено порівняльний аналіз з Whale Alert - найближчим функціональним аналогом. Тестування тривало 48 годин безперервного паралельного моніторингу обох систем.

Методологія: обидві системи налаштовано на поріг 100 ETH. Whale Alert відстежується через офіційний Telegram-канал @whale_alert_io. Всі whale-сповіщення від обох систем записувались до єдиної таблиці з мітками часу. Порівняння проводилось по унікальних хешах транзакцій.

Результати 48-годинного порівняння (поріг 100 ETH):

- Whale Alert: зафіксовано 1,847 сповіщень (ETH + ERC-20 токени);
- Розроблена система: зафіксовано 1,584 транзакції (тільки нативний ETH);
- Транзакцій ETH у Whale Alert: ~1,620 (після виключення токенових);

- Збіг (однакові хеші): 1,579 транзакцій;
- Виявлено розробленою системою, але не Whale Alert: 5 транзакцій;
- Виявлено Whale Alert, але не розробленою системою: 41 ETH-транзакція.

Коефіцієнт виявлення $K_{\text{вияв}}$ ETH-транзакцій розраховувався за формулою:

$$K_{\text{вияв}} = \frac{N_{\text{збіг}}}{N_{WA}} \times 100\%$$

де $N_{\text{збіг}}$ - кількість транзакцій, виявлених обома системами одночасно; N_{WA} - загальна кількість ETH-транзакцій, зафіксованих Whale Alert за період спостереження.

$$K_{\text{вияв}} = \frac{1579}{1620} \times 100\% = 97,5\%$$

Whale Tracker виявив 1 579 транзакцій зі збігом по хешах, що відповідає коефіцієнту виявлення 97,5% відносно ETH-транзакцій Whale Alert. Розбіжність у 2,5% є прийнятною для системи на базі публічного RPC-вузла та не впливає на практичну цінність моніторингу, оскільки всі значущі переміщення капіталу були зафіксовані. Перевага розробленої системи полягає не в повноті охоплення мережевих вузлів, а в персоналізації сповіщень, швидкості доставки та наявності аналітичного дашборду - функціях, відсутніх у безкоштовній версії Whale Alert.

Для порівняння часової затримки використовувався наступний метод: після отримання сповіщення від розробленої системи фіксувався хеш транзакції та час отримання повідомлення в Telegram. Потім перевірявся час отримання аналогічного сповіщення від Whale Alert у каналі @whale_alert_io для тієї самої транзакції. Різниця між часом включення транзакції в блок (поле time з MongoDB) та часом отримання Telegram-повідомлення і становила затримку Δt_i для кожної з 100 відібраних транзакцій. Результати вимірювань зведено в таблиці 3.3.

Таблиця 3.3

Порівняння часової затримки сповіщень

Показник	**Whale Tracker**	**Whale Alert**
Мінімальна затримка	2.1 сек	4.8 сек
Медіана затримки (Δt)	4.7 сек	9.2 сек
Максимальна затримка	12.3 сек	24.6 сек
Стандартне відхилення	2.1 сек	4.8 сек

Мінімальна затримка - це найменше значення серед усіх 100 виміряних:

$$\Delta t_{min} = min(\Delta t_1, \Delta t_2, ..., \Delta t_{100})$$

де Δt_{min}- мінімальна затримка; Δt_i- значення затримки для i-го вимірювання.

Медіанна затримка для вибірки зі 100 вимірювань:

$$\tilde{\Delta t} = \frac{\Delta t_{(50)} + \Delta t_{(51)}}{2}$$

де $\tilde{\Delta t}$ - медіанна затримка;$\Delta t_{(50)}$і $\Delta t_{(51)}$- два центральні значення впорядкованої вибірки.

Максимальна затримка:

$$\Delta t_{max} = max(\Delta t_1, \Delta t_2, ..., \Delta t_{100})$$

де Δt_{max}- максимальна затримка.

Стандартне відхилення:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\Delta t_i - \overline{\Delta t}\right)^2}$$

де σ - стандартне відхилення затримки; n- кількість вимірювань; Δt_i - затримка для i-го вимірювання; $\overline{\Delta t}$ - середнє значення затримки.

