Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка модуля "Моніторинг та аналіз взаємодій з рунами" у мережі «Bitcoin»
(тема)

Виконав:
здобувач IV курсу, групи ІТУ-21-1
Денис ЛЕЩЕНКО
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційні технології управління
(повна назва освітньої програми)
Керівник доцент кафедри ІУС
Вікторія ШЕХОВЦОВА
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ Костянтин ПЕТРОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Інформаційних управляючих систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Інформаційні технології управління
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«__19__» __травня_ 20 25 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувача Лещенко Денис Сергійович
(прізвище, власне ім'я, по батькові)

1. Тема роботи Розробка модуля "Моніторинг та аналіз взаємодій з рунами" у мережі «Bitcoin»
затверджена наказом університету від «_19_» __травня__2025_ р. № _370Cm_
2. Термін подання студентом роботи до екзаменаційної комісії «17» __06___ 2025 р.
3. Вихідні дані до роботи _результати передпроєктного дослідження об'єкта автоматизації (мережі Bitcoin), методичні вказівки до виконання кваліфікаційної роботи
4. Перелік питань, що потрібно опрацювати в роботі опис та аналіз структурних і функціональних особливостей протоколу Runes у мережі Bitcoin; аналіз організаційної структури інформаційної технології блокчейн-мережі Bitcoin; огляд та порівняльний аналіз існуючих систем декодування та моніторингу транзакцій з Runes; опис вимог до об'єкта розробки, формулювання завдання створення модуля моніторингу; розробка та обґрунтування елементів інформаційного забезпечення модуля; розробка елементів математичного забезпечення модуля; розробка елементів програмного забезпечення модуля на основі сучасних технологій; розробка та обґрунтування технічного забезпечення модуля; синтез і обґрунтування засобів захисту модуля від несанкціонованого доступу.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів	Примітка
1	Аналіз і дослідження структурних та функціональних особливостей об’єкта автоматизації (протоколу Runes у мережі Bitcoin). Аналіз організаційної структури інформаційної технології блокчейн-мережі Bitcoin. огляд та порівняльний аналіз існуючих систем декодування та моніторингу транзакцій з Runes	19.05.2025 – 23.05.2025	Виконано
2	Розробка і опис вимог до об’єкта розробки, формулювання завдання створення модуля моніторингу.	24.05.2025 – 27.05.2025	Виконано
3	Розробка та опис елементів інформаційного забезпечення модуля	28.05.2025	Виконано
4	Розробка та опис елементів математичного забезпечення модуля	29.05.2025 – 30.05.2025	Виконано
5	Розробка та опис елементів програмного забезпечення модуля	31.05.2025 – 05.06.2025	Виконано
6	Розробка та опис технічного забезпечення модуля	06.06.2025 – 07.06.2025	Виконано
7	Синтез засобів захисту інформації від несанкціонованого доступу до модуля	08.06.2025 – 09.06.2025	Виконано
8	Оформлення пояснювальної записки та графічної частини кваліфікаційної роботи	10.06.2025 – 12.06.2025	Виконано
9	Підготовка роботи до захисту (перевірка кваліфікаційної роботи на плагіат, перевірка роботи керівником, проходження нормоконтролю)	14.06.2025	Виконано
10	Попередній захист кваліфікаційної роботи	15.06.2025	Виконано
11	Рецензування кваліфікаційної роботи	15.06.2025	Виконано
12	Захист кваліфікаційної роботи в екзаменаційній комісії	17.06.2025	Виконано

Дата видачі завдання «__19__» ___травня__2025 р.

Студент _____

Керівник роботи _____ доц. каф. ІУС Вікторія ШЕХОВЦОВА
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 90 с., 15 табл., 37 рис., 33 дод., 32 джерела.

АНАЛІЗ, ДЕКОДУВАННЯ, БЛОКЧЕЙН, BITCOIN, RUNE, RUNESTONE, МОНІТОРИНГ, ТРАНЗАКЦІЇ, UTXO, АНАЛІТИКА, ЦИФРОВІ АКТИВИ.

Об'єктом дослідження є блокчейн-мережа Bitcoin. Об'єктом розробки є модуль «Моніторинг та аналіз взаємодій з рунами». Цей модуль є частиною інформаційної системи для аналізу цифрових активів у мережі Bitcoin.

Метою розробки модуля є підвищення ефективності, точності та швидкості декодування й аналізу транзакцій з рунами, забезпечення автоматизованого формування звітів та надійного зберігання аналітичних даних.

Методами дослідження виступають: системний аналіз, порівняльний аналіз технологій (Bitcoin Core RPC, bitcoinjs-lib, runestone-lib), експериментальне тестування, статистична обробка даних.

В результаті аналізу обґрунтовано вибір спеціалізованих бібліотек для декодування та запропоновано власну архітектуру модуля. Створено прототип модуля моніторингу та аналізу транзакцій з використанням технології Runestone.ts. Розроблено елементи інформаційного, математичного, програмного та технічного забезпечень. PostgreSQL було обрано як систему управління базами даних. Технічна реалізація базується на мікросервісній та клієнт-серверній архітектурі з використанням мови програмування TypeScript та бібліотек React, NestJS, Next.js і технології Redis, Docker.

ABSTRACT

The explanatory note to the attestation work contains: 90 p., 15 tables, 37 images, 33 addition, 32 sources.

ANALYSIS, DECODING, BLOCKCHAIN, BITCOIN, RUNE, RUNESTONE, MONITORING, TRANSACTIONS, UTXO, ANALYTICS, DIGITAL ASSETS.

The object of the research is the Bitcoin blockchain network. The subject of development is the module "Monitoring and Analysis of Interactions with Runes." This module is a part of an information system for analyzing digital assets in the Bitcoin network.

The goal of the module development is to improve the efficiency, accuracy, and speed of decoding and analyzing rune-related transactions, ensure automated report generation, and reliable storage of analytical data.

The research methods include systems analysis, comparative analysis of technologies (Bitcoin Core RPC, bitcoinjs-lib, runestone-lib), experimental testing, and statistical data processing.

As a result of the analysis, the selection of specialized libraries for decoding was justified and a custom module architecture was proposed. A prototype of the monitoring and analysis module was developed using the Runestone.ts technology. Elements of informational, mathematical, software, and technical support were implemented. PostgreSQL was chosen as the database management system. The technical implementation is based on microservice and client-server architecture using the TypeScript programming language and libraries such as React, NestJS, Next.js, as well as Redis and Docker technologies.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІЗ – інформаційне забезпечення

ІС – інформаційна система

ОС – операційна система

ПЗ – програмне забезпечення

СУБД – системи управління базами даних

API – application programming interface

BIP – bitcoin improvement proposal

BRC-20 – bitcoin request for comment 20

BTC – bitcoin

CPFP – child-pays-for-parent

FK – foreign key

HEX – hexadecimal

HTTP(S) – hypertext transfer protocol (secure)

ID – identifier

PnL – profit and loss

PK – primary key

RBF – replace-by-fee

REST – representational state transfer

RPC – remote procedure call

SQL – structured query language

SSH – secure shell

SSL – secure sockets layer

TLS – transport layer security

TX – transaction

UML – unified modeling language

UTXO – unspent transaction output

ВСТУП

З розвитком блокчейн-технологій та зростанням популярності мережі Bitcoin з'являється нагальна потреба в оперативному та точному моніторингу транзакцій. Окремим викликом є аналіз специфічних метаданих – так званих «рунів», що вбудовуються у структуру транзакцій Bitcoin. Протокол Runes («Руни») – це новий підхід, запропонований автором Ordinals Кейсі Родармором для створення взаємозамінних токенів безпосередньо на блокчейні Bitcoin. Цей протокол покликаний зробити випуск та обіг фунгібельних токенів більш ефективним, використовуючи модель unspent transaction output (UTXO) та зберігаючи всю інформацію «на ланцюгу» замість зовнішніх баз даних. Розвиток протоколу Runes підсилює актуальність задачі аналізу таких токенів, оскільки успішна реалізація токенів на Bitcoin може стимулювати нову хвилю інновацій і потребує надійних засобів відстеження їхнього обігу.

Наразі інструменти загального призначення недостатньо пристосовані для роботи з рунами. Стандартні засоби на кшталт Bitcoin Core Remote Procedure Call (RPC) надають лише сирі дані транзакцій і не виконують спеціалізованого розбору руничної інформації. Універсальні JavaScript бібліотеки (наприклад, bitcoinjs-lib) також не мають вбудованої підтримки протоколу Runes, їх універсальність іноді призводить до зниження точності при декодуванні специфічних структур токенів. Іншими словами, без спеціальних засобів важко автоматично визначити, чи містить транзакція «руни», розшифрувати їхній зміст та простежити переміщення токенів між виходами. Відсутність зручних інтерфейсів для відстеження взаємодій з рунами ускладнює роботу як розробникам, так і користувачам токенів: виникають ризики помилкової інтерпретації даних (наприклад, зловмисник може згенерувати фальшивий запис про передачу токенів без реального покриття), а перевірка автентичності таких записів вимагає спеціального

аналізу всієї історії випуску токена. Таким чином, постає потреба у розробці спеціалізованого інструменту, здатного точно та ефективно декодувати транзакції з рунами та відслідковувати рух токенів.

Метою кваліфікаційної роботи є дослідження та визначення найбільш ефективного підходу до моніторингу та аналізу транзакцій з рунами (Runes) у мережі Bitcoin. У рамках цього дослідження необхідно з'ясувати оптимальний набір інструментів для декодування специфічних метаданих руничних транзакцій, порівняти ефективність спеціалізованих і універсальних бібліотек, а також визначити напрями можливого вдосконалення існуючих технологій для більш якісного аналізу інформаційних потоків у мережі Bitcoin.

Для досягнення поставленої мети необхідно проаналізувати сучасний стан інструментів для роботи з транзакціями Bitcoin, які підтримують або можуть підтримувати роботу з рунами (Runes). Провести порівняльний аналіз існуючих інструментів (зокрема, Runestone.ts або runestone-lib та bitcoinjs-lib), визначивши їх переваги, недоліки та потенційні обмеження для роботи з рунами. Розробити та протестувати прототип спеціалізованого модуля моніторингу та декодування транзакцій з рунами, базуючись на технології Runestone.ts. Оцінити ефективність роботи створеного прототипу у порівнянні з існуючими рішеннями за критеріями швидкодії, точності та зручності інтеграції.

У процесі дослідження застосовані наступні методи та підходи:

– аналіз та порівняння існуючих бібліотек (Runestone.ts, bitcoinjs-lib та інші);

– створення прототипу модуля на основі бібліотеки Runestone.ts;

– експериментальне тестування розробленого модуля із застосуванням реальних даних Bitcoin-транзакцій;

– статистична обробка отриманих результатів для визначення ефективності роботи модуля.

Об'єктом дослідження виступає мережа Bitcoin (BTC).

Предметом дослідження є процес моніторингу та аналізу транзакцій, що містять руни (Runes), а також процес управління інформаційними потоками у контексті їхнього декодування та подальшої аналітичної обробки.

Функції, що планується автоматизувати та вдосконалити:

– автоматизоване виявлення транзакцій, що містять руничні повідомлення;

– декодування транзакцій та виділення структурованих руничних даних (identifier (ID) токенів, кількість токенів, тип дії тощо);

– автоматизований контроль та верифікація транзакцій з рунами на предмет валідності (баланси, історія випуску та передач токенів);

– формування агрегованих даних та автоматичний збір статистики про обіг токенів у мережі Bitcoin;

– створення зручного інтерфейсу для моніторингу і аналізу транзакцій з рунами.

У результаті виконання роботи очікується отримати підвищення ефективності процесу аналізу транзакцій, що містять руни; значне прискорення роботи системи моніторингу за рахунок автоматизації рутинних завдань; підвищення точності і безпомилковості при декодуванні та інтерпретації руничних даних; забезпечення належного рівня захищеності інформації за рахунок автоматизованої перевірки автентичності і валідності руничних транзакцій.

Потенційні користувачі це трейдери та аналітики, які працюють з блокчейн-рішеннями на основі Bitcoin; проєкти та стартапи, що впроваджують протокол Runes для реалізації токенів на мережі Bitcoin; біржі, сервіси та платформи для управління цифровими активами, які зацікавлені у інтеграції функціональності моніторингу та аналізу Runes.

1 ОПИС ТА АНАЛІЗ СТРУКТУРНИХ І ФУНКЦІОНАЛЬНИХ ОСОБЛИВОСТЕЙ МОНІТОРИНГА ТА АНАЛІЗА ВЗАЄМОДІЙ З РУНАМИ У МЕРЕЖІ BITCOIN

1.1 Опис особливостей функціонування об’єкту автоматизації

Об’єктом дослідження в кваліфікаційній роботі виступає Bitcoin мережа. Bitcoin – це децентралізована блокчейн-мережа, в якій транзакції об’єднуються в блоки, а обіг цифрових монет відстежується через модель UTXO (не витрачених виходів транзакцій). Кожна транзакція має вхід(и), що посилаються на попередні виходи, та вихід(и), які можуть містити певну кількість BTC і скрипт, що визначає умови витрачання. Структура біткоїн-транзакції також дозволяє вбудовувати довільні дані – зокрема, через спеціальний службовий вихід з опкодом OP_RETURN, який позначає, що вміст цього виходу не використовуватиметься як валюта [1]. OP_RETURN-вихід зазвичай містить до 80 байт даних і слугує для запису метаданих у блокчейн.

Runes використовують ці базові принципи для реалізації цифрових активів (токенів) на Bitcoin без окремого блокчейну чи сайдчейну. Протокол визначає формат повідомлення під назвою runestone (рунічний камінь), яке розміщується саме в OP_RETURN-виході транзакції [2]. Кожна транзакція може містити не більше одного runestone. Формат скрипту такого виходу починається з OP_RETURN OP_13, після чого йдуть один або кілька блоків даних (push bytes), що інтерпретуються як послідовність 128-бітних чисел, які потім парсуються у структуру runestone. Такий підхід забезпечує компактне кодування інформації про токени у межах стандартної біткоїн-транзакції. Графічне відображення можна побачити на рисунку 1.1.

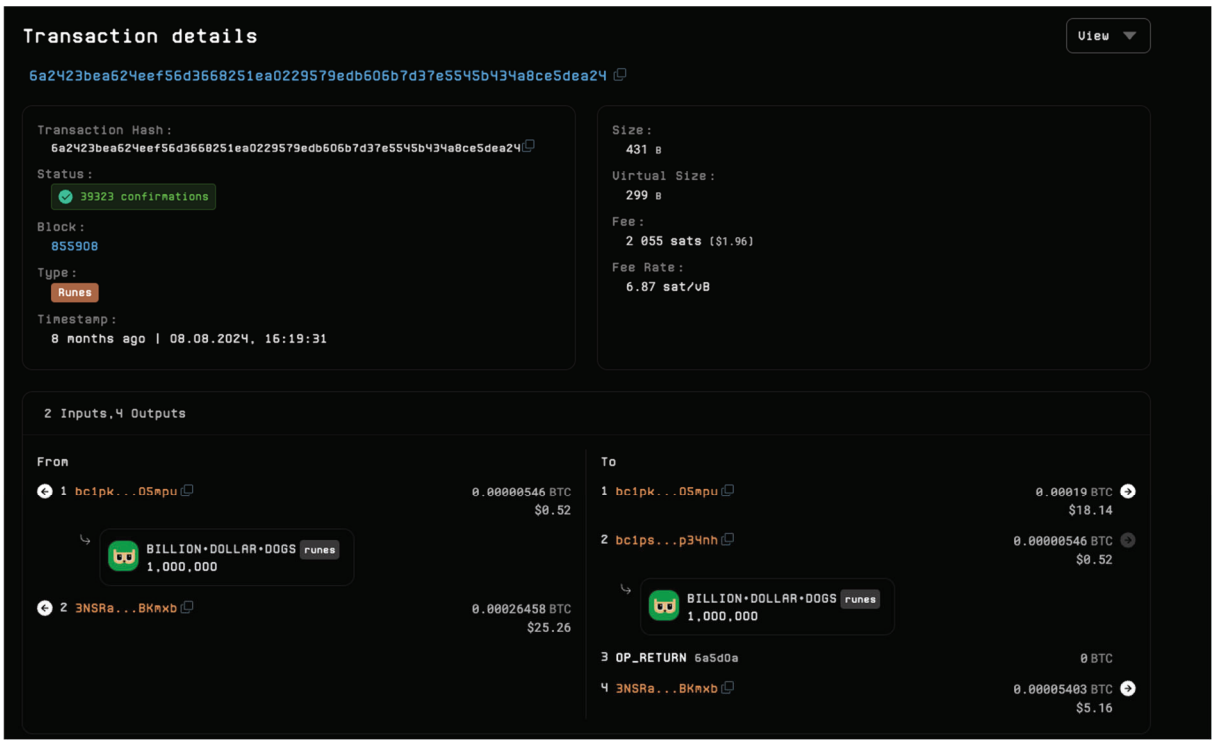

Рисунок 1.1 – графічне відображення транзакції з руної в блокчейні Bitcoin

Runestone-повідомлення визначає дію з токеном. Протоколом Runes передбачено три основні функції: Etching (гравіювання) – створення нового токена; Minting – додаткова емісія існуючого токена; Transfer (через структуру, що зветься Edict) – передача певної кількості токенів від входів до виходів транзакції [2]. При гравіюванні відбувається «народження» нової руни: транзакція, що містить runestone з полем etching, визначає унікальний ID токена та його незмінні властивості. Зокрема, при створенні нової руни задається її назва(послідовність літер A-Z до 26 символів, з опціональними роздільниками «•» для покращення читабельності), подільність –скільки десяткових розрядів допускається для часток токена (наприклад, 0 означає неподільний токен, 2 – це максимальна точність до сотих частин), а також можливо символ (умовне позначення валюти) та інші параметри (наприклад, гранична кількість токенів чи попередній премайн) [2]. Новоствореному токену присвоюється ідентифікатор RuneId, що складається з номера блоку і номера транзакції, в якій цей токен було вигравірувано (формат BLOCK:TX). Наприклад, якщо токен etched у блоці 500, двадцятою

транзакцією, його ID буде 500:20. Цей ID надалі використовується для посилання на токен при виконанні емісії чи передач.

Minting дозволяє збільшити кількість одиниць вже існуючої руни. Транзакція з runestone, що містить поле mint (значення якого – RuneId відповідної руни), створює нові токени цього типу і зараховує їх на виходи транзакції. Важливо, що протокол може передбачати обмеження на емісію (cap) або інші умови в рамках параметрів, заданих при гравіюванні (наприклад, terms можуть визначати графік чи ліміт випуску). Передача токенів реалізується за допомогою однієї або декількох структур Edict у повідомленні runestone. Кожен Edict містить посилання на RuneId токена, кількість токенів для передачі та індекс виходу, на який слід зарахувати цю кількість [2]. Сукупність таких «наказів» виконується послідовно – протокол інтерпретує їх як зменшення балансу певної руни на стороні входів транзакції і збільшення на вказаних виходах. Якщо вихід не зазначений явно, за замовчуванням токени надходять на перший регулярний вихід (перед OP_RETURN).

В результаті, один runestone може переносити декілька різних токенів одночасно і розподіляти їх між кількома адресами, що підвищує ефективність – на відміну від Bitcoin Request for Comment 20 (BRC-20), де для однієї передачі нерідко потрібно кілька транзакцій, що є застарілим [3]. Крім того, вихідні UTXO можуть містити баланси кількох різних рун одночасно, оскільки в протоколі Runes токени «прив’язані» до виходів транзакцій, але не вимагають окремого виходу для кожного виду токена [2]. Ця властивість зменшує «навантаження» на блокчейн за рахунок уникнення надлишкових виходів і підвищує масштабованість рішення.

Таким чином, предметна область дослідження охоплює як базові принципи роботи блокчейну Bitcoin (модель UTXO, транзакції та скрипти), так і спеціалізований протокол Runes, що накладається на ці принципи для реалізації цифрових активів. Розуміння структури runestone-повідомлень та правил обігу рун є критичною основою для подальшого розроблення

інструментів їх аналізу. У межах цього дослідження важливо враховувати, що будь-яка обробка руничних транзакцій має виконувати валідацію послідовності дій: наприклад, перевіряти, що при передачі токенів відправник дійсно мав відповідний баланс (це потребує відстеження попередніх транзакцій з даною руною аж до її генезису). В протоколі введено поняття «cenotaph» – збитковий або некоректно сформований runestone, який не несе дійсної інформації [2]. Аналітичні інструменти повинні вміти розпізнавати та ігнорувати подібні випадки, забезпечуючи коректне трактування лише валідних руничних даних.

Наведені процеси накладають специфічні вимоги до точності, швидкості обробки, зручності інтеграції та надійності аналітичних інструментів, виконання яких потребує використання сучасних технологій для декодування блокчейн-даних (наприклад, Runestone.ts, bitcoinjs-lib) та розробки спеціалізованого модуля моніторингу та аналізу rune-транзакцій, здатного автоматизувати верифікацію і статистичну обробку цих даних [4], [5].

1.2 Опис організаційної структури бізнес процесів інформаційної технології блокчейн IC Bitcoin

За організаційною структурою блокчейна Bitcoin поділяється на такі компоненти:

- вільні розробники (Open-source Developers);
- вузол мережі (Full Node);
- черга очікування транзакцій (Mempool);
- майнери (Miner);
- RPC-інтерфейс вузол (RPC Node);
- гаманець;

– користувач;

– біржи, інші сервіси з обміну (exchanges);

– блокчейн-експлорери (Explorers).

Схема організаційної структури блокчейна Bitcoin представлена на рисунку 1.2.

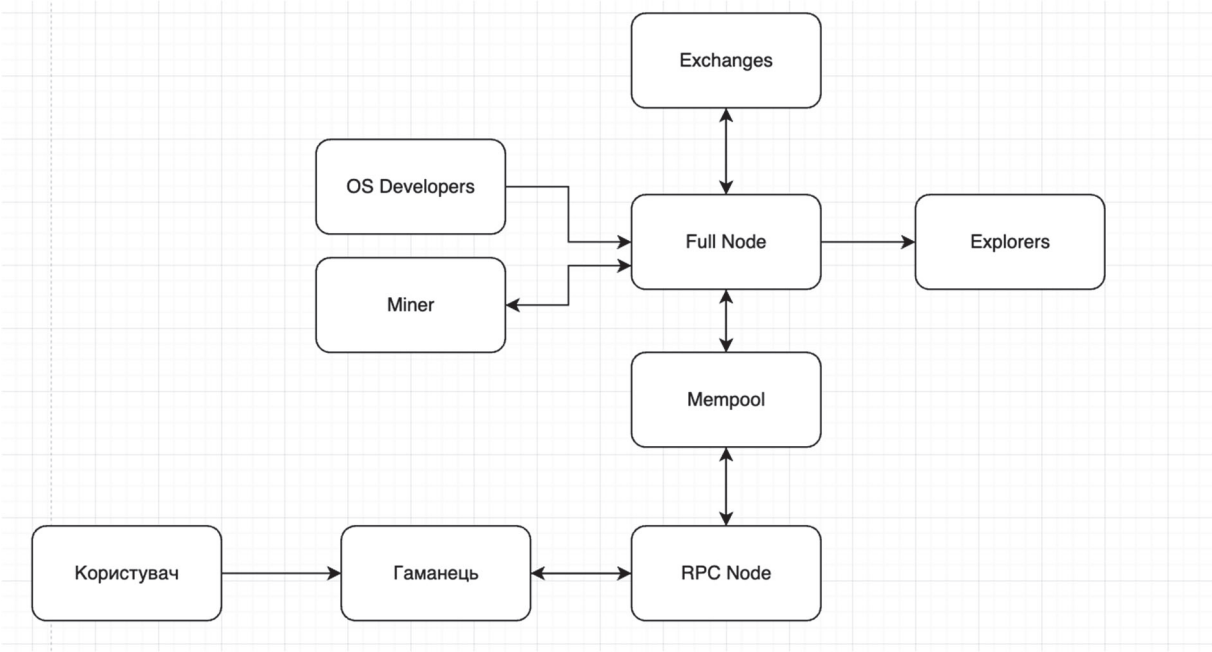

Рисунок 1.2 – Схема організаційної структури блокчейна Bitcoin

OS Developers (вільні розробники) відповідають за розробку й підтримку відкритого вихідного коду Bitcoin-клієнта (наприклад, Bitcoin Core та інших реалізацій), впровадження нових Bitcoin Improvement Proposal (BIP) пропозицій із подальшим їхнім тестуванням і випуском релізів для повних вузлів, моніторинг безпеки із оперативним випуском патчів у разі виявлення вразливостей (зокрема в бібліотеках криптографії), а також за забезпечення зворотного зв’язку з мережею - обговорення та узгодження реалізації нових функцій протоколу [6]. Окрім цього, вони документують і готують інструкції для операторів вузлів (інструкції з розгортання, налаштування конфігурацій).

Full Node (вузол мережі) приймає й валідовує транзакції, що надходять із мережі Peer-to-Peer, перевіряючи правильність структури транзакції

(формат, підписи, наявність UTXO), відкидаючи некоректні або підозрілі транзакції (double-spend, неправильні підписи). Він формує локальну копію блокчейну (Blockchain DB), зберігаючи послідовність блоків із їхніми заголовками (Block Header) та даними транзакцій, а після додавання нового блоку оновлює набір незатребуваних виходів (UTXO) [1]. Одночасно Full Node поширює валідні транзакції іншим вузлам (flooding) і приймає нові згенеровані блоки від майнерів, перевіряє їхню коректність і додає до локальної копії ланцюга. Крім того, Full Node забезпечує інтерфейс для сторонніх компонентів – він підключається до бірж (Exchanges) для перевірки стану блокчейну і контролю балансу адрес, до експлорерів (Explorers) для індексації та аналітики, а також обслуговує запити RPC Node, що дозволяє віддаленим сервісам і гаманцям отримувати інформацію про блокчейн [4].

Mempool (черга непідтверджених транзакцій) тимчасово зберігає всі валідні, але ще не включені в блок транзакції, оптимізує чергу з урахуванням пріоритету (розмір transaction (TX), розмір комісії: вищий sats/байт дає вищу позицію в мемпулі) та забезпечує майнерів достатньою кількістю транзакцій для формування нового блоку [1]. Крім того, мемпули синхронізуються між усіма Full Node так, що коли нова транзакція надходить на один вузол, вона реплікується між його сусідами, забезпечуючи рівномірне розповсюдження TX.

Miner (майнер) підключається до одного або кількох Full Node, щоб регулярно отримувати оновлений мемпул (список транзакцій для блоку) та формує [1]:

– власний блок-хедер із полем PrevBlockHash (посилання на останній блок локальної ланки);

– Merkle Root (хеш-корінь дерева Merkle з усіх транзакцій у блоці);

– Timestamp (відмітка часу);

– Bits (поточна цільова складність);

– Nonce (число для Proof-of-Work).

Далі майнер запускає алгоритм Proof-of-Work, ітеруючи різні значення Nonce (іноді змінюючи Timestamp або включаючи «dummy» вихід у coinbase), поки подвійний SHA-256 хедеру не стане меншим за цільовий. Після успішного знаходження валідного хешу майнер формує повний блок із усіма TX і блок-хедером, подає його у мережу через Full Node, а в нагороду отримує coinbase (нові монети) і комісії, які потрапляють у його власний UTXO (доступні для витрачання після maturity).

RPC Node (інтерфейс віддалених викликів) забезпечує можливість взаємодії з Full Node для гаманців і зовнішніх сервісів через JSON-RPC, наприклад, методи getbalance, sendrawtransaction, getblock та інші [4]. Він виступає посередником – гаманець формує й підписує raw TX, надсилає його на RPC Node, який перекидає TX до Full Node, а натомість Full Node повертає через RPC Node результати запитів щодо підтверджень, стану адрес та блоків. RPC Node можна налаштувати локально на потужностях користувача або використовувати як віддалений Application Programming Interface (API) провайдер.

Гаманець (Wallet) займається генерацією seed-фрази (BIP39) для створення приватних ключів і відповідних адрес (BIP44/BIP84), відповідальним зберіганням приватних ключів (шифрування локального сховища, Hardware Security Module або апаратний гаманець) та формуванням транзакцій – він отримує від RPC Node список UTXO для адрес користувача, обирає їх за допомогою алгоритму Coin Selection (щоб мінімізувати суму комісій і оптимально використати входи), обчислює розмір майбутньої транзакції та рекомендовану суму комісії (fee rate), після чого формує внутрішню структуру TX (inputs, outputs, change) [1]. Далі гаманець за допомогою приватного ключа накладає ECDSA-підпис (secp256k1) на кожен input і створює raw-hex-string транзакції, яку передає у мережу викликом sendrawtransaction через RPC Node до Full Node [7]. Після цього гаманець періодично опитує getmempoolentry чи getrawmempool, щоб перевірити, чи потрапила транзакція у мемпул, а також стежить за кількістю

підтверджень (confirmations) через gettransaction або getrawtransaction із verbose. Усе це відображається користувачеві – гаманець показує баланс адрес, історію транзакцій, кількість підтверджень, а також надає можливість генерувати нові адреси, резервні копії, імпортувати чи експортувати приватні ключі.

Користувач безпосередньо взаємодіє з гаманцем:

– формує запити на відправку коштів;

– контролює оновлення балансу;

– стежить за вхідними й вихідними транзакціями.

Він приймає рішення щодо розміру плати за транзакцію (Low, Medium, High fee rate), може використовувати спеціальні функції, як-от Replace-By-Fee (RBF) або Child-Pays-For-Parent (CPFP), а також піклується про безпеку своїх коштів — зберігає seed-фразу чи приватні ключі у надійному місці, за потреби використовує апаратний гаманець або мультипідписи (Multisig) [4], [8].

Exchanges (обмінники) підключені до власних Full Node для швидкої верифікації депозитів і виводів BTC, відстежують події блокчейну в режимі реального часу (нові блоки, зміни статусу транзакцій), обробляють депозити клієнтів (коли користувач надсилає BTC на внутрішню адресу біржі, Full Node валі-дує це, а біржа зараховує кошти до балансу «on-exchange») і формують транзакції на вивід (Exchange підписує TX і відправляє її через свій Full Node). Також обмінники реалізують Know Your Customer/Anti-Money Laundering процедури (верифікація користувачів, блокування підозрілих адрес, співпраця з регуляторами) та підтримують ліквідність і маркет-мейкерські операції (гарантують достатній рівень ордер-буку) [3].

Explorers (блокчейн-експлорери) займаються індексацією та агрегацією даних – отримують від Full Node всю інформацію про блоки, транзакції й адреси, будують власну базу індексів, яка дозволяє шукати TX за хешем, перевіряти статус підтверджень тощо. Вони надають публічний API і веб-інтерфейс, що дає змогу користувачам та стороннім розробникам

отримувати історію транзакцій, баланс адрес, статистику мережі (хешрейт, difficulty, середні комісії). Крім того, експлорери виконують аналітику [9]:

– відображають графіки зміни труднощі майнінгу або хешрейту;

– розміру блоків;

– середньої плати за ТХ, показують стан мемпулу (кількість ТХ, середня комісія);

– інформують про великі транзакції чи тренди (наприклад, аномальний сплеск активності).

1.3 Аналіз бізнес-процесу декодування транзакцій з Runes та відстеження їх.

Процес декодування та відстеження транзакцій з Runes у мережі Bitcoin полягає в систематичному аналізі транзакцій для отримання структурованих даних, необхідних для подальшого моніторингу та аналітики.

Загалом процес декодування транзакцій організований наступними етапами [10]:

– отримання сирих даних транзакцій (OP_RETURN, UTXO, ідентифікатори);

– первинне розпізнавання транзакцій, що містять Runes;

– декодування runestone-повідомлень;

– класифікація транзакцій за типами (Etching, Minting, Transfer);

– перевірка валідності транзакцій (баланси, історія попередніх транзакцій);

– агрегування інформації для формування статистичних звітів та аналітики.

Для реалізації цього процесу існують кілька технологічних рішень, які

забезпечують взаємодію з блокчейном Bitcoin:

– Bitcoin Core RPC - надає доступ до сирих даних транзакцій, але не виконує автоматичної інтерпретації метаданих протоколу Runes [4];

– bitcoinjs-lib - універсальна бібліотека для роботи з транзакціями Bitcoin, не підтримує специфічний формат runestone, вимагає додаткового написання логіки декодування [11];

– runestone-lib, спеціалізована бібліотека для роботи з протоколом Runes, забезпечує базове декодування runestone-структур, однак має обмежені можливості щодо моніторингу, верифікації стану та агрегування інформації [5].

Для порівняння зазначених технологій сформовано порівняльну таблицю 1.1.

Таблиця 1.1 – порівняння зазначених технологій для декодування транзакцій з рунами.

Критерій оцінки	Bitcoin RPC	bitcoinjs-lib	runestone-lib
Автоматичне декодування Runes	Відсутнє	Відсутнє	Часткове (лише декодування)
Зручність інтеграції	Низька	Середня	Висока
Підтримка типів транзакцій	Немає	Часткова (необхідна доробка)	Часткова (необхідна доробка)
Верифікація та перевірка стану	Відсутня	Відсутня	Обмежена
Формування статистичних звітів	Відсутнє	Відсутнє	Обмежене
Швидкодія при обробці	Середня	Висока	Висока

Виходячи з аналізу існуючих технологій, можна зробити висновок, що найбільш придатною для реалізації модуля моніторингу та аналізу транзакцій з Runes є бібліотека runestone-lib. Вона надає необхідні функції для декодування runestone-повідомлень, проте потребує вдосконалення у частині інтеграції процесів моніторингу, верифікації транзакцій та формування аналітичних звітів.

Таким чином, для ефективного виконання поставлених завдань необхідна модернізація технології runestone-lib шляхом розширення її функціональності для комплексного моніторингу та автоматизації аналітичних процесів.

2 ОГЛЯД Й АНАЛІЗ ІСНУЮЧИХ СИСТЕМ АВТОМАТИЗАЦІЇ ДЕКОДУВАННЯ ТА АНАЛІЗУ ТРАНЗАКЦІЙ З РУНАМИ

Попри появу перших реалізацій протоколу Runes, наразі відсутні комплексні засоби для зручного відстеження і аналізу руничних транзакцій. Існуючі інструменти можна умовно поділити на універсальні бібліотеки загального призначення та спеціалізовані рішення для Runes. До перших належать, зокрема, згаданий bitcoinjs-lib (JavaScript-бібліотека для формування і розбору біткоїн-транзакцій) та RPC-інтерфейси Bitcoin-вузлів. Ці засоби здатні отримати базову інформацію про транзакцію (структуру входів/виходів, скрипти у вигляді hex-рядка, тощо), але не надають готових методів для інтерпретації протоколу Runes. Розробнику, що прагне відстежувати рух токенів, довелося б вручну обробляти OP_RETURN-виходи, парсити байти та відновлювати логіку розподілу токенів. Це є нетривіальним завданням і джерелом потенційних помилок.

Як зазначено вище, універсальна природа бібліотеки bitcoinjs не гарантує правильної інтерпретації спеціалізованих структур – зокрема, може виникати зниження точності декодування руничних даних без належної підтримки протоколу. Так само, пряме використання Bitcoin Core RPC видає лише «сирі» дані (наприклад hexadecimal (HEX) транзакції), залишаючи всю обробку на стороні користувача. В результаті, аналіз взаємодій з рунами суто засобами загального призначення є обмеженим і незручним, немає простого способу швидко дізнатися, які саме токени передаються в транзакції, скільки їх, хто емітент і чи валідна ця передача згідно історії токена. Спеціалізовані бібліотеки та прототипи, що з’явилися разом із зародженням Runes, значно спрощують технічну сторону роботи з рунами. Одним із перших рішень стала бібліотека Runestone.ts – реалізація інтерпретатора runestone-повідомлень, названа на честь протоколу [5]. Зокрема, Runestone.ts дозволяє декодувати «руничні» виходи транзакцій, виділяючи структуровані дані (ID

руни, кількість, тип дії тощо), а також може генерувати ці виходи для формування нових транзакцій з токенами.