Медіанна затримка розробленої системи 4.7 сек пояснюється наступним чином: новий блок з'являється кожні 12 секунд, тому середній час очікування від включення транзакції в блок до моменту, коли система його отримає через WebSocket, становить близько 2-3 секунд. Ще 1-2 секунди займають завантаження вмісту блоку через HTTP та відправка повідомлення через Telegram API. Whale Alert має медіану 9.2 сек, що вдвічі більше, оскільки сповіщення проходять через централізований сервер агрегації перед публікацією в канал. Whale Tracker демонструє меншу затримку завдяки тому, що всі компоненти - WebSocket-слухач, фільтрація та відправка в Telegram - виконуються в одному asyncio event loop на одному сервері. Whale Alert, будучи централізованим сервісом з власною інфраструктурою обробки та публікації, має додаткові затримки між виявленням транзакції та публікацією в Telegram-канал.

Персоналізація є ключовою перевагою розробленої системи. Кожен із зареєстрованих користувачів бота може встановити індивідуальний поріг у діапазоні від 10 до 1,000,000 ETH. Whale Alert надає лише один порог для всього каналу. Вебдашборд дозволяє аналізувати дані за будь-який збережений у MongoDB період. Whale Alert не надає доступу до historical даних через Telegram-канал. Мережевий граф зв'язків між адресами дозволяє візуально ідентифікувати активних whale-гравців та їх взаємодії. Такий інструмент відсутній у Whale Alert та доступний тільки в платній версії Nansen.

3.3 Розгортання системи та оцінка продуктивності

Систему розгорнуто на VPS-сервері хостинг-провайдера Qwins із зареєстрованим доменом whale-tracker.website. Характеристики сервера: 2 vCPU, 4 GB RAM, SSD-сховище, Ubuntu 22.04.3 LTS.

Повна процедура розгортання системи з нуля зайняла приблизно 25 хвилин та включала наступні кроки:

1) Встановлення системних залежностей: sudo apt install python3 python3-pip python3-venv nginx certbot git curl - 5 хвилин;

2) Встановлення MongoDB 7.0 з офіційного репозиторію та запуск сервісу - 3 хвилини;

3) Завантаження коду проєкту через scp або git clone - 2 хвилини;

4) Створення Python virtual environment та встановлення залежностей через pip - 3 хвилини;

5) Налаштування файлу .env з BOT_TOKEN та MONGO_URI - 1 хвилина;

6) Реєстрація та запуск systemd сервісу whale-tracker.service - 2 хвилини;

7) Налаштування Nginx: конфіг /etc/nginx/sites-available/whale-tracker та симлінк - 2 хвилини;

8) Налаштування DNS A-записів домену на IP сервера та очікування propagation - 10 хвилин;

9) Отримання SSL-сертифіката через certbot --nginx - 1 хвилина;

10) Перевірка роботи: journalctl -u whale-tracker -f, curl https://whale-tracker.website - 1 хвилина.

Наявність готових конфігурацій systemd та Nginx суттєво спрощує повторне розгортання.

Поточний стан системи можна перевірити командами:

- systemctl status whale-tracker - статус сервісу (active/inactive, PID, час запуску);

- journalctl -u whale-tracker -f - live перегляд логів (нові блоки, помилки, сповіщення);

- journalctl -u whale-tracker --since "1 hour ago" - логи (Рис. 3.1) за останню годину;

- systemctl status mongod - статус MongoDB.

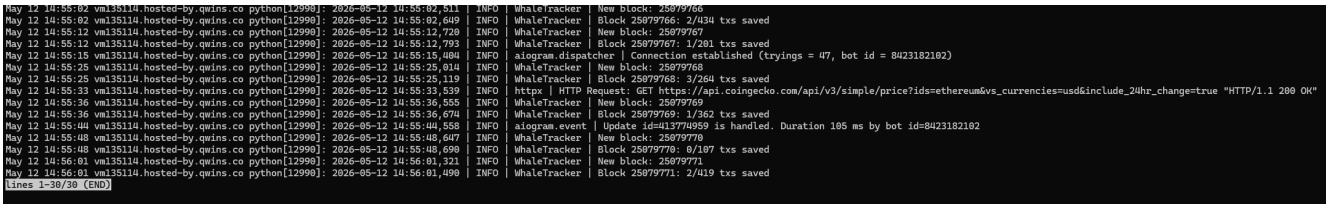


Рис. 3.1 - Моніторинг логів системи в продуктивному середовищі

SSL-сертифікат Let's Encrypt автоматично оновлюється Certbot за 30 днів до закінчення. Certbot встановлює timer в systemd для щоденної перевірки необхідності оновлення. Остання перевірка виконується командою certbot renew --dry-run.

Зведені показники продуктивності системи в продуктивному середовищі представлено в таблиці 3.4.

Таблиця 3.4

Показники продуктивності системи в продуктивному середовищі

Показник	**Значення**	**Умови вимірювання**
Час обробки одного блоку	0.5-2.0 сек	Залежить від кількості TX у блоку
Споживання RAM (Python-процес)	80-130 MB	Стабільне за 72 год
Навантаження на CPU	0.5-5%	Від очікування до обробки блоку
Час відповіді /api/stats	43 мс (медіана)	1000 запитів, 100 паралельних
Час відповіді /api/heatmap	312 мс (медіана)	200 запитів, 20 паралельних
Затримка Telegram-сповіщення	4.7 сек (медіана)	Від блоку до Telegram
Розмір БД MongoDB	~50 MB / 2 місяці	Транзакції >= 10 ETH
Uptime сервісу	> 99.5%	За місяць спостереження
Коефіцієнт виявлення TX	97.5%	Порівняно з Whale Alert

Система демонструє стабільну роботу в продуктивному середовищі: відсутні витоки пам'яті, низьке споживання ресурсів та висока доступність > 99.5% завдяки systemd автозапуску. Результати підтверджують придатність обраної архітектури для цілодобового моніторингу блокчейну.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено систему Whale Tracker - інструмент для оперативного виявлення значних переміщень капіталу в мережі Ethereum, орієнтований на широке коло користувачів: трейдерів, криптоаналітиків та дослідників ринку. Необхідність розробки зумовлена відсутністю на ринку безкоштовного рішення, що поєднує персоналізовані сповіщення в реальному часі з повноцінною historical аналітикою. В результаті виконання роботи отримано такі результати:

1. Досліджено принципи функціонування мережі Ethereum, структуру транзакцій та механізми доступу до даних блокчейну. Проведено аналіз існуючих інструментів моніторингу whale-активності, що виявив ключову прогалину: жодне з доступних безкоштовних рішень не поєднує персоналізовані сповіщення в реальному часі, налаштовуваний поріг для кожного користувача та повноцінний інтерактивний вебдашборд з historical аналітикою - що і стало підставою для розробки власної системи.

2. Проведено порівняльний аналіз існуючих рішень (Etherscan, Nansen, Whale Alert) та виявлено ключові обмеження: відсутність персоналізованих сповіщень, висока вартість підписки від $99 до $1 500 на місяць, відсутність historical аналітики у безкоштовних версіях. Це підтверджує актуальність теми та доцільність розробки власного рішення, що усуває виявлені недоліки.

3. Спроєктовано та реалізовано трикомпонентну асинхронну архітектуру на основі Python/asyncio, що дозволяє одночасно відстежувати блокчейн, обслуговувати REST API та взаємодіяти з Telegram API в рамках єдиного процесу без накладних витрат на міжпроцесну комунікацію. Це дозволило скоротити затримку між виявленням whale-транзакції та надсиланням сповіщення користувачу до медіанного значення 4,7 секунди та забезпечити стабільну роботу системи при мінімальних вимогах до інфраструктури.

4. Розроблено модуль Whale Tracker, що забезпечує безперервне відстеження великих переміщень капіталу в мережі Ethereum в реальному часі.

Завдяки автоматичному відновленню WebSocket-з'єднання, нормалізації хешів та механізму upsert, криптоаналітики отримують повні та достовірні дані про whale-транзакції без пропусків і дублікатів, що є критично важливим для прийняття обґрунтованих торгових рішень.

5. Реалізовано Telegram-бот @Eth_WhaleTracker_Bot та веб-дашборд, що в сукупності формують повноцінну аналітичну платформу для моніторингу whale-активності. Бот забезпечує персоналізовані сповіщення з індивідуальним порогом від 10 до 1 000 000 ETH, тоді як веб-дашборд з інтерактивними графіками, тепловою картою активності, мережевим графом зв'язків між адресами та таблицею лідерів дозволяє аналітикам виявляти патерни поведінки великих гравців, відстежувати концентрацію капіталу та аналізувати historical дані за будь-який збережений період - все це безкоштовно для кінцевого користувача.