Подібні можливості надає і бібліотека RuneLib (TypeScript-реалізація від компанії sCrypt), яка фактично лежить в основі Runestone.ts, вона забезпечує функції decipher() для розбору runestone зі строкового коду транзакції та encipher() для зворотного кодування повідомлення [12]. На основі RuneLib було побудовано утиліти для роботи з токенами – наприклад, btc-rune-toolbox з набором скриптів для etching (створення) нових рун, minting додаткових монет, transfer і масової розсилки (airdrop) токенів. Спілкою Ordinals, у своєму репозиторії (проєкт Casey Rodarmor) з'явилися структури даних та початковий варіант індексатора для Runes, а компанія Magic Eden опублікувала бібліотеку runestone-lib з подібним функціоналом в open-source [5]. Це свідчить про потребу зацікавлених сторін у розвитку екосистеми навколо Runes [13], [14].

Попри наявність згаданих бібліотек, існують суттєві обмеження, що мотивують розробку нового модулю аналізу.

Відсутність інтегрованих засобів моніторингу, тобто наявні рішення здебільшого постачають функції низького рівня – декодування або формування runestone-повідомлення – і призначені для використання розробниками. Вони не забезпечують повноцінного інтерфейсу моніторингу в реальному часі (на кшталт готового експлорера чи аналітичної панелі). Тобто, щоб відстежувати всі руничні транзакції в мережі, потрібно самостійно організувати цикл: підключення до вузла Bitcoin, фільтрацію транзакцій з OP_RETURN, виклик decipher() для кожної, збереження результатів у базі та візуалізацію – готового рішення «під ключ» наразі не існує. Лише нещодавно з'явилися перші веб-експлорери (OKLink, Ordiscan, Unisat та ін.) для перегляду інформації про руни, але їх внутрішні індексатори є пропрієтарними і недоступні для гнучкого використання дослідниками. Таким чином, досі актуальне завдання – створити модуль, який легко інтегрується з Bitcoin-вузлом та автоматизує збір, декодування і

агрегування даних про токени.

Обмежені аналітичні можливості існуючих бібліотек, інструменти на кшталт Runestone.ts або RuneLib не зберігають стан балансу токенів. Вони розбирають окрему транзакцію, але не відслідковують зв’язок з попередніми (наприклад, чи достатньо було токенів на входах для виконання Edict, який баланс залишається на виходах тощо) [12]. Відповідно, для глибшого аналізу (побудови історії обігу руни, підрахунку її загальної пропозиції, виявлення активності адрес) потрібно дописувати додаткову логіку. Немає також штатних засобів класифікації транзакцій за типами (Etching/Minting/Transfer) окрім як вручну перевіряти поля розпарсеної структури. У спеціалізованих бібліотеках відсутній високорівневий API для типових запитів на зразок «отримати всі транзакції випуску даного токена» або «отримати поточний баланс руни на заданій адресі» – такі функції має забезпечити окремий модуль аналізу.

Проблеми інтерпретації та верифікації руничних даних, хоча Runestone.ts значно спрощує декодування, відповідальність за коректну інтерпретацію покладається на розробника. Потрібно враховувати спеціальні випадки, визначені протоколом – зокрема, pointer (вказівник на нестандартний «основний» вихід) або burn-транзакції (спалення токенів на OP_RETURN-виході). Без належної перевірки можна дійти хибних висновків щодо переміщення токенів. Наприклад, якщо транзакція містить некоректно сформований runestone (cenotaph), бібліотека може повернути помилку або порожній результат, і це слід обробляти окремо. Так само, як наголошувалося [9], можливі випадки навмисного внесення недостовірних даних – лише читаючи окрему транзакцію, не можна достеменно знати, чи була вона легітимною операцією з токеном – для цього необхідно перевірити її в контексті всього ланцюжка (чи існував токен з вказаним ID, чи не перевищено випуск, чи виходи збігаються з адресами отримувачів тощо). Наявні бібліотеки не проводять такої перевірки автоматично, тож спеціалізований модуль має взяти на себе і цю функцію.

Перелічені фактори підкреслюють необхідність розробки власного програмного модуля для моніторингу та аналізу Runes-транзакцій. Очікується, що такий модуль поєднає в собі точність спеціалізованого декодування і зручність інтеграції в ширші системи. Проведений попередній аналіз показав, що використання Runestone.ts як ядра для декодування дає значні переваги над універсальними підходами.

3 ОПИС ВИМОГ ДО ОБ'ЄКТА РОЗРОБКИ. ФОРМУЛЮВАННЯ ЗАВДАННЯ РОЗРОБКИ

3.1 Опис функціональних вимог до об'єкта розробки

В рамках кваліфікаційної роботи розглядається бізнес-процес моніторингу та аналізу взаємодій з рунами (Runes) у мережі Bitcoin.

Об'єктом розробки виступає модуль «Моніторинг та аналіз взаємодій з рунами», який інтегрується до загальної інформаційної системи для автоматизації процесів моніторингу, декодування та аналізу транзакцій, пов'язаних із цифровими активами (рунами).

Розроблюваний модуль повинен забезпечувати автоматизацію таких функцій:

– автоматичне визначення транзакцій, що містять runestone-повідомлення;

– декодування runestone-даних із транзакцій Bitcoin;

– класифікація транзакцій за типами (Etching, Minting, Transfer, Sell, Buy);

– автоматична перевірка валідності та правильності даних транзакцій з урахуванням історії операцій;

– збір та зберігання інформації про транзакції, баланси рун та історії їх передач;

– формування та генерація аналітичних звітів (наприклад, про кількість токенів, активність гаманців, топ-гаманці за прибутковістю, динаміка цін тощо).

До процесу моніторингу та аналізу транзакцій входять наступні функції:

– отримання та обробка сирих даних транзакцій (OP_RETURN, UTXOs, ідентифікатори транзакцій);

– декодування runestone-повідомлень;

– зберігання розшифрованих та агрегованих даних у базі;

– забезпечення можливості вибору параметрів для генерації аналітичних та статистичних звітів;

– виведення згенерованих звітів на екран користувача.

Генерація аналітичних звітів включає такі функції:

– вибір періоду для аналізу;

– вибір сортування звіту (кількість токенів, активність гаманців, ціна токенів, топ-гаманці за прибутковістю тощо);

– отримання оброблених даних відповідно до обраних параметрів;

– формування та візуалізація звітів.

Користувачами модуля є:

– розробники блокчейн-додатків, що використовують протокол Runes;

– аналітики та трейдери, які проводять моніторинг цифрових активів;

– адміністратори інформаційних систем, які здійснюють управління блокчейн-інфраструктурою.

Розробники блокчейн-додатків мають можливість інтеграції модуля з власними системами та доступ до всіх функцій моніторингу, декодування та аналізу даних.

Аналітики та трейдери отримують доступ до аналітичних звітів та статистичних даних, що дозволяють приймати обґрунтовані рішення щодо цифрових активів.

Адміністратори відповідають за налаштування модуля, підтримку його функціонування, актуалізацію стандартів і правил декодування даних та здійснюють контроль за безперебійною роботою інструментів.

Таким чином, розробка модуля «Моніторинг та аналіз взаємодій з рунами» забезпечить комплексну автоматизацію ключових бізнес-процесів, необхідних для ефективного моніторингу та аналізу транзакцій з рунами у мережі Bitcoin.

3.2 Опис нефункціональних вимог до об’єкта розробки

Для побудови бази данних (БД) модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» має використовуватися реляційна модель.

СУБД, яка буде використовуватися в рамках розроблюваного модуля, має бути оптимезованою та швідкою.

Час отримання будь-яких даних з БД не має перевищувати 2 секунд. Максимальна затримку під час формування звіту може бути не більше 5 секунд. Час реагування системи не має перевищувати 1 секунду.

Для роботи модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» використовується наступне інформаційне забезпечення (ІЗ):

– документація мережі Bitcoin;

– документація Рун.

Програмне забезпечення (ПЗ) модуля клієнтської частини має:

– підключатися до віддаленого серверу;

– мати підтримку щонайменьш 2 найбільш відомих гаманців(Unisat, Xverse);

– бути доступним у будь-якому браузері.

ПЗ модуля серверної частини повинно мати:

– встановлену ОС Linux;

– оперативну пам’ять не менше 32 ГБ;

– постійну пам’ять не менше 4 ТБ.

Розроблюваний модуль “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin», використовує клієнт-серверна архітектура.

З метою захисту від несанкціонованого доступу система має надавати можливість авторизації за визначеними ролями, завдяки чому кожний користувач буде мати доступ лише до визначеного своєю роллю

функціоналу. До процесу авторизації користувачів входять наступні функції:

– система має відобразити кнопку для підключення гаманців;

– система має відобразити вікно гаманця, що підключене;

– система має брати підпис гаманця, щоб удостовіритись що це володій гаманця;

– система має ідентифікувати користувача за номером гаманця;

– система має надати визначені роллю користувача доступи до системи при успішній авторизації.

Розроблюваний модуль має отримувати необхідні оновлення, які будуть направлені на виправлення помилок функціональних і нефункціональних, а також розширювати і надавати нові можливості користувачам модуля. Для встановлення оновлень модуля буде використовуватися система автоматичних оновлень (Github) для забезпечення безперервного процесу оновлення, що дозволить користувачам отримувати нові версії без необхідності виконання дій вручну.

Технічна підтримка має бути налаштована таким чином, щоб забезпечити максимальну зручність для користувачів.

Користувачі можуть надіслати свої побажання щодо покращення роботи електронною поштою. Також планується налаштувати вбудовані інструменти для звітування про помилки.

Технічне обслуговування відбуватиметься щомісяця з інформуванням користувачів заздалегідь за допомогою сповіщення на електронну пошту.

3.3 Обґрунтування мети та критеріїв ефективності об’єкта розробки

Модуль “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» призначений для автоматизації декодування, визначення, первинного аналізу, а також формування відповідних звітів.

Процес декодування і первинного аналізу наразі не автоматизований.

Шляхом аналізу при формуванні завдання для створення та подальшого використання модуля було виділено наступні критерії ефективності:

– швидкість виконання декодування транзакції;

– наявність запису декодованих та проаналізованих даних у БД;

– компактність БД, завдяки правильно побудованої архітектури;

– швидкість запису та доступу до даних в БД;

– наявність захисту від несанкціонованого доступу до серверної частини модуля;

– зручність користувацького інтерфейсу (забезпечення високого рівня задоволеності користувачів за рахунок зручного та інтуїтивного інтерфейсу);

– здатність масштабуватися (модуль може бути масштабований для підтримки зростаючої кількості користувачів та обсягів даних);

– модульність та розширюваність (структура модуля повинна дозволяти легке додавання нового функціоналу та інтеграцію з іншими сервісами);

– забезпечення енергоефективності (оптимізація використання комп’ютерних ресурсів задля зниження енергоспоживання та підвищення загальної ефективності модуля).

Впровадження модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» сприятиме покращенню якості виконання аналізу та забезпечить їх швидше виконання, що дасть можливість приймати рішення користувачам швидше, або використовувати меньші ресурси за для цього.

4 ОПИС АРХІТЕКТУРИ МОДУЛЯ “МОНІТОРИНГ ТА АНАЛІЗ ВЗАЄМОДІЙ З РУНАМИ” У МЕРЕЖІ BITCOIN

Об’єктом розробки і дослідження є модуль “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin».

Метою розробки даного модуля є:

– збільшення ефективності декодування та аналізу;

– підвищення точності та швидкості формування первичного аналізу;

– підвищення точності декодування і перевірки валідності транзакцій;

– забезпечення зрозумілого та інформативного інтерфейсу для користувачів.

Контекстна діаграма модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» представлена на рисунку 4.1.

На рисунку 4.2 зображена декомпозиція першого рівня модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin».

Вхідними даними для функціонування модуля є:

– дані транзакції(OP_RETURN, ID, UTXOs);

– запит за Profit and Loss (PnL);

– запит за назвою руни.

Управління модуля та обмеження його використання здійснюються з використанням:

– документації мережі Bitcoin,

– протоколом Runestone.

Механізмами, які позначені на рисунках 4.2 і 4.3, є:

– трейдер;

– аналітик;

– інвестор.

Вихідними повідомленнями модуля є:

– дані про руни;

– інформація гаманців;

– топ гаманців за PnL;

– дані за взаємодією (продаж/купівля).

У межах модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» проводиться вирішення наступних задач:

– декодування транзакцій;

– обробка та збереження у БД даних за взаємодією рун всіх гаманців;

– первинний аналіз даних;

– формування вихідних форм для огляду результата.

Модуль працюватиме 24 годин на добу, найбільше навантаження очікується в проміжку часу з 8:00 до 22:00.

Умови припинення вирішення задач модуля автоматизованим способом:

– відсутність електричного постачання;

– відсутність підключення до мережі Інтернет;

– відсутність підключення до БД;

– відсутність підключення до Bitcoin RPC;

– некоректність роботи програми;

– поломка комп’ютерного обладнання.

У таблицях 4.1 та 4.2 наведений перелік та опис вихідних та вхідних повідомлень «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin».

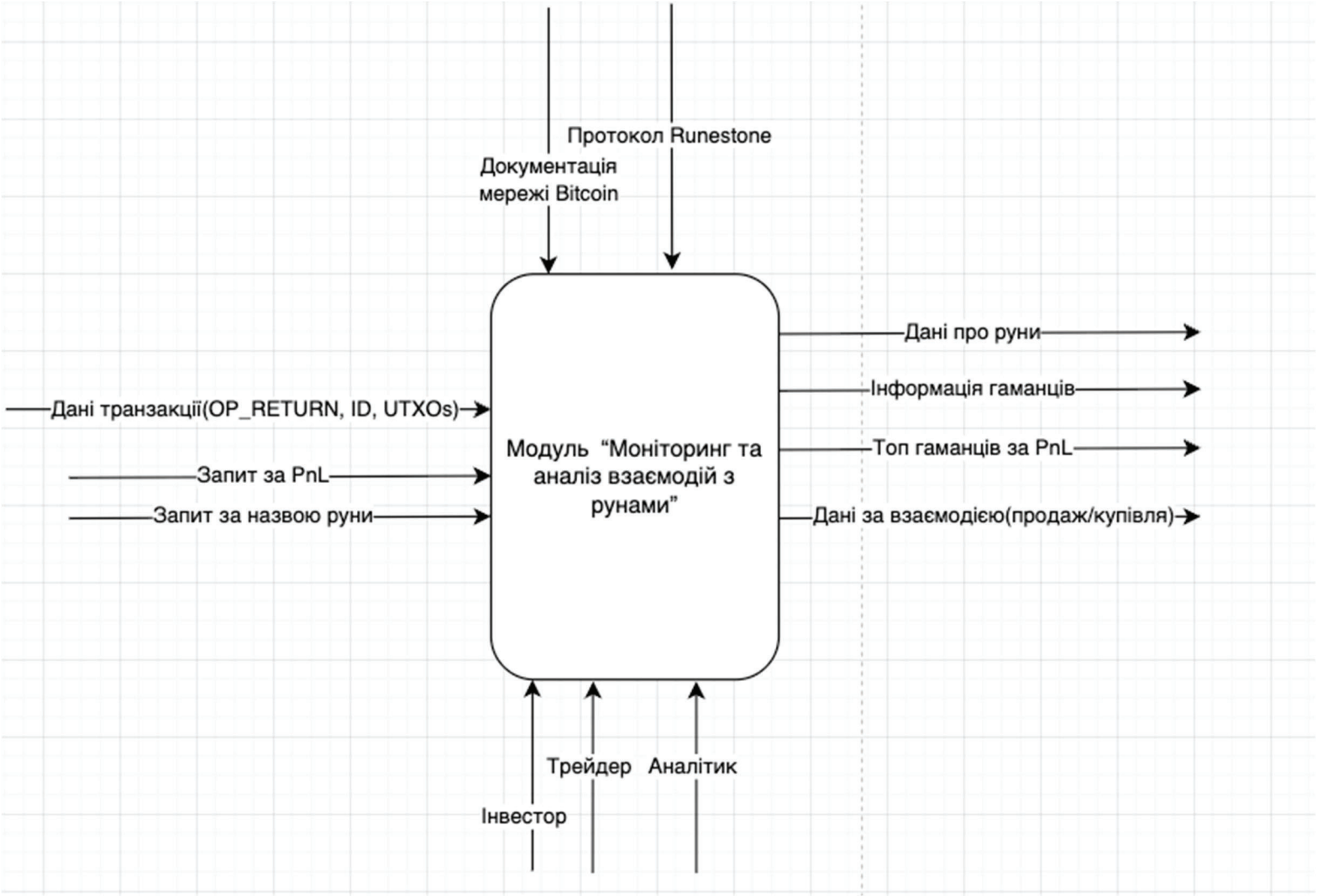

Рисунок 4.1 – Контекстна діаграма модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin»

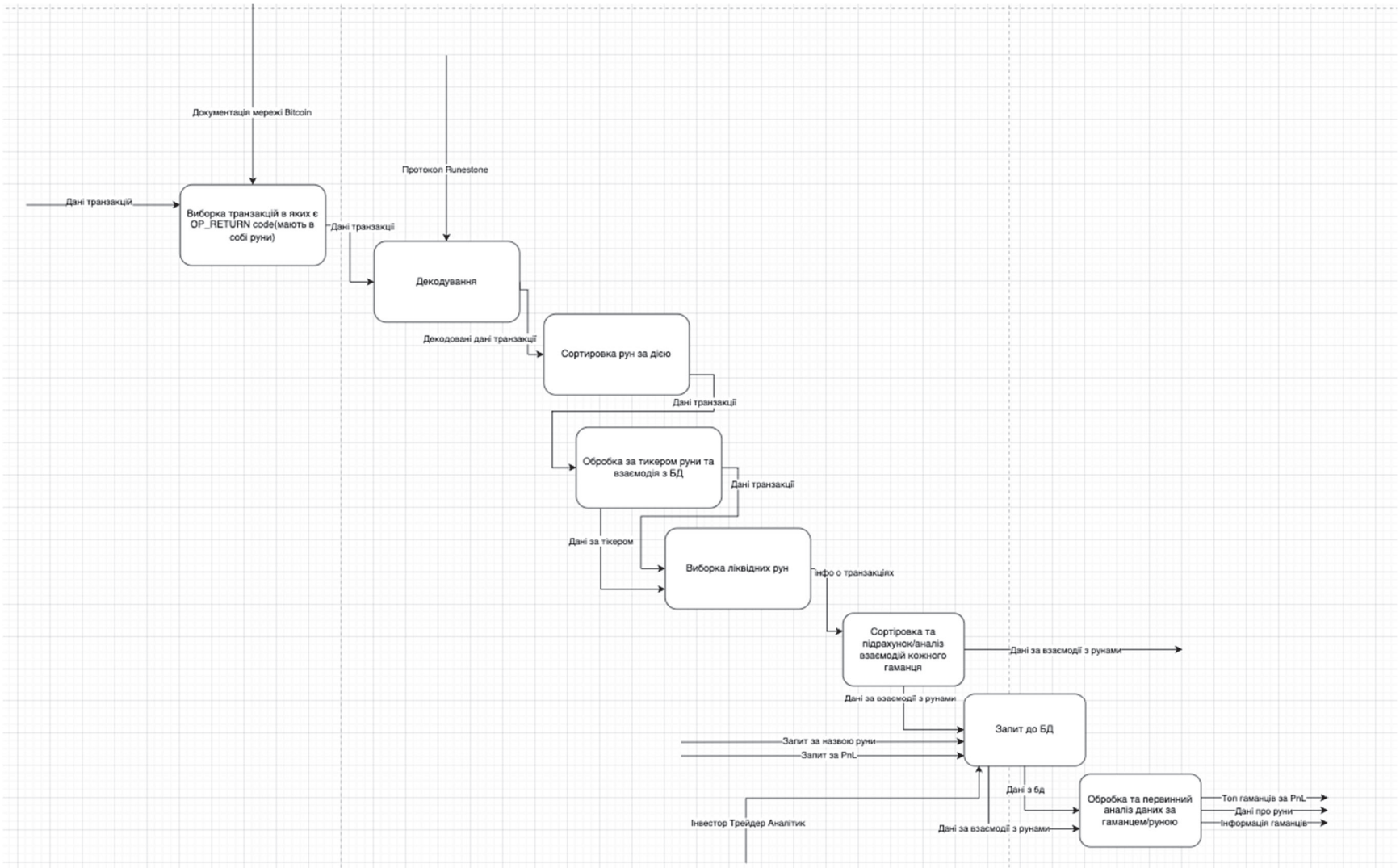

Рисунок 4.2 – Діаграма декомпозиції першого рівня модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin»

Таблиця 4.1 – Перелік та опис вихідних повідомлень модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin»

Назва	Форма представ-лення	Періодичність видачі	Допусти-мий час затримки	Отримувач інформації
Дані за взаємодії з рунами	Масив обє’ктів у TypeScript	Видача після обробки повного блока	До 1 хвилини	БД
Топ гаманців за PnL	Відеограма(форма) на сайті	За запитом	До 3-5 секунд	Інвестор, Трейдер, Аналітик
Дані про руни	Відеограма(форма) на сайті	За запитом	До 3-5 секунд	Інвестор, Трейдер, Аналітик
Інформація гаманців	Відеограма(форма) на сайті	За запитом	До 3-5 секунд	Інвестор, Трейдер, Аналітик

Таблиця 4.2 – Перелік та опис вхідних повідомлень модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin»

Назва	Форма представлення	Частота одержання	Джерело
Дані транзакцій	Масив закодованих строк	Після кожного включення блока у мережу Bitcoin. В середньому 10хв	Bitcoin RPC вузол
Запит за назвою руни	Відеограма	Після запиту	Інвестор, Трейдер, Аналітик
Запит за PnL	Відеограма	Після запиту	Інвестор, Трейдер, Аналітик

Таблиці 4.3 – 4.8 містять опис структурних одиниць вхідних та вихідних повідомлень.

Таблиця 4.3 – Структурні одиниці ввихідного повідомлення «Топ гаманців за PnL»

Повне найменування поля	Ідентифікатор	Тип інформації	Загальна кількість символів	Формат представлення	Точність обчислення і представлення
Ім’я гаманця	wallet_name	Символьний	66 (адреса Base58/Bech32)	string	до символів адреси
Загальний PNL	total	Числовий (рядок)	16	"0.000078 BTC"	до 8 десяткових знаків після коми
PNL за 30d	pnl_30d	Числовий (рядок)	16	"0.000078 BTC"	до 8 десяткових знаків після коми
PNL за 7d	pnl_7d	Числовий (рядок)	16	"0.000078 BTC"	до 8 десяткових знаків після коми
Загальна вартість	total_value	Числовий (рядок)	16	"$210.00"	дві цифри після десяткової крапки

Таблиця 4.4 – Структурні одиниці вихідного повідомлення «Дані про руни»

Повне найменування поля	Ідентифікатор	Тип інформації	Загальна кількість символів	Формат представлення	Точність обчислення і представлення
1	2	3	4	5	6
Назва руни	rune_name	Символьний	26	string	символи A–Z, «•»
Номер руни	rune_number	Цілий	5	number	ціле число
Символ	symbol	Символьний /Null	3	string або null	до 3 символів
Подільність	divisibility	Цілий	1	number	ціле число від 0 до 18
URI зображення	image_uri	URL/Null	128	string або null	URL
Ринкова капіталізація	price.market_cap	Символьний	20	string	до 2 десяткових знаків
Ціна (floor)	price.floor_unit_price_value	Символьний	20	string	до 8 десяткових знаків
Зміна ціни за 1d	price.delta_floor_1d	Символьний	6	string	відсоток без дробу
Зміна ціни за 7d	price.delta_floor_7d	Символьний	6	string	відсоток без дробу
Зміна ціни за 30d	price.delta_floor_30d	Символьний	6	string	відсоток без дробу
Обсяг 1h	volume.volume_1h	Символьний	16	string	до 8 десяткових знаків
Обсяг 1d	volume.volume_1d	Символьний	16	string	до 8 десяткових знаків

Кінець таблиці 4.4

1	2	3	4	5	6
Обсяг 7d	volume.volume_7d	Символьний	16	string	до 8 десяткових знаків
Обсяг 30d	volume.volume_30d	Символьний	16	string	до 8 десяткових знаків
Обсяг від початку	volume.volume_all	Символьний	16	string	до 8 десяткових знаків
Усього транзакцій	transactions.total_txns	Цілий	10	number	ціле число
Транзакцій за 1d	transactions.txn_count_1d	Цілий	10	number	ціле число
Транзакцій за 7d	transactions.txn_count_7d	Цілий	10	number	ціле число
Транзакцій за 30d	transactions.txn_count_30d	Цілий	10	number	ціле число
Кількість власників	holder_count.holder_count	Цілий	6	number	ціле число
Кількість «розумних» власників	holder_count.smart_holders_count	Цілий	6	number	ціле число
Кількість очікуваних ТХ	pending_count_tx	Цілий	4	number	ціле число

Таблиця 4.5 – Структурні одиниці частини(saveUtxo) вихідного повідомлення «Дані за взаємодіями з рунами»

Повне найменування поля	Ідентифікатор	Тип інформації	Загальна кількість символів	Формат представлення	Точність обчислення і представлення
Ідентифікатор руни	params.runeId	Символьний	12	string	"BLOCK:TX", лише цифри та ':'
Кількість	params.amount	Цілий (bigint)	20	bigint	в сатоші, до 8 десяткових знаків
Адреса	params.address	Символьний	66	string	Base58/Bech32
Статус	params.status	Символьний	10	string	"unspent"/"spent"
TX ID	params.created_txid або txid	Символьний	66	string	Base58/Bech32
Vout index	params.vout	Цілий	2	number	цілі числа
Block height	params.blockHeight	Цілий	6	number	цілі числа
Timestamp	params.timestamp	Цілий	10	number	секунди з епохи

Таблиця 4.6 – Структурні одиниці частини(saveRuneTransfer) вихідного повідомлення «Дані за взаємодіями з рунами»

Повне найменування поля	Ідентифікатор	Тип інформації	Загальна кількість символів	Формат представлення	Точність обчислення і представлення
TX ID	params.txid	Символьний	66	string	Base58/Bech32
Ідентифікатор руни	params.runeId	Символьний	12	string	"BLOCK:TX"
Кількість	params.totalAmount	Цілий (bigint)	20	bigint	в сатоші, до 8 десяткових знаків
Адреса відправника	params.fromAddress	Символьний/ Null	66	string	Base58/Bech32
Адреса отримувача	params.toAddress	Символьний/ Null	66	string	Base58/Bech32
Block height	params.blockHeight	Цілий	6	number	цілі числа
Timestamp	params.timestamp	Дата/час	10	number	секунди з епохи

Таблиця 4.7 – Структурні одиниці частини (saveRune) вихідного повідомлення «Дані за взаємодіями з рунами»

Повне найменування поля	Ідентифікатор	Тип інформації	Загальна кількість символів	Формат представлення	Точність обчислення і представлення
Ідентифікатор руни	params.runeId	Символьний	12	string	"BLOCK:TX"
Подільність	params.divisibility	Цілий	2	number	ціле число від 0 до 18
Назва руни	params.runeName	Символьний	26	string	символи A–Z, «•»
Символ	params.symbol	Символьний/Null	3	string or null	до 3 символів
Cap	params.terms?.cap	Цілий (bigint)/Null	20	bigint or null	в сатоші
Кількість за mint	params.terms?.amount	Цілий (bigint)/Null	20	bigint or null	в сатош

Таблиця 4.8 – Структурні одиниці вхідного повідомлення «Дані транзакцій»

Повне найменування поля	Ідентифікатор	Тип інформації	Загальна кількість символів	Формат представлення	Точність обчислення і представлення
Хеш блоку	hash	Символьний	64	hex string	64 hex-символи
Підтвердження	confirmations	Цілий	5	number	цілі числа
Номер блоку	height	Цілий	6	number	цілі числа
Час UNIX	time	Цілий	10	timestamp	секунди з епохи
Кількість транзакцій	nTx	Цілий	4	number	цілі числа
Список транзакцій	tx	Масив об’єктів	залежить від nTx	array of objects	кожен об’єкт має поля txid, vout, vin тощо
Розмір (bytes)	size	Цілий	6	number	цілі числа
Вага (weight)	weight	Цілий	7	number	цілі числа
Hex попереднього блоку	previousblockhash	Символьний	64	hex string	64 hex-символи
Hex наступного блоку	nextblockhash	Символьний /null	64	hex string or null	64 hex-символи

5 РОЗРОБКА ЕЛЕМЕНТІВ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ МОДУЛЯ “МОНІТОРИНГ ТА АНАЛІЗ ВЗАЄМОДІЙ З РУНАМИ” У МЕРЕЖІ «BITCOIN»

5.1 Обґрунтування вибору моделі даних та системи управління базами даних

Проведений аналіз моделей даних показав, що для розробки інформаційних систем, які працюють з великими обсягами структурованих даних та потребують складних аналітичних запитів, найкраще підходить реляційна модель даних. Реляційна модель організовує дані у вигляді таблиць, які складаються з рядків (записів) та стовпців (атрибутів), що забезпечує чіткість, структурованість та простоту роботи з інформацією. Найпоширенішими системами управління базами даних (СУБД) на основі реляційної моделі є PostgreSQL, MySQL, Oracle Database та Microsoft SQL Server [15].

Для розробки модуля «Моніторинг та аналіз взаємодій з рунами» обрано реляційну модель даних завдяки її наступним перевагам:

– чітка структурованість даних, що дозволяє легко виконувати аналітичні запити;

– підтримка складних SQL-запитів, необхідних для моніторингу та аналітики;

– можливість нормалізації даних для уникнення їх дублювання та забезпечення ефективного зберігання;

– вбудовані механізми цілісності даних (унікальність, зовнішні ключі, обмеження значень).

З-поміж наявних СУБД було обрано систему PostgreSQL – об'єктно-реляційну систему управління базами даних з відкритим вихідним кодом. Вибір саме PostgreSQL зумовлений такими її перевагами [16]:

– відкритий вихідний код і безкоштовність використання, що дозволяє значно знизити витрати на впровадження та підтримку;

– висока продуктивність та ефективна робота з великими обсягами даних, що особливо актуально для роботи з транзакціями блокчейну;

– підтримка широкого спектру розширень і модулів, таких як JSON-обробка, текстовий пошук, просторові та аналітичні функції, що підвищує гнучкість у роботі;

– багатоплатформенність, що дозволяє використовувати її в різних операційних системах (Linux, Windows, MacOS);

– розвинені механізми реплікації та забезпечення надійності, що гарантують високий рівень доступності та захищеності даних.

Таким чином, вибір реляційної моделі та СУБД PostgreSQL є оптимальним для реалізації завдань модуля моніторингу та аналізу rune-транзакцій у мережі Bitcoin, оскільки відповідає всім необхідним вимогам щодо обробки, зберігання та аналітики структурованих даних.

5.2 Опис сутностей та зв’язків модуля

Після проведеного аналізу предметної області в рамках модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin» було виділено наступні сутності, атрибути, що їх характеризують, та зв’язки між сутностями.

Оскільки генерування вихідних форм, пов’язаних з первинним аналізом, відбувається кожен раз при запиті на сайті, було прийняте рішення не зберігати дані в БД, а кожний раз виконувати генерацію необхідного аналізу або виконувати сортування із вказаним проміжком часу, який буде оновлювати на сервері ці дані. Це забезпечить економію ресурсів БД, окрім того, зробить процес формування аналізу більш гнучким, адже за потреби

можна буде змінити вивід або зміст, формат і кількість даних.

При формуванні БД було виділено 6 сутностей, кожна з яких має своє унікальне ім’я: marketplace (Біржа), rune_transfers_spent (рун транзакції витрачені), rune_transfers_created (рун транзакції створені), utxo (частина транзакції, в якій зберігається інформація), runes (Руни), mints_events (Створення рун).

Під час формування структури БД використовується наступна класифікація зв’язків: один-до-багатьох (1:M). У таблиці 5.1 наведені зв’язки між сутностями БД.

Таблиця 5.1 – Відомості про типи зв’язків

Тип сутності	Тип сутності	Тип зв’язку	Кардіналність
1	2	3	4
utxo	runes	Has (має)	M:1
utxo	mints_events	Part of (частина)	M:1
utxo	rune_transfers_spent	Part of (частина)	M:1
utxo	rune_transfers_created	Part of (частина)	M:1
runes	utxo	Part of (частина)	1:M
mints_events	utxo	Part of (частина)	1:M
rune_transfers_spent	utxo	Has (має)	1:M
rune_transfers_spent	marketplace	Part of (частина)	M:1
rune_transfers_created	utxo	Has (має)	1:M
rune_transfers_created	marketplace	Part of (частина)	M:1
marketplace	rune_transfers_created	Has (має)	1:M
marketplace	rune_transfers_spent	Has (має)	1:M

5.3 Опис атрибутів сутностей та їх доменів

Кожна сутність має свої атрибути та домени. В таблиці 5.2 представлені домени атрибутів, а атрибути сутностей – в таблиці 5.3.

Таблиця 5.2 – Відомості про домени атрибутів

Ім’я домену	Характеристика домену	Приклади припустимих значень
divisibility, amount_per_mint, mint_id, amount_of_mints, deadline та інші	Рядок змінної довжини, числовий тип даних	232015, 2, 17, 12948, 174000231
rune_id, rune_name, symbol та інші	Рядок змінної довжини, символьний тип даних, до 20-70 символів	801456:1945, BILLION•DOLLAR•CAT, іконка
total_sale_price, total_rune_amount, amount	64-розрядний цілочисельний тип даних із знаком.	1000000n
volume_24h	Рядок змінної довжини, числовий тип даних. Дробові числа	5132480,55234
block_height, timestamp та інші	Рядок змінної довжини, числовий тип даних. Цілі числа	810351, 174000547

Таблиця 5.3 – Відомості про атрибути сутностей

Тип сутності	Атрибут	Тип даних, кількість символів	Опис	Обмеження	Припустимість Null
1	2	3	4	5	6
utxos	utxo_id_key	Символьний (70)	Унікальний ідентифікатор UTXO	PK (первинний ключ)	Ні
	rune_id	Символьний (20)	Ідентифікатор руни	FK(вторинний ключ)	
	amount	Ціле число (довге)	Кількість рун в UTXO		
	address	Символьний (64)	Адреса отримувача		
	status	Символьний (20)	Статус UTXO (наприклад, active/spent)		
	created_txid	Символьний (64)	TXID, який створив UTXO	FK(вторинний ключ)	
	vout	Ціле число	Індекс виходу транзакції		
	spent_txid	Символьний (64)	TXID, який витратив UTXO	FK(вторинний ключ)	Так
	mint_id	Ціле число	Ідентифікатор події створення руни	FK(вторинний ключ)	
runes	rune_id	Символьний (20)	Унікальний ідентифікатор руни	PK	
	divisibility	Ціле число	Кількість знаків після коми		
	rune_name	Символьний (100)	Назва руни		
	symbol	Символьний (20)	Символ руни		
	cap	Символьний (40)	Максимальна емісія		
	amount_per_mint	Ціле число	Кількість рун за 1 мінт		
	rune_ticker	Символьний (100)	Тікер руни		
	premine	Символьний (40)	Кількість на премайн		

Кінець таблиці 5.3

1	2	3	4	5	6
rune_ transfers_ created	txid	Символьний (64)	Ідентифікатор створюючої транзакції	PK	
	from_address	Символьний (62)	Адреса відправника		
	to_address	Символьний (62)	Адреса отримувача		
	total_rune_amount	Ціле число (довге)	Загальна кількість рун у трансфері		
	block_ height	Ціле число	Висота блоку		
	timestamp	Ціле число	Час створення		
	marketplace_id	Ціле число	Ідентифікатор маркетплейсу	FK	
	total_sale_ price	Ціле число (довге)	Загальна ціна продажу		
rune_ transfers_ spent	txid	Символьний (64)	Ідентифікатор витрачаючої транзакції	PK	
	from_ address	Символьний (62)	Адреса відправника		
	to_address	Символьний (62)	Адреса отримувача		
	total_rune_ amount	Ціле число (довге)	Загальна кількість рун у трансфері		
	block_ height	Ціле число	Висота блоку		
	timestamp	Ціле число	Час витрати		
	marketplace_id	Ціле число	Ідентифікатор маркетплейсу	FK	
	total_sale_ price	Ціле число (довге)	Загальна ціна продажу		
marketplace	marketplace_id	Ціле число	Ідентифікатор маркетплейсу	PK	
	volume_24h	Дробове число(20,8)	Обсяг торгів за 24 години		
	marketplace_name	Символьний (50)	Назва маркетплейсу		
mints_ events	mint_id	Ціле число	Ідентифікатор події минта	PK	
	amount_of_ mints	Ціле число	Кількість можливих мінтів		
	deadline	Ціле число	Кінцева дата минта (unix час)		

5.4 Розробка логічної та фізичної схеми бази даних модуля

Наступним етапом після виділення основних сутностей, доменів та атрибутів є побудова логічної моделі БД модуля. Ця модель є основою, яка описує БД на абстрактному рівні. Вона зображена на рисунку 5.1.