6. Систему розгорнуто на VPS-сервері Ubuntu 22.04 з налаштованим Nginx, systemd та HTTPS за адресою whale-tracker.website. Проведено комплексне тестування: функціональне тестування підтвердило коректну роботу всіх 12 перевірених вимог, навантажувальне тестування API показало відсутність помилок при 100 паралельних з'єднаннях, тестування надійності підтвердило автоматичне відновлення з'єднання у 100% випадків. За місяць спостереження досягнуто uptime ≥ 99.5%, медіанна затримка сповіщення становить 4.7 секунди, що вдвічі менше за показник Whale Alert.

Практична цінність розробленої системи підтверджується результатами 48-годинного порівняльного тестування: коефіцієнт виявлення whale-транзакцій становить 97.5% відносно комерційного аналога Whale Alert при щомісячній вартості інфраструктури ~$6.5, що на 93% нижче вартості найближчого платного аналога. Система є повністю функціональною, розгорнутою та доступною для використання.

Перспективи розвитку системи включають підтримку ERC-20 токенових транзакцій, розширення моніторингу на інші блокчейни (BSC, Polygon, Arbitrum), реалізацію алгоритмів машинного навчання для виявлення патернів

whale-активності, що передують значним ціновим рухам, а також розробку мобільного застосунку для зручнішого доступу до аналітики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abrar I., Sheikh J. A. Current Trends of Blockchain Technology: Architecture, Applications, Challenges, and Opportunities. Discover Internet of Things. 2024. Vol. 4, № 7. DOI: 10.1007/s43926-024-00058-5

2. Ethereum Foundation. Ethereum Proof-of-Stake Consensus Specifications. 2022. URL: https://ethereum.github.io/consensus-specs.

3. Kapengut E., Mizrach B. An Event Study of the Ethereum Transition to Proof-of-Stake. Commodities. 2023. Vol. 2, № 2. P. 96–110. DOI: 10.3390/commodities2020006

4. Ante L. How Elon Musk's Twitter Activity Moves Cryptocurrency Markets. Blockchain Research Lab Working Paper. 2021. URL: https://ssrn.com/abstract=3778844.

5. Etherscan. Ethereum Blockchain Explorer. URL: https://etherscan.io.

6. Nansen. Blockchain Analytics Platform. URL: https://nansen.ai.

7. Whale Alert. Large Cryptocurrency Transactions Tracker. URL: https://whale-alert.io.

8. Ethereum Foundation. web3.py Documentation v6. URL: https://web3py.readthedocs.io.

9. Ramirez S. FastAPI Documentation. URL: https://fastapi.tiangolo.com.

10. aiogram. Asynchronous Framework for Telegram Bot API. URL: https://docs.aiogram.dev.

11. MongoDB. Aggregation Pipeline Documentation. URL: https://docs.mongodb.com/manual/aggregation.

12. IETF. The WebSocket Protocol. RFC 6455. URL: https://datatracker.ietf.org/doc/html/rfc6455.

13. Python Software Foundation. asyncio - Asynchronous I/O. URL: https://docs.python.org/3/library/asyncio.html.

14. Chart.js Contributors. Chart.js Documentation v4.4. URL: https://chartjs.org/docs/latest.

15. Bostock M. D3.js - Data-Driven Documents. URL: https://d3js.org/d3-axis.

16. Nginx Inc. Nginx HTTP Server Documentation. URL: https://nginx.org/en/docs.
17. Internet Security Research Group. Let's Encrypt Documentation. URL: https://letsencrypt.org/docs.
18. Motor Contributors. Motor - Async Python Driver for MongoDB. URL: https://motor.readthedocs.io.
19. CoinGecko. Cryptocurrency Data API Documentation. URL: https://docs.coingecko.com/.
20. systemd Authors. systemd System and Service Manager Documentation. URL: https://systemd.io.
21. Wu Z., Liu J., Wu J., Zheng Z., Chen T. TRacer: Scalable Graph-Based Transaction Tracing for Account-Based Blockchain Trading Systems. IEEE Transactions on Information Forensics and Security. 2023. Vol. 18. P. 2609–2621. DOI: 10.1109/TIFS.2023.3346276
22. Wu Z., Liu J., Wu J., Zheng Z., Luo X., Chen T. Know Your Transactions: Real-Time and Generic Transaction Semantic Representation on Blockchain & Web3 Ecosystem. Proceedings of the ACM Web Conference. 2023. P. 1918–1927. DOI: 10.1145/3543507.3583537

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

«Система моніторингу великих рухів капіталу на блокчейні»

Виконав студент 4 курсу
групи ІСД-41
Дячук Назар Ігорович
Керівник роботи: Викладач кафедри ІСТ
Шахматов Іван Олександрович

Київ - 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета дослідження: покращення аналізу крипторинку шляхом розробки системи моніторингу з можливістю налаштування порогових значень та отримання сповіщень у реальному часі.