На рисунку 5.2 зображена фізична модель БД, яка побудована на основі логічної схеми і визначає дані, що будуть зберігатися фізично у зовнішній пам’яті комп’ютера, з урахуванням особливостей обраної СУБД.

Для розробки схем модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» був використаний безкоштовний web-платформа app.diagrams, яка дозволяє будувати діаграми, блок-схеми, графіки та інше.

На схемі фізичної моделі БД для атрибутів сутностей були використані наступні типи даних: Varchar, Numeric різної довжини, Integer та BigInt.

Відповідно до таблиці 5.3, на якій наведені відомості про атрибути сутностей, на фізичній моделі даних деякі атрибути мають значення Not Null. Оскільки значення Null опускається і вказується лише обмеження на недопустимість нуля, всі колонки, що не мають явно визначеної специфікації, за замовчуванням, набувають значення Null.

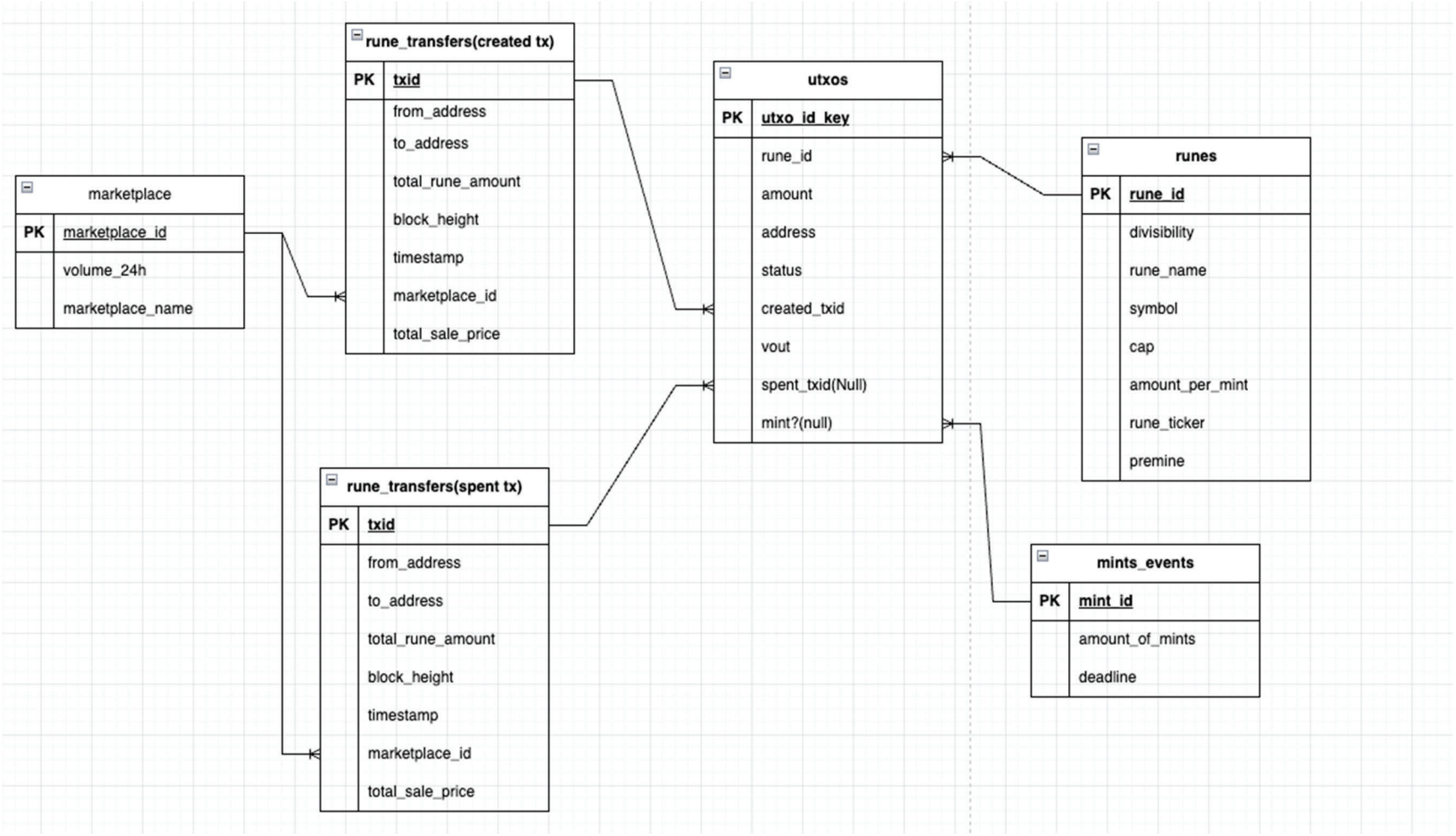

Рисунок 5.1 – Схема логічної моделі бази даних модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin»

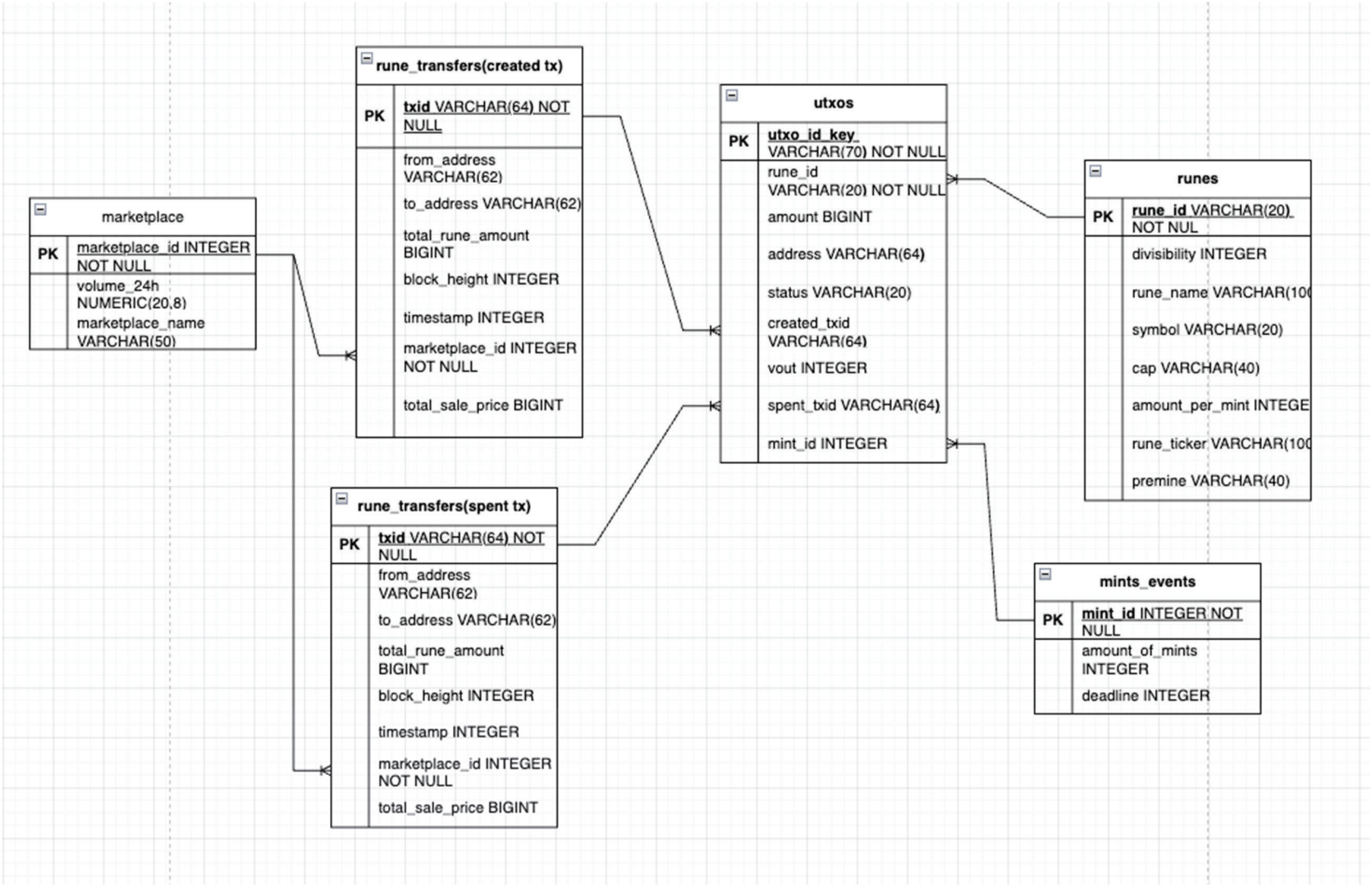

Рисунок 5.2 – Схема фізичної моделі бази даних модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin»

6 РОЗРОБКА ЕЛЕМЕНТІВ МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ «МОНІТОРИНГ ТА АНАЛІЗ ВЗАЄМОДІЙ З РУНАМИ» У МЕРЕЖІ BITCOIN

6.1 Аналіз часу виконання декодування

Проведення аналізу часу виконання процесів декодування та запису даних у модулі «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin» дозволяє оцінити ефективність роботи модуля та його здатність оперативно надавати актуальну інформацію користувачам.

Для аналізу часу декодування транзакцій та запису даних використовуються такі ключові параметри:

– час початку процесу декодування транзакцій;

– час завершення декодування та запису даних у базу даних;

– кількість транзакцій, що обробляються за одиницю часу;

– загальний час виконання декодування та запису даних.

Процес аналізу виконання здійснюється емпірично. Спершу ми запускаємо тест декодування найближчої за функціоналом технології runestone-lib, у якому перевіряємо час, за який здійснюється аналіз одного блока. Наприклад, використано блок з номером 811001. Для цього був створений файл indexTest.ts, в якому було викликано основний клас для роботи з модулем декодування. На рисунку 6.1 представлено фрагмент коду з цього файлу.

```
357
358   await indexer.start();
359
360   console.log('\n=== Single Block Benchmark Example ===');
361   try {
362     const blockToBenchmark = 811001;
363     const benchmarkResult = await indexer.benchmarkSingleBlock(blockToBenchmark);
364     console.log('\nBenchmark completed successfully!');
365     console.log(
366       `Block ${benchmarkResult.blockHeight} processed in ${benchmarkResult.totalTime}ms`
367     );
368   } catch (error) {
369     console.log('Benchmark failed:', error);
370   }
371
```

Рисунок 6.1 – Скріншот коду з файлу indexTest.ts

На риснку 6.2 можна побачити логування процесу декодування у файлі updater.ts.

```
const runeProcessingStartTime = Date.now();
const runeUpdater = new RuneUpdater(this._network, block, false, this._storage, this._rpc);
perfTracker.mark('rune_updater_created');

for (const [txIndex, tx] of block.tx.entries()) {
  await runeUpdater.indexRunes(tx, txIndex);
}

const runeProcessingTime = Date.now() - runeProcessingStartTime;
perfTracker.mark('rune_processing_complete');

await this._storage.saveBlockIndex(runeUpdater);
perfTracker.mark('storage_save_complete');

const totalTime = perfTracker.getElapsed();
const metrics = perfTracker.getMetrics();

const benchmark = {
  blockHeight,
  totalTime,
  runeProcessingTime,
  blockFetchTime: metrics['block_fetch'],
  storageTime: metrics['storage_save_complete'] - metrics['rune_processing_complete'],
  txCount: block.tx.length,
  blockSize: block.size,
  avgTimePerTx: runeProcessingTime / block.tx.length,
  processingSpeedKBs: block.size / 1024 / (totalTime / 1000),
  throughputTxs: block.tx.length / (totalTime / 1000),
};

console.log(`  - Total time: ${benchmark.totalTime}ms`);
console.log(`  - Storage time: 442ms`);
console.log(`  - Avg time per tx: ${benchmark.avgTimePerTx.toFixed(2)}ms`);
console.log(`  - Processing speed: ${benchmark.processingSpeedKBs.toFixed(2)} KB/s`);
console.log(`  - Throughput: ${benchmark.throughputTxs.toFixed(2)} tx/s`);
```

Рисунок 6.2 – Скріншот коду з файлу updater.ts

Після проведення тестування, результати якого зображені на рисунках 6.3-6.4, можна зробити висновок, що після модернізації та впровадження нашого модуля ми досягли таких результатів порівняно з найближчою технологією runestone-lib:

– середній час виконання декодування однієї транзакції зменшився;

– реалізована додаткова функція збереження даних до бази даних;

– типізовані дані після декодування, що покращило надійність та читабельність коду;

– покращений алгоритм декодування шляхом усунення зайвих функцій та оптимізації циклів з обробки транзакцій.

Хоча загальний час виконання не зменшився через додаткову операцію запису до бази даних, ефективність процесу декодування значно зросла за рахунок оптимізації алгоритму. Попередній алгоритм мав складність O(n), де n – кількість транзакцій. Новий підхід використовує асинхронну обробку з використанням Workers Threads, що дозволило виконувати операції паралельно, зменшуючи загальний час обробки.

Покращення продуктивності алгоритму можна оцінити за допомогою наступних формул:

Загальний час обробки транзакцій:

$$T_{\text{заг}} = T_{\text{завершення}} - T_{\text{початку}}$$

Середній час обробки однієї транзакції:

$$T_{\text{сер}} = \frac{T_{\text{заг}}}{N}$$

де $T_{\text{завершення}}$– час завершення процесу декодування;

$T_{\text{початку}}$– час початку процесу декодування;

N – кількість транзакцій, які були оброблені.

Покращення продуктивності після оптимізації:

$$P_{\text{покращення}} = \left(\frac{T_{\text{сер старий}} - T_{\text{сер новий}}}{T_{\text{сер старий}}}\right) \times 100\%$$

де $T_{\text{сер старий}}$ – середній час обробки транзакцій до оптимізації;

$T_{\text{сер новий}}$ – середній час обробки транзакцій після оптимізації.

Використовуючи асинхронні Worker Threads, можна досягти паралельної обробки транзакцій, що дозволяє розподілити навантаження на процесорні ядра і суттєво скоротити час виконання задач. Таким чином, впровадження асинхронної обробки є перспективним напрямом для подальшого підвищення продуктивності системи.

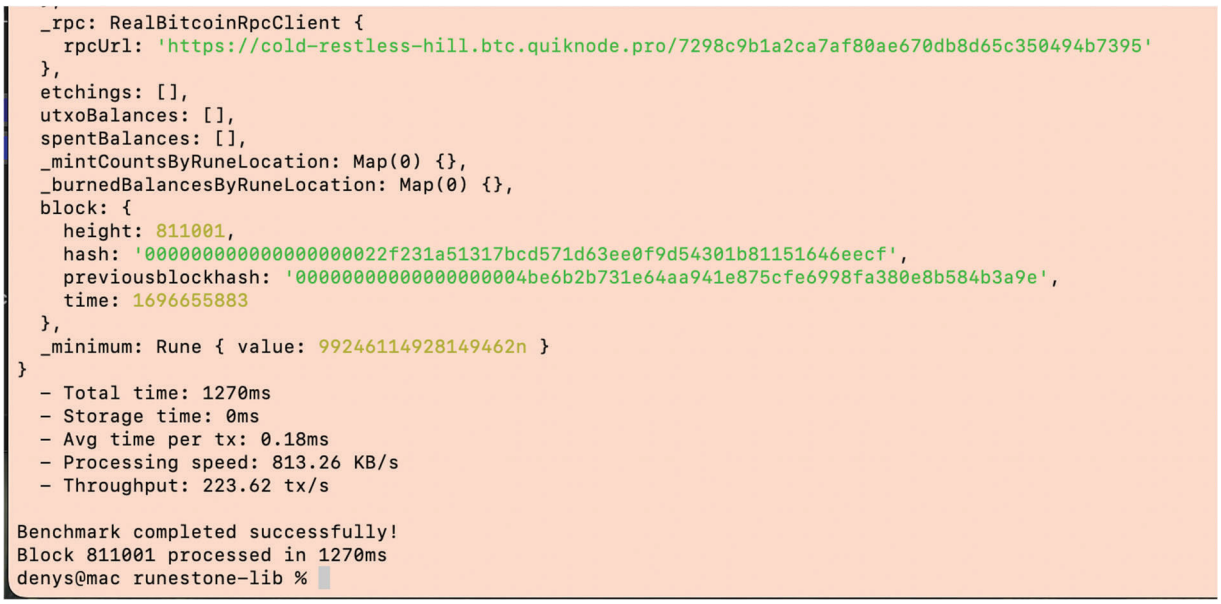

Рисунок 6.3 – Результат проведення першого іспиту

Рисунок 6.4 – Результат проведення другого іспиту

6.2 Аналіз частоти виконання декодування та середня кількість

Для аналізу частоти добування блоків вводиться індикаторна функція, яка набуває значення 1, якщо певний блок був знайдений конкретним майнером, і 0 в іншому випадку. Підсумовуючи значення цієї функції по всіх

здобутих блоках, отримуємо число блоків, знайдених майнером. Відносна частота його успіхів обчислюється за формулою [4]:

$$f_m = \frac{N_m}{B},$$

де f_m – індикаторна функція відносної частоти успішно здобутих блоків;

N_m – число блоків, знайдених майнером;

B – всі здобуті блоки.

Ця частота апроксимується як відношення власного хешрейту (H_m) до загального хешрейту мережі [4].

$$f_m = \frac{H_m}{H_{total}},$$

де H_m – власний хешрейт;

H_{total} – загальний хешрейт мережі.

n_i – число рун-транзакцій у і-му блоці. Ця величина показує, наскільки насиченими контентом «рунів» є блоки в середньому.