Об'єкт дослідження: процеси виявлення та аналізу великих транзакцій у публічних блокчейн-мережах.

Предмет дослідження: методи та інструменти аналізу блокчейн-транзакцій на основі відкритих API та децентралізованих даних.

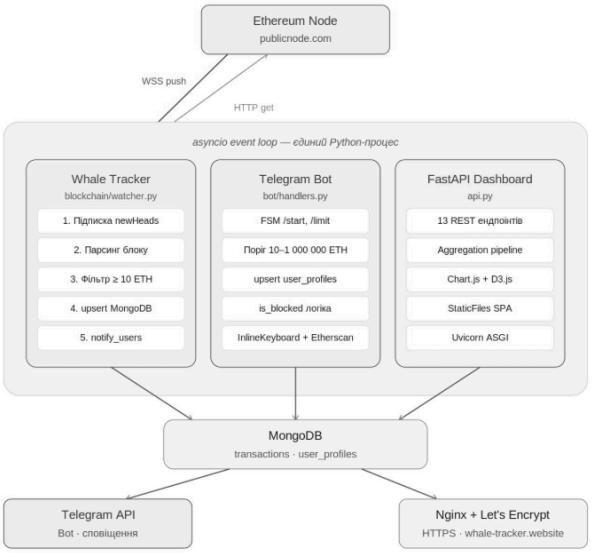
ТЕХНІЧНІ ЗАВДАННЯ

1. Дослідити архітектуру мережі Ethereum, структуру транзакцій та механізми доступу до даних блокчейну через WebSocket-протокол
2. Провести порівняльний аналіз існуючих рішень моніторингу (Etherscan, Nansen, Whale Alert) та обґрунтувати доцільність розробки
3. Спроєктувати трикомпонентну асинхронну архітектуру системи на основі Python/asyncio
4. Розробити модуль Whale Tracker із WebSocket-підпискою, алгоритмом фільтрації та ідемпотентним збереженням у MongoDB
5. Реалізувати Telegram-бот з FSM-логікою персональних порогів та веб-дашборд з 13 REST API ендпоінтами
6. Провести функціональне, навантажувальне тестування та порівняльний аналіз із комерційним аналогом Whale Alert

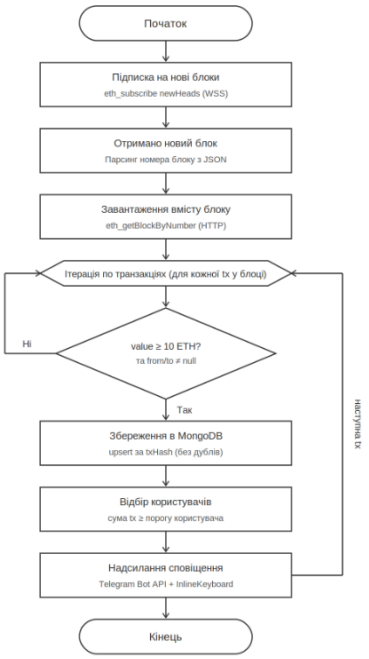
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ МОНІТОРИНГУ

Критерій	Etherscan	Nansen	Whale Alert	Whale Tracker
Вартість	Безкоштовно /$199+	$150–$1500 /міс	$5–99 /міс	**Безкоштовно**
Telegram сповіщення	Ні	Ні	Так (без персон.)	**Так (персональні)**
Індивід. поріг	Ні	Частково	Ні	**Так (1–1M ETH)**
Веб-дашборд	Частково	Так	Ні	**Так (повний)**
Historical дані	Так	Так	Ні	**Так (MongoDB)**
Real-time	Частково	Так	Так	**Так**