Середня кількість рун-транзакцій на один блок:

$$n_{avg} = \frac{1}{B}\sum_{i=1}^{B} n_i,$$

де n_i – кількість рун-транзакцій у і-му блоці;

B – кількість блоків.

Оцінка часу добування блоку:

$$N_{hashes} = D \times 2^{32},$$

де D – складність мережі;

2^{32} – фіксований множник, що відповідає середній кількості хешів для одного раунду.

$$T_{block} = \frac{D \times 2^{32}}{H},$$

де D – складність мережі;

H – хешрейт майнера або мережі.

Цільовий інтервал між блоками [4]:

$$T_{target} \approx 600 \text{ секунд}$$

Загальний хешрейт мережі [4]:

$$H_{total} = \frac{D \times 2^{32}}{T_{target}},$$

де D – складність мережі;

T_{target} – цільовий час між блоками.

Час між успішними блоками для майнера:

$$T_m = \frac{T_{target}}{f_m},$$

де T_{target} – цільовий час між блоками;

f_m – частка успішно знайдених блоків цим майнером.

6.3 Розробка алгоритму роботи модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin»

Розроблюваний модуль «Моніторинг та аналіз взаємодій з рунами» у мережі Bitcoin виконує функції моніторингу, декодування та аналізу даних щодо транзакцій з рунами.

Користувачами модуля є трейдери, аналітики та інвестори, які здійснюють запити для отримання інформації про взаємодії з рунами.

Загальний алгоритм роботи модуля полягає в наступному:

а) користувач (трейдер або аналітик) через web-додаток здійснює запит на отримання інформації щодо певної руни або гаманця. Web-додаток передає цей запит через API серверній частині;

б) серверна частина постійно виконує фонові процеси взаємодії з Bitcoin RPC, регулярно перевіряючи оновлення у блокчейні:

– отримує номер останнього блоку від Bitcoin RPC;

– запитує транзакції з останнього блоку;

– декодує отримані транзакції та перевіряє їх на наявність runestone-повідомлень;

– записує нові або оновлює існуючі дані про руни та транзакції в базу даних;

в) після отримання API-запиту серверна частина звертається до бази даних за актуальною інформацією відповідно до запиту, обробляє та структурує отримані дані і повертає їх web-додатку;

г) web-додаток у зручній формі представляє отриману інформацію користувачу, що дозволяє ефективно здійснювати моніторинг та аналіз взаємодій з рунами у мережі Bitcoin.

Таким чином, модуль забезпечує безперервну роботу фонових процесів, що гарантує актуальність, точність і оперативність інформації для кінцевих користувачів.

Ілюстрація послідовності операцій представлена на рисунку 6.5 у вигляді діаграми UML.

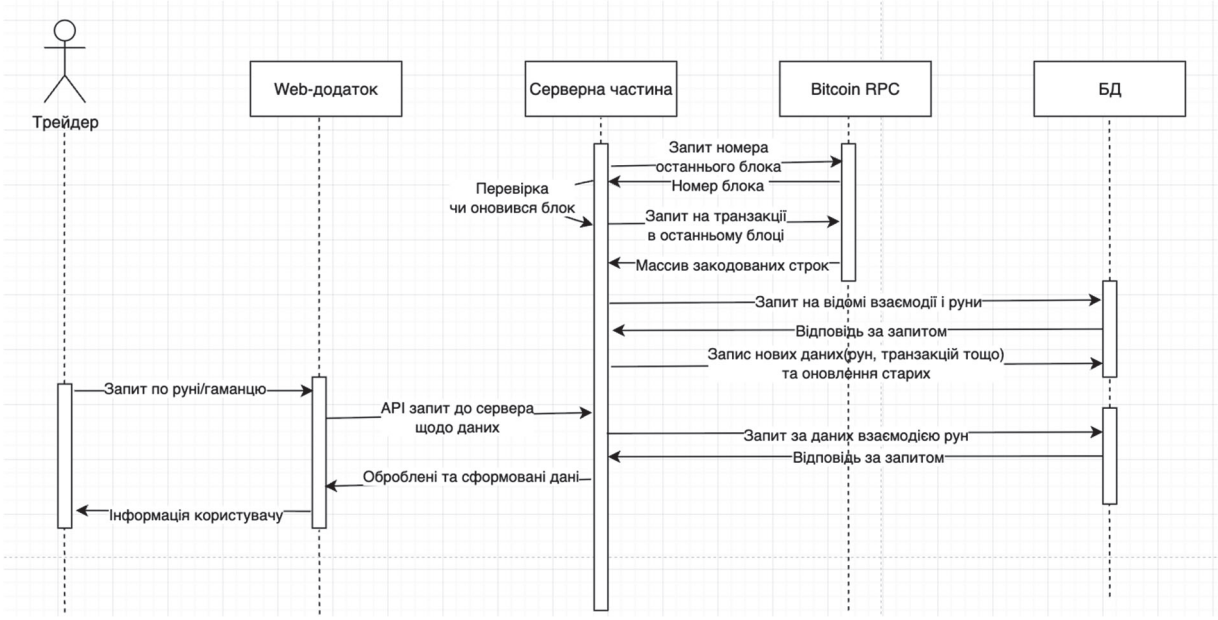

Рисунок 6.5 – Діаграма UML до роботи модуля «Моніторинг та аналіз взаємодій з рунами» у мережі Bitcoin

7 РОЗРОБКА І ОБҐРУНТУВАННЯ ВИБОРУ ЕЛЕМЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

7.1 Обґрунтування вибору елементів програмного забезпечення модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin»

Під час проєктування модуля, розробки та опису бізнес-процесів та підпроцесів використовувався програмний засіб app.diagrams [17] – онлайн додаток, який є безкоштовним та дає можливість створювати різноманітні діаграми.

За допомогою app.diagrams було розроблено схеми організаційної структури, IDEF0 діаграми, логічна та фізична схеми БД, також UML діаграми та схема архітектури технічного забезпечення модуля.

Для розробки прикладного додатку модуля використовується комбінація засобів, що включає в себе бібліотекі Next.js, React , а також використовує axios для HTTP-запитів [18], [19], [20]. Next.js забезпечує серверний рендеринг, автоматичний роутинг і можливість статичної генерації сторінок, що дає змогу створити швидкий і SEO-дружній інтерфейс. React у поєднанні з TypeScript відповідає за компонентну архітектуру – чітка типізація пропсів і стейтів дозволяє уникати помилок під час розробки та спрощує підтримку коду. Axios виступає клієнтом для Representational State Transfer (REST) API – він відправляє запити до бекенда, обробляє відповіді і передає отримані дані у React-компоненти, забезпечуючи динамічний обмін інформацією між браузером і сервером [30].

Серверна частина застосунку побудована на TypeScript у рамках мікросервісної архітектури з використанням NestJS та бібліотеки runestone-lib [21]. Замість монолітної структури кожен функціональний блок реалізовано як окремий сервіс, що дозволяє розподілити навантаження й полегшити масштабування. Один із таких сервісів відповідає за декодування

даних і запис у базу – він отримує сирі OP_RETURN-транзакції, передає їх на обробку runestone-lib для парсингу та валідації рун, а потім за допомогою TypeORM зберігає результати (інформацію про гаманці, транзакції та PnL) у PostgreSQL [22]. Окремий сервіс забезпечує роботу REST-API – у контролерах NestJS описано кінцеві точки, які приймають запити від фронтенду та повертають клієнтові вже підготовлені дані. Бізнес-логіку формування відповідей перенесено до окремих сервісних класів, які звертаються до сервісу декодування через внутрішні виклики, тому API-сервер виконує лише фасування та віддачу JSON-файлів без безпосереднього опрацювання «важкої» логіки.

Щоб виконувати ресурсоємні та тривалі завдання, такі як періодичне оновлення PnL-розрахунків або генерація звітів, передбачено окремий Jobs-Service, що працює з чергою повідомлень на базі Redis [23]. Коли настає час для оновлення даних, цей сервіс запускає задачу, звертається до модуля декодування, проводить необхідні оновлює відповідні записи в базі. Після завершення фонова служба надсилає REST-API-серверу повідомлення про готовність нових даних або просто залишає оновлені значення в PostgreSQL, до яких API-сервер звертається під час наступного запиту.

Для реалізації модулю було обрано клієнт-серверну архітектуру, оскільки вона чітко розділяє обов’язки між інтерфейсом користувача і бізнес-логікою, зменшує вимоги до робочих станцій користувачів та забезпечує центральну точку зберігання й обробки даних. Фронтенд-застосунок, побудований із використанням Next.js, React і TypeScript, запускається у веб-браузері, що свідчить про мінімальні системні вимоги на стороні клієнта. Незалежно від того, чи працює браузер на Windows, macOS чи Linux, користувач завжди отримує уніфіковану та відлагоджену поведінку інтерфейсу. Використання Next.js із серверним рендерингом і автоматичною генерацією статичних сторінок дозволяє значно підвищити продуктивність і SEO-індексацію застосунку, а TypeScript у поєднанні з React гарантує сувору типізацію компонентів та станів, що зменшує ризик помилок на етапі

розробки й полегшує подальшу підтримку коду. Axios, який використовується для організації HTTP-запитів, забезпечує зручний механізм обміну даними з бекендом і коректну обробку відповідей та помилок, що є критично важливим для стабільної взаємодії між браузером і сервером.

Серверну частину було реалізовано у середовищі Linux із застосуванням NestJS і бібліотеки runestone-lib на TypeScript. Вибір Linux обґрунтований його високою надійністю, безпекою та широкою підтримкою серед DevOps-спеціалістів, а також можливістю запуску контейнеризованих сервісів у Docker-оточенні.

Для зберігання структурованих даних обрана СУБД PostgreSQL, оскільки вона характеризується стабільністю, потужним механізмом транзакцій і здатністю обробляти великі обсяги записів. Взаємодію з PostgreSQL організовано через TypeORM, який дозволяє описувати схему таблиць у вигляді Entity-класів TypeScript та керувати її еволюцією за допомогою міграцій [23]. Завдяки цьому оновлення структури бази даних проходить автоматично – достатньо створити новий файл міграції і команда TypeORM застосує необхідні зміни, що спрощує підтримку та розвиток модулю з плином часу.

Крім основних сервісів декодування та REST-API, у модуль інтегровано Jobs-Service, що працює з чергою повідомлень на базі Redis, для виконання ресурсоємних і тривалих завдань (наприклад, періодичного оновлення PnL-розрахунків). Використання Redis у ролі кешу дозволяє зберігати результати часто запитуваних запитів, зменшуючи навантаження на базу даних і пришвидшуючи відгук REST-API. Сам Jobs-Service організовує фонові обчислення в окремому процесі, не блокуючи основний потік обробки вхідних HTTP-запитів. Після завершення роботи Jobs-Service відповідні кешовані дані оновлюються, або REST-API отримує повідомлення про доступність актуальних значень, що підвищує загальну швидкодію та відмовостійкість системи.

Таким чином, запропонована сукупність технологій Next.js, React, TypeScript, Axios на клієнтській стороні та NestJS, runestone-lib, TypeORM, PostgreSQL, Redis на сервері, відповідає всім вимогам до продуктивності, масштабованості та простоти підтримки. Завдяки клієнт-серверній архітектурі та розподіленню відповідальності між мікросервісами забезпечується можливість незалежного розвитку кожного компонента, а також гнучка адаптація під зростання навантаження або зміну бізнес-логіки.

7.2 Опис прикладної програмної частини модуля

Інтерфейс користувача модуля «Моніторинг та аналіз взаємодій з рунами» реалізовано як веб-додаток. Основні екрани та їх функції:

Сторінка Runes – після входа користувач потрапляє на головну сторінку «Runes» (рисунок 7.1). Тут відображаються ключові метрики: Total Volume за 24h та 1h, Market Cap, кількість гаманців та активних за 24h. Нижче розміщено таблицю зі списком руни: її назва, поточна ціна, об’єми та кількість транзакцій. Кнопка «Buy» дозволяє перейти до процесу покупки токена (при інтеграції з торговою платформою).

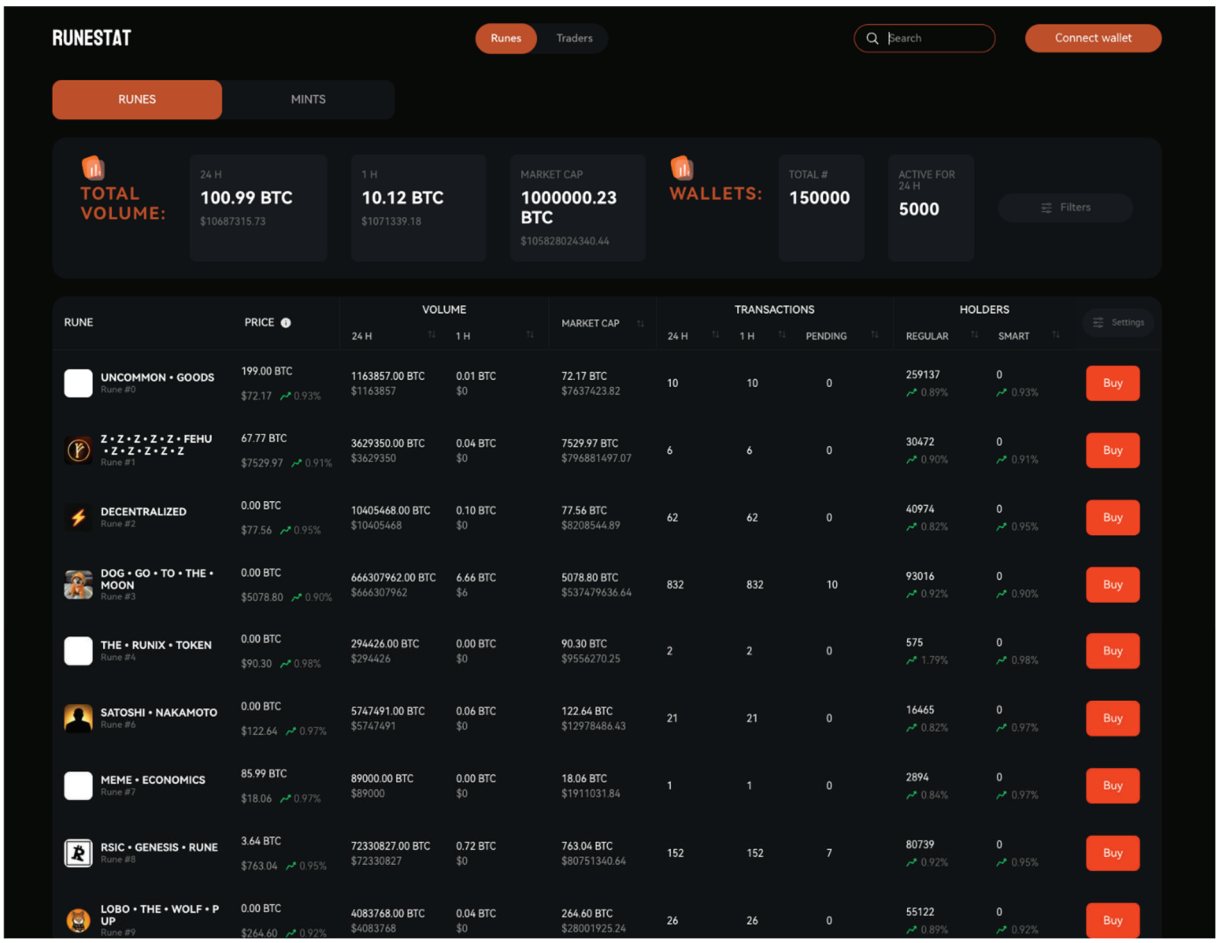

Рисунок 7.1 – Сторінка Runes

Перемикання в режим «Traders» відображає список гаманців (рисунок 7.2) з їхніми PNL та Total Value. Користувачі можуть сортувати за будь-яким стовпцем, а також застосовувати фільтри через ту ж панель.

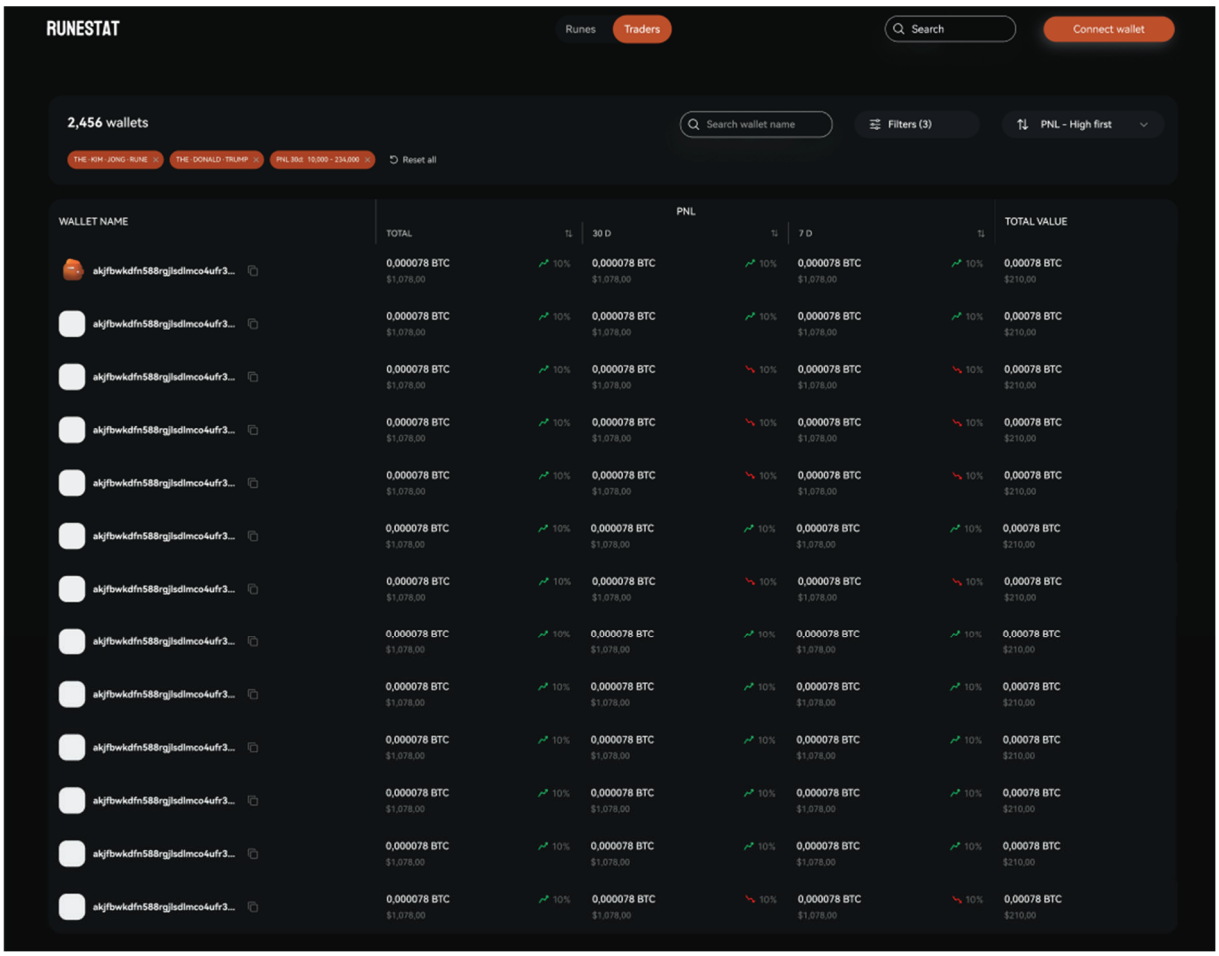

Рисунок 7.2 – Сторінка «Traders»

Деталі гаманця – при виборі окремого гаманця відкривається сторінка деталей (рисунок 7.3), де відображається поточне значення портфеля у BTC та USD, реалізований та нереалізований PNL, статистика торгів (token winrate, tokens traded) та таблиця зі списком токенів у портфелі: PNL по кожній руні, ціна, кількість.