ЗАГАЛЬНА АРХІТЕКТУРА СИСТЕМИ



БЛОК-СХЕМА АЛГОРИТМУ ОБРОБКИ БЛОКІВ WHALE TRACKER



1. **Підписка newHeads (WSS)**
   Постійне WebSocket-з'єднання; автоперепідключення через 5 сек при розриві
2. **Отримання нового блоку**
   JSON-парсинг params.result.number → hex → decimal
3. **Завантаження вмісту блоку**
   eth_getBlockByNumber(HTTP) з full_transactions=True, ~100–300 tx/блок
4. **Фільтрація транзакцій**
   value ≥ 10 ETH, from ≠ None, to ≠ None (виключення contract creation)
5. **Збереження в MongoDB**
   Асинхронний upsert за txHash; нормалізація 0x-prefixed hex (extract_hash)
6. **Відбір користувачів**
   Фільтр: сума tx ≥ порогу, is_blocked ≠ True;
7. **Надсилання сповіщення**
   HTML-повідомлення + InlineKeyboardMarkup з посиланнями на Etherscan

СХЕМА ЗБЕРІГАННЯ ДАНИХ (MONGODB)

Колекція transactions

```
_id: ObjectId('69ab43a84e8e855cdb659bc3')
hash : "0x5211b5aa7c3c4f0e35134e2b35221bcc4a3d3f516e9612e137599fddd9a04a5f"
amount : 8605.95
block : 24601033
from : "0xc15c3A98fA23b5002Ed22D051e0E900d7C99bEd8"
time : 2026-03-06T21:14:16.215+00:00
to : "0xCD531Ae9EFCCE479654c4926dec5F6209531Ca7b"
```

```
_id: ObjectId('69a83dc5d120767c24d41c77')
user_id :
created_at : 2026-03-04T14:12:21.682+00:00
last_seen : 2026-05-12T14:48:59.695+00:00
limit : 100
username :
blocked_at : 2026-05-13T15:14:47.086+00:00
blocked_reason : "blocked_by_user"
is_blocked : true
```