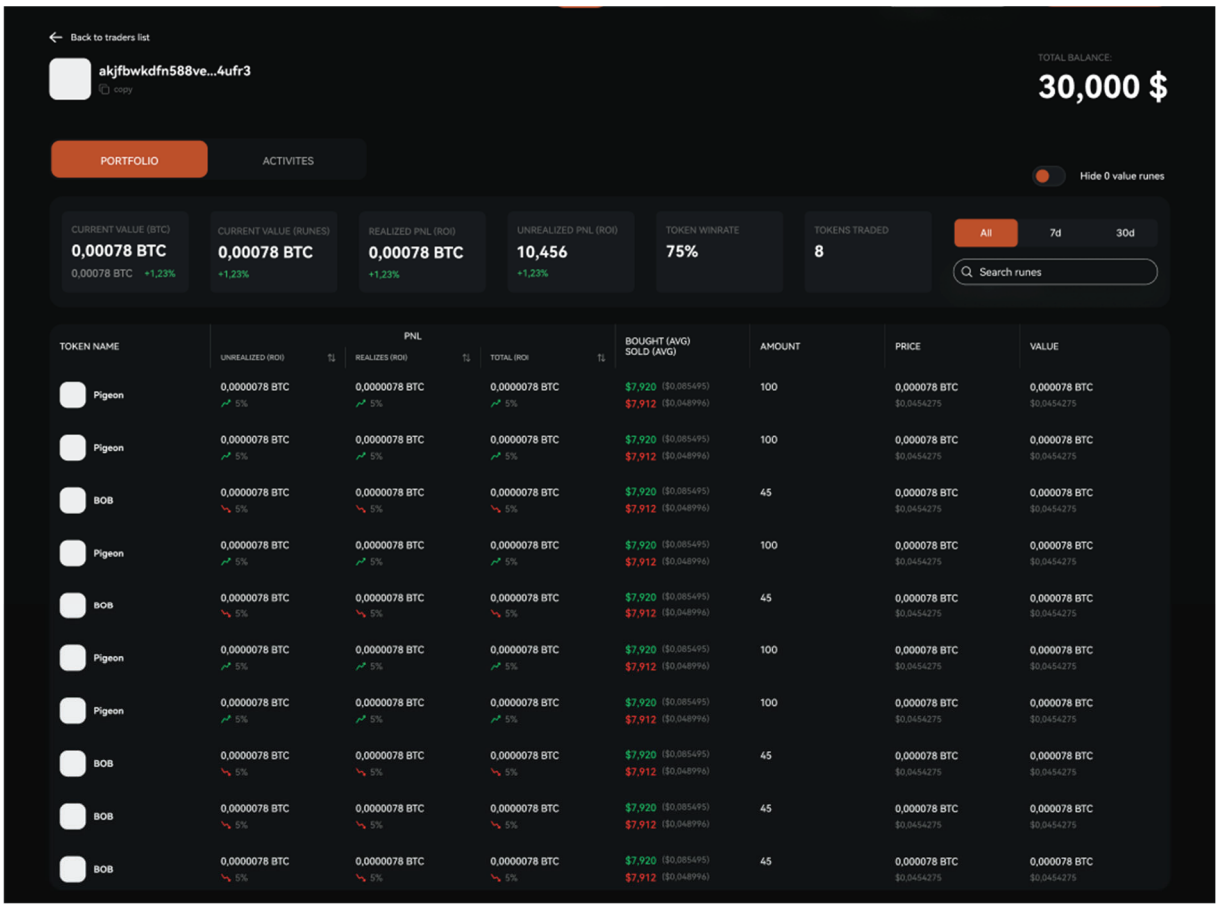

Рисунок 7.3 – Сторінка окремого гаманця

Пошук – вгорі інтерфейсу у правому куті присутнє поле пошуку (рисунки 7.4 – 7.5), яке дозволяє швидко знайти руну за назвою або гаманець за адресою. Пошуковий запит виконується при натисканні Enter або кліці на іконку лупи.

Рисунок 7.4 – Фрагмент сторінки Пошук

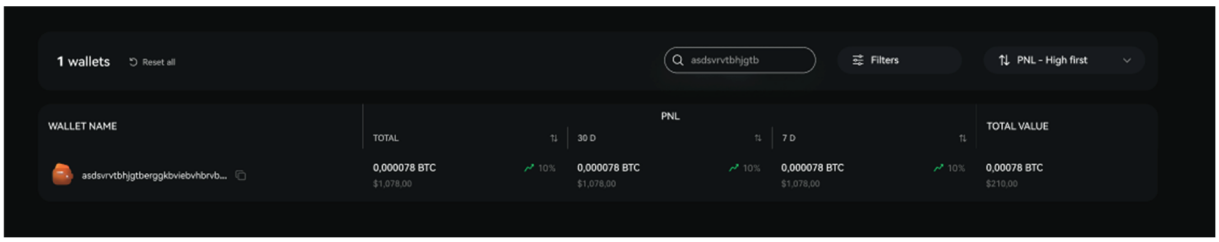

Рисунок 7.5 – Результат пошуку за номером гаманця

Фільтри – на екрані списків (Runes / Traders) користувач може натиснути кнопку «Filters» для відкриття панелі фільтрації. На рисунку 7.6 показано модальне вікно фільтрів, яке дозволяє задати діапазони PNL Total, PNL за 30 та 7 днів, Total Value. На рисунку 7.7-7.8 можна побачити інше вікно фільтру на сторінці «Runes», яке дозволяє задати діапазони за об’ємом, назвою, кількістю транзакцій і власників руни. Після натискання «Apply» застосовуються обрані фільтри, а «Reset all» повертає початкові налаштування.

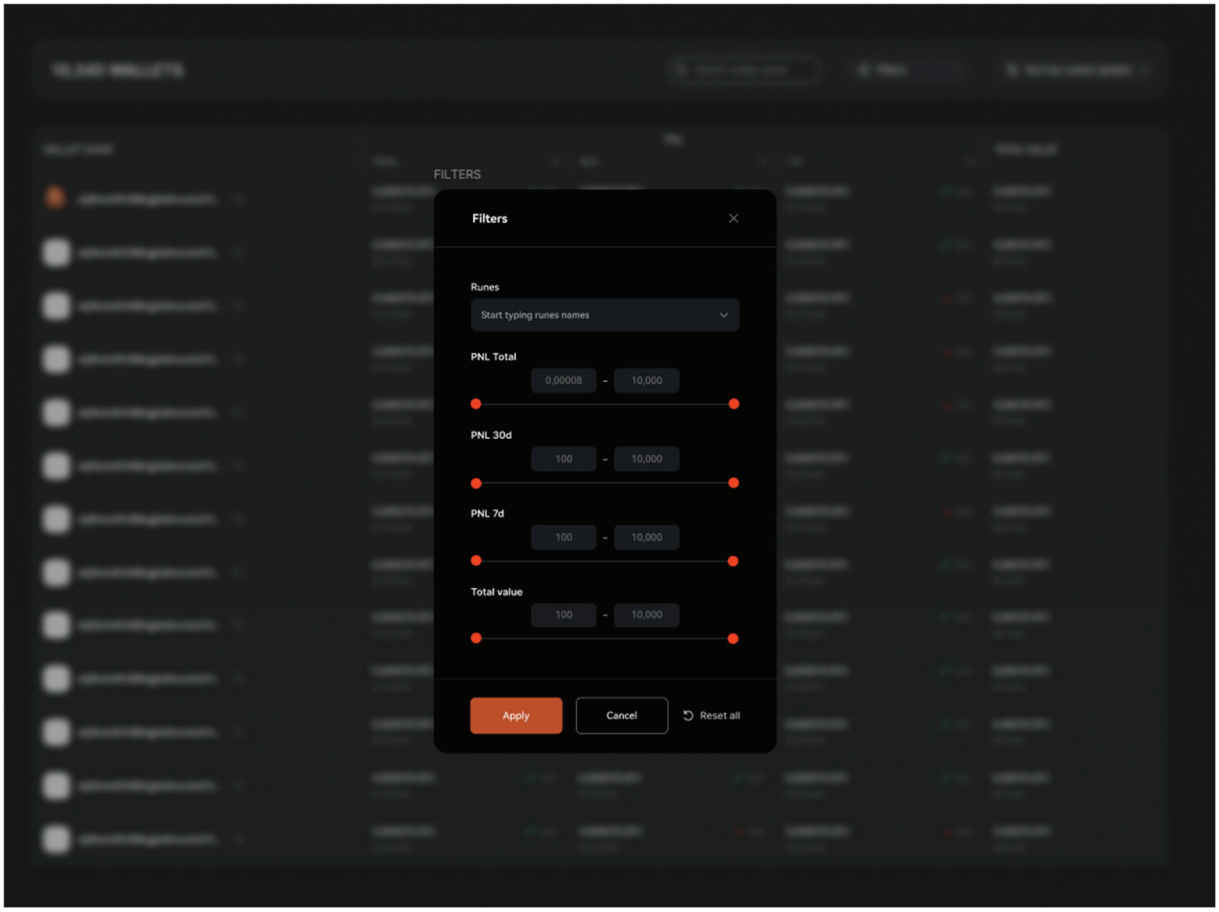

Рисунок 7.6 – Модальне вікно фільтру на сторінці «Traders»

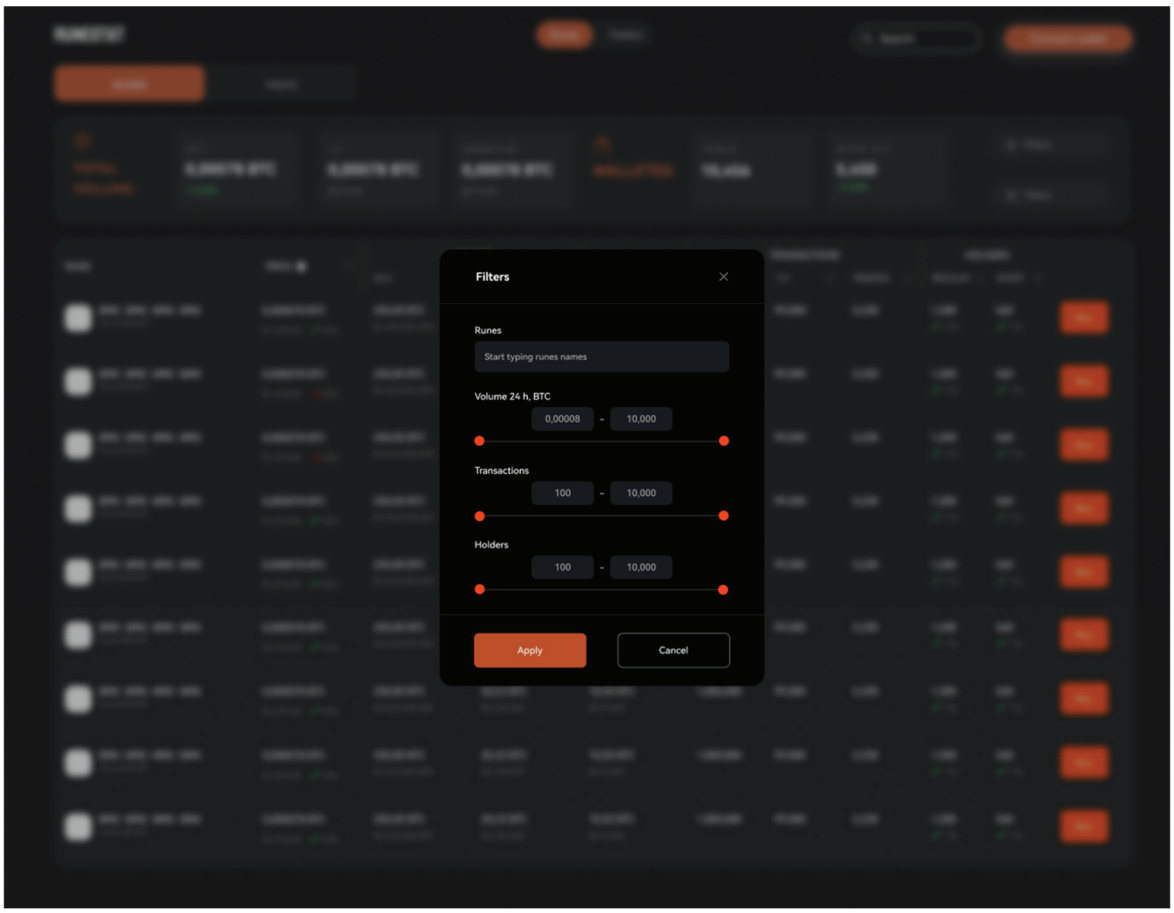

Рисунок 7.7 – Модальне вікно фільтру на сторінці «Runes»

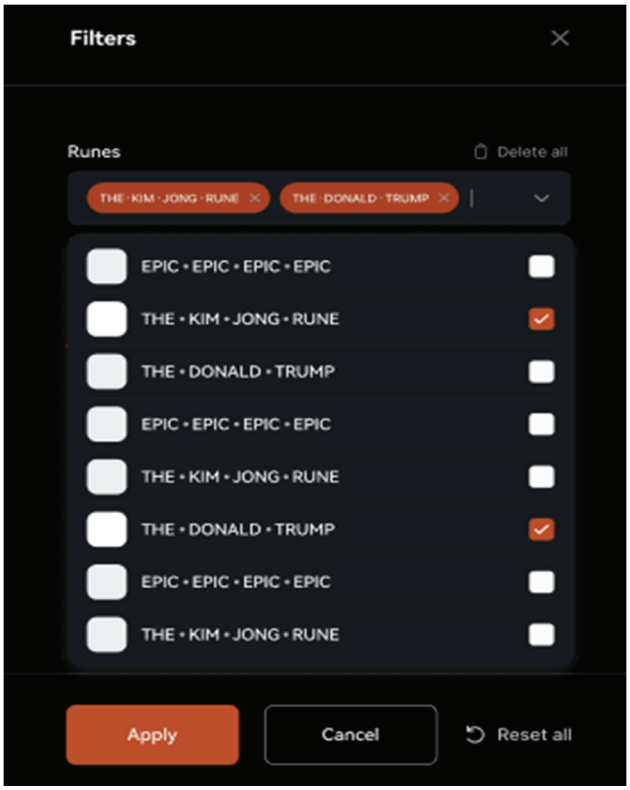

Рисунок 7.8 – Модальне вікно фільтру на сторінці «Runes»

При натисканні на кнопку підключення гаманця виводиться модальне вікно для підписання повідомлення своїм гаманцем(рисунок 7.9).

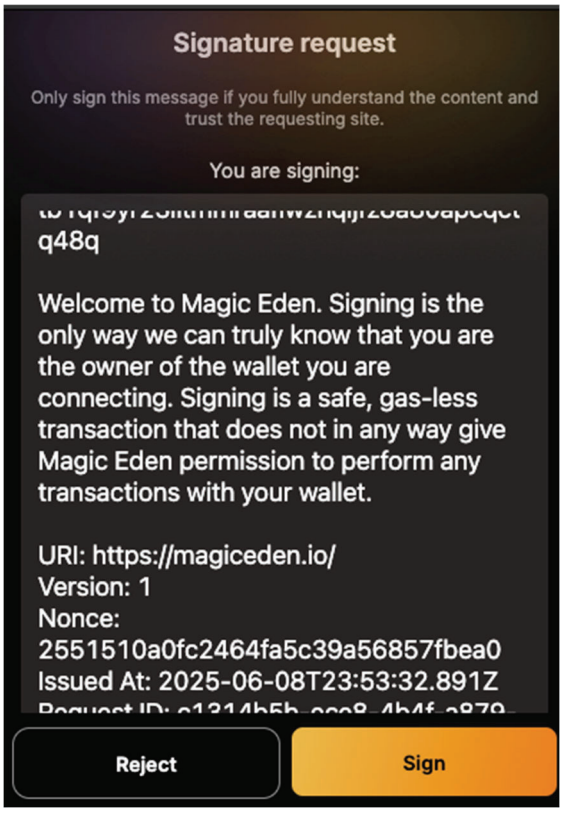

Рисунок 7.9 – Модальне вікно підписання повідомлення

Усі екрани взаємодіють із REST-API сервером через HTTPS, а дані оновлюються в реальному часі після виконання запитів до фонових сервісів індексації. Такий дизайн забезпечує зручність, швидкість та високу продуктивність користувацького досвіду.

8 РОЗРОБКА І ОБҐРУНТУВАННЯ ВИБОРУ ЕЛЕМЕНТІВ ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

При проєктуванні технічного забезпечення модуля “Моніторинг та аналіз взаємодій з рунами” у мережі «Bitcoin» було обрано мікросервісну архітектуру, що складається з п’яти основних компонентів: Bitcoin RPC, сервер декодування, REST-API сервер, фронтенд та сховище даних на базі PostgreSQL, розгорнутого у контейнері Docker.

Компонент Bitcoin RPC забезпечує прямий зв’язок із повноцінним вузлом Bitcoin. Він відповідає за отримання інформації про нові блоки і транзакції та передає сирі дані далі на сервер декодування. Для захисту RPC-з’єднання використовується Secure Shell (SSH) тунелювання зі списком дозволених IP-адрес та SSL/TLS [24].

Сервер декодування реалізовано як окремий сервіс з використанням Node.js, який в реальному часі підписується на оновлення блокчейну та обробляє транзакції, виявляючи runestone-повідомлення. Для підвищення продуктивності застосовано Worker Threads для паралельного декодування та батчеву обробку блоків. Всередині контейнера Docker на цьому сервері працює PostgreSQL для тимчасового зберігання даних.

REST-API сервер приймає запити від фронтенду та зовнішніх клієнтів, виконує аутентифікацію через підпис повідомлень приватним ключем Bitcoin-гаманця, перевірку підпису за допомогою RPC-методу verifymessage та повертає структуровані результати [25]. Використовується Redis для кешування найчастіших запитів та зменшення навантаження на базу даних.

Фронтенд реалізовано як SPA на React із використанням JWT для підтримки сесій. Він забезпечує інтерфейс для формування запитів, перегляду таблиць з інформацією про руни та візуалізації даних активності.

Сховище даних побудоване на PostgreSQL, запущеному в Docker-контейнері. Схема бази містить таблиці для UTXO-балансів, історії передач,

метаданих runestones та індексів оброблених блоків. Дані зберігаються в змонтованому томі для збереження стану після перезапусків.

Такий розподіл компонентів забезпечує високий рівень доступності та масштабованості – при зростанні навантаження сервери декодування та REST-API можна масштабувати горизонтально, а сховище даних переносити між кластерами. Контейнеризація за допомогою Docker.

Таблиця 8.1 – Апаратура для розгортання модуля «Моніторинг та аналіз взаємодій з рунами»

Компонент	Процесор	Оперативна пам’ять	Сховище	Мережевий інтерфейс
Bitcoin RPC сервер	Intel Xeon E-2246G, 6 ядер @ 3.6 GHz	32 GB DDR4	2× NVMe SSD 1 TB	10 GbE
Сервер декодування	AMD Ryzen 9 5900X, 12 ядер @ 3.7 GHz	64 GB DDR4	NVMe SSD 2 TB	10 GbE
REST-API сервер	Intel Xeon Silver 4216, 16 ядер @ 2.1 GHz	32 GB DDR4	SATA SSD 1 TB	1 GbE
PostgreSQL Docker-хост	Intel Core i7-12700, 12 ядер @ 3.6 GHz	32 GB DDR4	NVMe SSD 500 GB	1 GbE
Web-клієнт (ПК)	Intel Core i5-12400, 6 ядер @ 2.5–4.4 GHz	16 GB DDR4	SATA SSD 512 GB	1 GbE + Wi-Fi 6

Додатковим важливим компонентом технічної архітектури є Apache Kafka, яка використовується для організації асинхронної комунікації між сервером декодування та REST-API сервером і БД [34]. Kafka дозволяє

ефективно передавати великі обсяги повідомлень у реальному часі завдяки моделі продюсер-споживач та підтримці потокової обробки. Вона забезпечує високу пропускну здатність, низьку затримку і гарантує збереження даних при зростанні навантаження. Використання Kafka значно покращує масштабованість системи, дозволяючи легко додавати нові екземпляри сервісів, що обробляють дані, без втрати продуктивності.

При такому виборі обладнання забезпечується оптимальний баланс між продуктивністю, надійністю та вартістю. Кожен серверний компонент працює в ізольованому середовищі, що спрощує управління ресурсами та гарантує безперебійну роботу модуля.

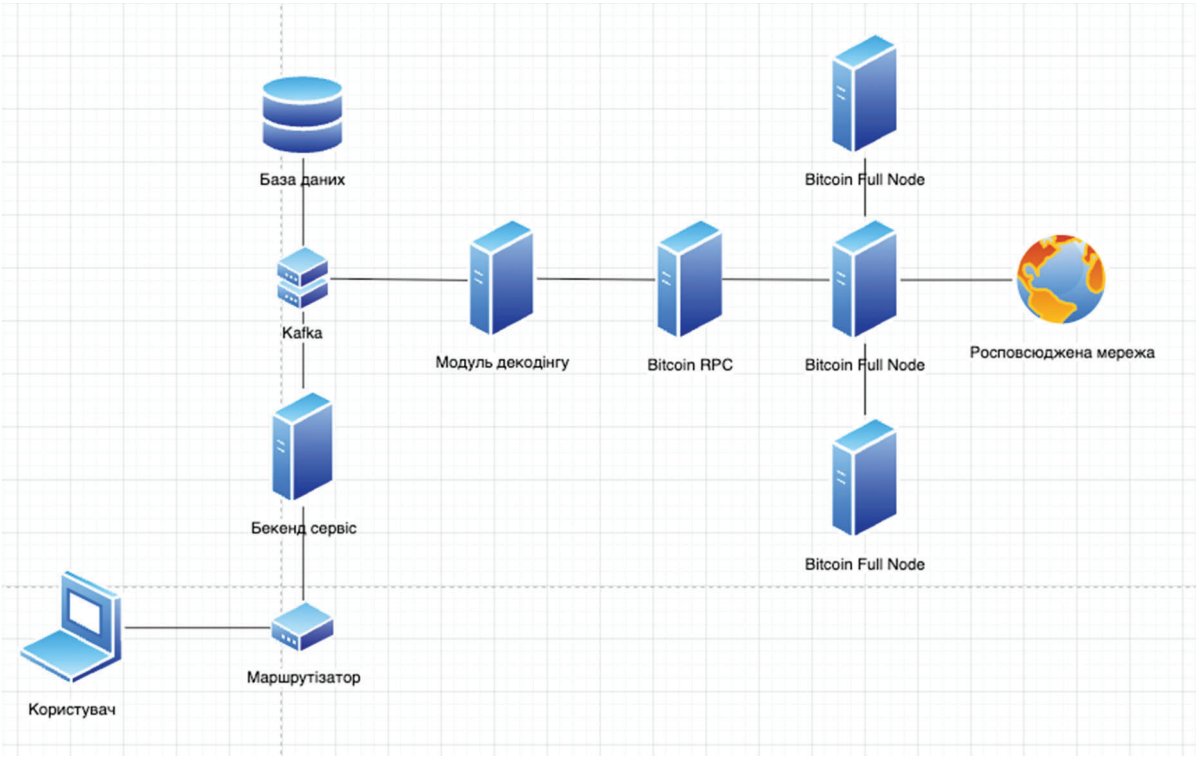

Рисунок 7.8 – Схема роботи технічного забезпечення роботи модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin»

9 СИНТЕЗ І ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ ЗАХИСТУ ІНФОРМАЦІЇ ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ

Несанкціонований доступ до модуля «Моніторинг та аналіз взаємодій з рунами» у мережі «Bitcoin» може призводити до компрометації конфіденційних даних про транзакції, створення фальшивих записів про обіг токенів та маніпуляцій аналітичними звітами. Для запобігання таким загрозам необхідно реалізувати багаторівневу систему захисту.

На рівні мережі використовується SSH-доступ, обмежений набором довірених IP-адрес з налаштуванням sshd_config (AllowUsers, відключення PasswordAuthentication), а також SSH-тунелювання для забезпечення зашифрованого каналу передачі даних між компонентами системи [26]. Міжмережевий екран (firewall) налаштовано через Uncomplicated Firewall або iptables, що закриває всі непотрібні порти та дозволяє лише необхідні сервіси, а Fail2Ban автоматично блокує спроби надмірних невдалих підключень.

На сервері індексації і аналітики застосовано централізоване логування (auditd, journald) з вивантаженням до SIEM-системи для оперативного виявлення підозрілих дій. Аудит доступу до критичних файлів та системних команд проводиться за допомогою системи моніторингу, що реєструє та аналізує події.

База даних PostgreSQL працює в режимі SSL з обов’язковим використанням сертифікатів для підключення (ssl = on, ssl_cert_file, ssl_key_file), а доступ до неї фільтрується за IP у pg_hba.conf [22]. Ролі в базі даних розподілені за принципом найменших привілеїв: окрема роль для читання аналітиками, ще одна - для запису фонових процесів індексації, та адміністраторська з правами повного доступу. Дані на рівні диску шифруються через LUKS, щоб унеможливити витік інформації в разі фізичного доступу до сховища.

Клієнтська сторона веб-інтерфейсу використовує криптографічне підписання запитів приватним ключем Bitcoin-гаманця. При формуванні API-запиту клієнт підписує nonce, що гарантовано унікальний, а сервер перевіряє підпис через RPC-метод verifymessage. Цей механізм забезпечує доведення права власності на запитувача та унеможливлює несанкціоновані запити.

Поєднання захисту на рівнях мережі, сервера, бази даних та клієнта створює надійний бар’єр від зовнішніх та внутрішніх загроз. Використання зашифрованих каналів зв’язку, суворих мережевих правил, ролей з найменшими привілеями, шифрування даних на диску та криптографічної аутентифікації користувачів гарантує, що тільки уповноважені учасники системи можуть ініціювати процеси моніторингу та аналізу rune-транзакцій.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено та впроваджено спеціалізований програмний модуль для моніторингу та аналізу взаємодій з цифровими активами на основі протоколу Runes у мережі Bitcoin.

Проведено комплексний аналіз існуючих технологій для декодування та аналізу транзакцій з рунами, який дозволив встановити, що існуючі інструменти, такі як bitcoinjs-lib або Bitcoin Core RPC, не мають достатнього функціоналу для ефективної роботи з метаданими протоколу Runes.

Обґрунтовано вибір спеціалізованих бібліотек (Runestone.ts або runestone-lib), які забезпечують ефективне декодування runestone-повідомлень і є найбільш придатними для створення спеціалізованих аналітичних інструментів.

Створено прототип модуля моніторингу та аналізу з використанням технології Runestone.ts, який включає автоматизоване виявлення транзакцій, декодування специфічних структур даних, перевірку валідності транзакцій і формування агрегованих аналітичних звітів.

Проведено експериментальне тестування на реальних даних з Bitcoin-транзакцій, що підтвердило працездатність модуля і його переваги у швидкості та точності декодування порівняно з існуючими універсальними інструментами.

Розроблено і реалізовано базу даних на основі реляційної моделі з використанням СУБД PostgreSQL, що дозволяє ефективно зберігати й обробляти великі обсяги даних, забезпечуючи швидкий доступ та високу продуктивність системи.

Запропоновано механізми захисту інформації, які включають авторизацію користувачів через криптографічні гаманці, а також автоматизовану систему оновлення і технічного обслуговування, що гарантує безперервну роботу та безпеку даних.

Запропоновано чітку структуру користувацького інтерфейсу, яка забезпечує зручність для аналітиків, трейдерів та розробників, дозволяючи їм оперативно отримувати інформацію щодо транзакцій, балансу токенів, активності адрес та прибутковості взаємодій з рунами.

Встановлено критерії ефективності, серед яких швидкість обробки даних, точність декодування, компактність бази даних, масштабованість системи та енергоефективність використання ресурсів.

Розроблений модуль дозволяє значно покращити процеси моніторингу та аналізу взаємодій з цифровими активами у мережі Bitcoin, що забезпечує користувачам більш точні й швидкі аналітичні дані, автоматизує рутинні процеси й підвищує загальний рівень безпеки і надійності системи. Перспективними напрямами подальших досліджень є розширення функціоналу модуля, інтеграція з іншими блокчейн-рішеннями, а також подальше вдосконалення методів аналізу великих обсягів даних з блокчейнів.

Кваліфікаційна робота виконувалась з використанням методичних вказівок до організації виконання та захисту кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти спеціальності 122 Комп’ютерні науки за освітньою програмою «Інформаційні технології управління» [31] та вимог, що були встановлені державними стандартами України [32], [33].

Результати роботи були представлені в доповідні на XXIX міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті», що відбувся 16 – 19 квітня 2025 року на базі Харківського національного університету радіоелектроніки [30].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ordiscan – Runes Explorer. URL: https://ordiscan.com/runes (дата звернення: 20.01.2025).

2. Ordinal Theory Handbook (Runes section) – офіційна документація до Runes від розробника Ordinals Кейсі Родармора. URL: https://docs.ordinals.com/runes.html (дата звернення: 23.03.2025).

3. Binance Research Report on Bitcoin tokens (2024) – аналітичний звіт щодо перспектив протоколу Runes та порівняння з іншими стандартами (BRC-20, Ordinals). URL: https://research.binance.com (дата звернення: 26.03.2025).

4. Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. – оригінальна стаття Сатоші Накамото. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 27.03.2025).

5. Magic Eden (ME) Foundation – «runestone-lib». Відкритий GitHub-репозиторій TypeScript-бібліотеки, що реалізує протокол Bitcoin Runes (Runestone) для Node.js і браузера. 2023. URL: https://github.com/me-foundation/runestone-lib (дата звернення: 28.03.2025).

6. Bitcoin Improvement Proposals (BIPs) – офіційні пропозиції вдосконалення Bitcoin. URL: https://github.com/bitcoin/bips (дата звернення: 03.04.2025).

7. Elliptic Curve Cryptography (secp256k1) – інформація щодо криптографії, яка використовується в Bitcoin. URL: https://www.secg.org/sec2-v2.pdf (дата звернення: 03.04.2025).

8. "UTXO vs Account model: pros and cons" – порівняння UTXO-моделі та Account моделі в блокчейнах. URL: https://medium.com/coinmonks/utxo-vs-account-model-9b33541d4a9c (дата звернення: 08.04.2025).

9. Mempool.space – Bitcoin block explorer. URL: https://mempool.space/

10. Kraken Learn: What is the Bitcoin Runes protocol? – детальний

огляд протоколу Runes. URL: https://www.kraken.com/learn/bitcoin-runes-protocol (дата звернення: 11.04.2025).

11. Офіційний GitHub-репозиторій JavaScript/TypeScript-бібліотеки для роботи з транзакціями Bitcoin. MIT License. 2013–2023. Доступно: https://github.com/bitcoinjs/bitcoinjs-lib (дата звернення: 14.04.2025).

12. RuneLib by sCrypt Inc. – бібліотека для роботи з протоколом Runes. URL: https://github.com/sCrypt-Inc/runelib (дата звернення: 17.04.2025).

13. Cointelegraph article by Brian Lindrea (2023) – стаття про альтернативу BRC-20 – протокол Runes. URL: https://cointelegraph.com/news/bitcoin-ordinals-creator-casey-rodarmor-pitches-brc-token-alternative-runes (дата звернення: 18.04.2025).

14. Casey Rodarmor Blog (Runes Introduction) – первинна пропозиція Родармора щодо Runes. URL: https://rodarmor.com/blog/runes (дата звернення: 18.04.2025).

15. Реляційна база даних. URL: https://ua5.org/database/189-reljacjjna-baza-danikh.html (дата звернення: 20.04.2025).

16. PostgreSQL. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 23.04.2025).

17. Flowchart Maker & Online Diagram Software. Flowchart Maker & Online Diagram Software. URL: https://app.diagrams.net/ (дата звернення: 29.05.2024).

18. React.js Documentation. URL: https://react.dev/ (дата звернення: 24.04.2025).

19. Next.js Documentation. URL: https://nextjs.org/docs (дата звернення: 15.04.2025).

20. ПРОТОКОЛ HTTP І HTTPS. HTTP & HTTPS Protocol. URL: https://cqr.company/ua/wiki/protocols/http-https-protocol/ (дата звернення: 06.06.2024). (дата звернення: 14.04.2025).

21. NestJS Documentation. URL: https://nestjs.com/ (дата звернення:

15.04.2025).

22. PostgreSQL: documentation. PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/docs/ (дата звернення: 31.05.2024).

23. pg-promise – PostgreSQL interface for Node.js – документація по взаємодії з БД через Node.js. URL: https://github.com/vitaly-t/pg-promise (дата звернення: 29.04.2025).

24. What is SSL/TLS Certificate? - SSL/TLS Certificates Explained - AWS. Amazon Web Services, Inc. URL: https://aws.amazon.com/what-is/ssl-certificate/?nc1=h_ls (дата звернення: 08.06.2024).

25. Bitcoin Core RPC documentation – документація до офіційного клієнта Bitcoin Core. URL: https://developer.bitcoin.org/reference/rpc/ (дата звернення: 01.04.2025).

26. What is SSH? https://www.ucl.ac.uk/isd/research-data-repository/what-ssh-and-how-do-i-use-it (дата звернення: 29.02.2025).

27. UniSat Runes Explorer. URL: https://unisat.io/runes (дата звернення: 22.04.2025).

28. Buterin, Vitalik. "On Bitcoin UTXO model" (2015) – аналіз моделі UTXO. URL: https://vitalik.ca/general/2015/01/20/utxo-model.html (дата звернення: 16.04.2025).

29. Casey Rodarmor Blog (Runes Introduction) – первинна пропозиція Родармора щодо Runes. URL: https://rodarmor.com/blog/runes/ (дата звернення: 09.05.2025).

30. Лещенко Д. С., Шеховцова В. І. Особливості застосування runestone.ts технології декодування транзакцій у мережі Bitcoin в задачі моніторингу та аналізу взаємодій з рунами в блокчейні // 29-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. матеріалів форуму. Т.6. (16-18 квітня 2025р.) Харків: ХНУРЕ, 2025. С. 219-221.

31. Методичні вказівки до організації виконання та захисту

кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти для студентів спеціальності 122 «Комп'ютерні науки» за освітньою програмою «Інформаційні технології управління». / Упоряд.: К.Е. Петров, А.В. Міхнова, М.С. Кудрявцева, М.В. Євланов, Т.І. Борисенко. – Електронне видання. – Харків: ХНУРЕ, 2023. – 68 с.

32. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання. / Видання офіційне. К.: ДП «УкрНДНЦ», 2016. 20 с.

33. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання, Чинний від 22.06.2015. Київ: ДП «УкрНДНЦ», 2016. 26 с.