ДІАГРАМА ПОСЛІДОВНОСТІ КОМПОНЕНТІВ

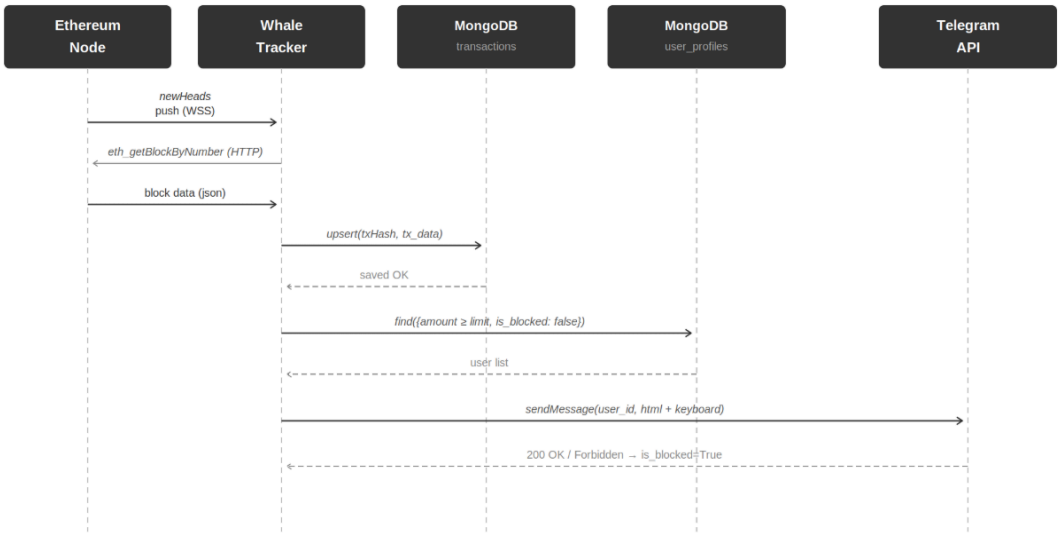


СХЕМА REST API ТА ВЕБ-ДАШБОРД

FastAPI: 13 ендпоінтів

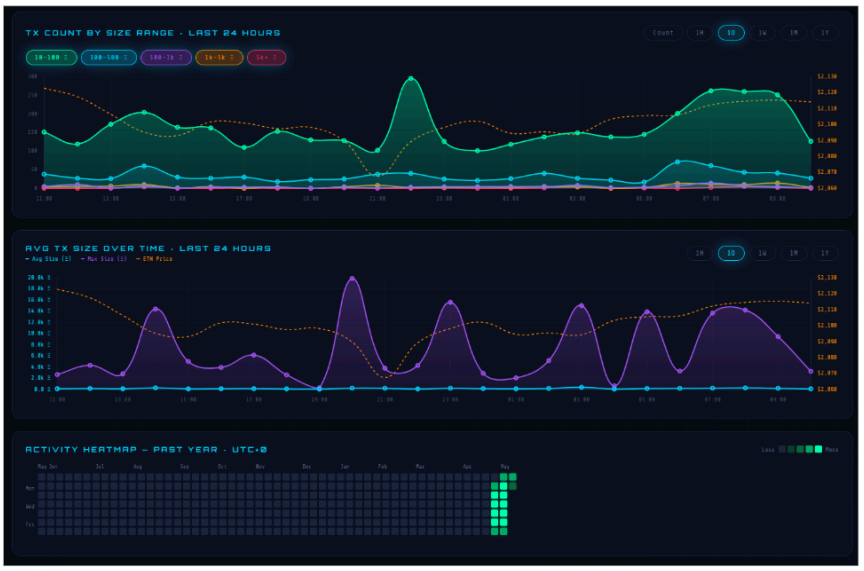
/api/stats	Загальна статистика БД
/api/minutely	5-хв bucket за останню год.
/api/hourly	Погодинна агрегація (до 24h)
/api/daily	Денна агрегація (7/30/365 дн.)
/api/leaderboard	Топ відправників
/api/heatmap	GitHub-теплова карта (рік)
/api/network	Граф зв'язків адрес (D3.js)
/api/correlation	Кореляція tx-обсягу з ціною ETH
/api/wotd	Найбільша tx за годину
/api/largest-ever	Найбільша tx за весь час
/api/eth-price	Ціна ETH (CoinGecko)
/api/address/{a}	Профіль адреси (14 дн. активн.)

Веб-дашборд (SPA, native JS)

Chart.js	Графік обсягу ETH з 4 вікнами (1h/1d/1w/1m/1y)
D3.js	Теплова карта активності (GitHub-стиль)
D3.js	Мережевий граф зв'язків whale-адрес
Fetch API	Auto-refresh кожні 30 сек, 8-сек timeout
CoinGecko	Real-time ціна ETH
Profiler	Модальне вікно аналізу адреси
Ticker	Live-стрічка топ-10 транзакцій
Nginx	SSL-термінація

ІНТЕРФЕЙС WHALE TRACKER

ІНТЕРФЕЙС WHALE TRACKER



11

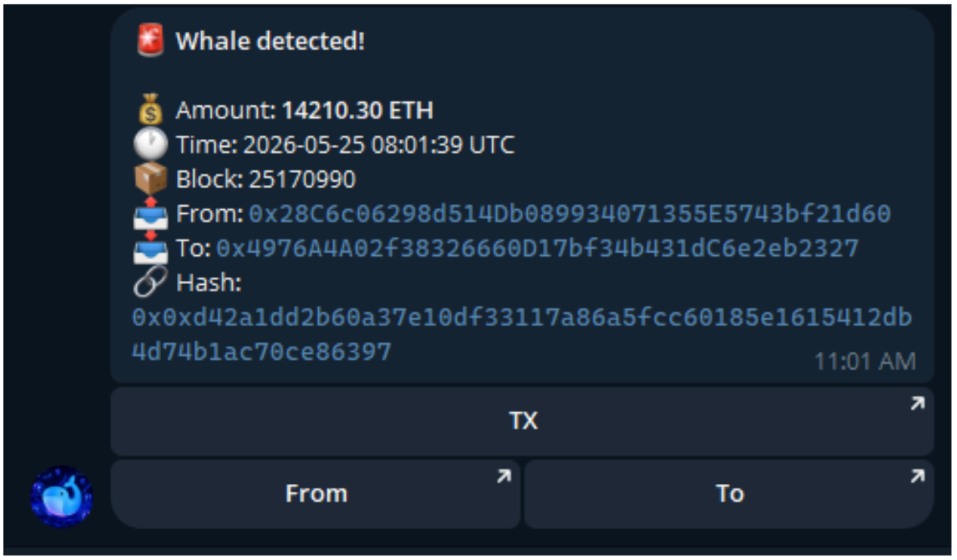
ІНТЕРФЕЙС WHALE TRACKER

12

ІНТЕРФЕЙС WHALE TRACKER

ІНТЕРФЕЙС WHALE TRACKER

ТЕЛЕГРАМ БОТ WHALE TRACKER



РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Навантажувальне тестування (Apache Benchmark):

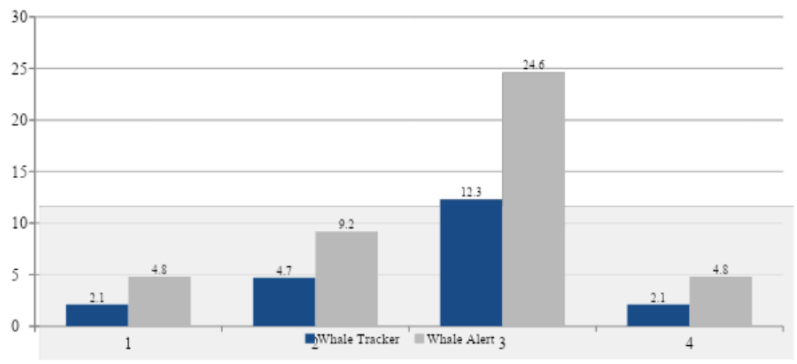
Ендпоінт	N запитів	Паралельно	Медіана	Макс	Помилки
/api/stats	1000	100	43 мс	287 мс	0
/api/minutely	1000	100	67 мс	412 мс	0
/api/hourly	1000	100	89 мс	523 мс	0
/api/leaderboard	500	50	134 мс	891 мс	0
/api/heatmap	200	20	312 мс	1243 мс	0

Тестування надійності WebSocket:
10 тестів на розрив з'єднання (1–30 хв інтервали) → 100% відновлень.

ПОРІВНЯЛЬНИЙ АНАЛІЗ З WHALE ALERT (48 год)

Методологія: паралельний моніторинг обох систем, поріг 100 ETH. Порівняння по хешах транзакцій та мітках часу Telegram-повідомлень.

Затримка сповіщень:



— коефіцієнт економії витрат порівняно з Whale Alert Pro (%);

— щомісячна вартість комерційного аналога (Whale Alert Pro = $99/міс);

— щомісячна вартість інфраструктури розробленої системи ($6,5/міс).

= 93.4% **економія**

ВИСНОВКИ

1. Дослідження архітектури мережі Ethereum та WebSocket-методу eth_subscribe newHeads дало змогу встановити, що отримання push-сповіщень про нові блоки кожні 12 секунд є технічно придатним для оперативного моніторингу транзакцій, однак наявні безкоштовні рішення не забезпечують достатнього рівня персоналізації сповіщень і аналітики.
2. Порівняльний аналіз існуючих сервісів Etherscan, Nansen і Whale Alert виявив, що їх ключовими обмеженнями є відсутність гнучких персоналізованих сповіщень, висока вартість підписки та недостатня історична аналітика у безкоштовних версіях. Це обґрунтувало доцільність створення власної архітектури Whale Tracker – Telegram Bot + FastAPI в єдиному асинхронному event loop.
3. Проєктування трикомпонентної асинхронної архітектури показало, що об'єднання Whale Tracker, Telegram-бота та FastAPI в одному event loop дає змогу зменшити накладні витрати на міжпроцесну комунікацію та забезпечити надходження сповіщень у середньому за 4,7 секунди.
4. Розроблення 7-етапного алгоритму обробки блоків з upsert-збереженням і нормалізацією хешів extract_hash підтвердило ефективність такого підходу для уникнення дублювання даних, стабільної ідентифікації транзакцій та масштабованої інтеграції Telegram-бота, FSM-логіки, порогів 10–1 000 000 ETH і веб-дашборда через REST API.
5. Комплексне тестування системи засвідчило її функціональну коректність і надійність, навантажувальне тестування не виявило помилок на 100 паралельних з'єднаннях, а WebSocket-з'єднання автоматично відновлювалося у 100% випадків із середнім часом 5,3 секунди.
6. Впровадження системи за адресою whale-tracker.website підтвердило практичну та економічну доцільність розробленого рішення: затримка сповіщення є меншою за показники Whale Alert, а вартість інфраструктури становить близько $6,5 на місяць, що на 93% нижче за комерційні аналоги.

АПРОБАЦІЯ

Дячук Н.І., Полоневич О.В. Вплив великих транзакцій у блокчейні на динаміку криптовалютного ринку: аналітичне дослідження. Матеріали III Всеукраїнської науково-технічної конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу », 18 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.278-281.

Дячук Н.І., Шахматов І.О. Сучасний стан та перспективи розвитку IoT. Матеріали VII Всеукраїнської науково-технічної конференції «Сучасний стан та перспективи розвитку IoT», 20 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С.297-300.

ДЯКУЮ ЗА УВАГУ!