Міністерство освіти і науки України
Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

ПЛАТІЖНА СИСТЕМА НА ОСНОВІ БЛОКЧЕЙНУ З ВЛАСНОЮ RPC-ІНФРАСТРУКТУРОЮ ДЛЯ МОНІТОРИНГУ ТРАНЗАКЦІЙ

BLOCKCHAIN-BASED PAYMENT SYSTEM WITH ITS OWN RPC INFRASTRUCTURE FOR TRANSACTION MONITORING

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІмз-21
Мотрій Дмитро Юрійович

(підпис)

Керівник:
к.т.н., доцент
Бортник Катерина Яківна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«____» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: «*Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарас Терлецький
«_____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Мотрію Дмитру Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Платіжна система на основі блокчейну з власною RPC-інфраструктурою для моніторингу транзакцій*

Керівник роботи: *Бортник Катерина Яківна, к.т.н., доцент*
затверджені наказом закладу вищої освіти від «18» жовтня 2025 року №434/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025 р.
3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*
4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Вступ.
Теоретичні відомості про блокчейн технології і її складові.
Аналіз серверного середовища та розробка платіжної системи.
Експерементальне дослідження власної RPC та порівняння з існуючими системами.
Висновки.
5. Перелік графічного (ілюстративного) матеріалу:
Рисунки блок схем програми, схеми моделей, рисунки опису алгоритмів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Теоретичні відомості про блокчейн технології	*Бортник К.Я., доцент*		
Аналіз серверного середовища та розробка платіжної системи	*Бортник К.Я., доцент*		
Експерементальне дослідження власної RPC та порівняння з існуючими системами.	*Бортник К.Я., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Гринюк С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., ст.викладач*		

7. Дата видачі завдання 18.10.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 01.11.2025 р.	
2.	*Аналіз серверного середовища та обґрунтування вибору технологій*	До 03.11.2025 р.	
3.	*Проєктування архітектури платіжної системи та розробка алгоритмів сканера транзакцій*	До 05.11.2025 р.	
4.	*Експериментальне дослідження власної RPC-інфраструктури та порівняльний аналіз з існуючими сервісами*	До 07.11.2025 р.	
5.	*Висновки та пропозиції*	До 09.11.2025 р.	
6.	*Формування списку використаних джерел*	До 13.11.2025 р.	
7.	*Формування додатків*	До 17.11.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 20.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 22.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Мотрій Д.Ю.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Бортник К.Я.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Мотрій Д. Ю. Платіжна система на основі блокчейну з власною RPC-інфраструктурою для моніторингу транзакцій. Рукопис.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено огляд сучасних блокчейн-систем, розглянуто технічні вимоги до RPC-інфраструктури, питання безпеки, моніторингу транзакцій, а також обґрунтовано вибір технологій Python, FastAPI та multiprocessing для побудови сервісів взаємодії з блокчейном.

У другому розділі описано процес розробки інструменту моніторингу транзакцій та розгортання власної RPC-архітектури. Деталізовано вимоги до серверного середовища, описано налаштування вузлів Geth та Erigon, реалізацію шардінгу, балансування запитів, кешування, логування та інтеграцію з Kafka і Redis для синхронізації mempool.

У третьому розділі представлено результати тестування архітектури в умовах навантаження, fault-injection сценарії, а також проведено порівняння продуктивності з типовими хмарними рішеннями. Оцінено ефективність розробленого рішення за показниками пропускної здатності, стабільності, вартості та надійності.

Ключові слова: блокчейн, RPC, платіжна система, FastAPI, multiprocessing, Geth, Erigon, моніторинг транзакцій, mempool, інфраструктура, Redis.

ANNOTATION

Motriy D. System for Monitoring Power Outages Based on the Arduino Platform and React Native Framework. Manuscript.

Master’s qualification thesis of the Educational Program «Computer Engineering», specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first chapter provides an overview of modern blockchain systems, analyzes technical requirements for RPC infrastructure, discusses security aspects, transaction monitoring, and substantiates the use of Python, FastAPI, and multiprocessing technologies for building blockchain interaction services.

The second chapter describes the development process of the transaction monitoring tool and the deployment of a custom RPC architecture. It details server environment requirements, node configuration using Geth and Erigon, implementation of sharding, request load balancing, caching, logging, and integration with Kafka and Redis for mempool synchronization.

The third chapter presents the results of infrastructure testing under load conditions, fault-injection scenarios, and performance comparison with typical cloud-based solutions. The efficiency of the developed solution is evaluated in terms of throughput, stability, cost, and reliability.

Keywords: blockchain, RPC, payment system, FastAPI, multiprocessing, Geth, Erigon, transaction monitoring, mempool, infrastructure, Redis.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах стрімкого розвитку блокчейн-технологій зростає потреба у побудові високопродуктивних, захищених та незалежних платіжних систем. Більшість сервісів, що працюють з Ethereum, покладаються на сторонні RPC-провайдери, такі як Infura, Alchemy чи Ankr. Хоча такі рішення спрощують інтеграцію, вони створюють низку обмежень: залежність від зовнішньої інфраструктури, нестабільність пропускної здатності, підвищену вартість при масштабуванні, а також неможливість повністю контролювати процес моніторингу транзакцій. Це є критично важливим у контексті платіжних систем, де точність, швидкість та надійність обробки транзакцій мають визначальне значення. Тому розробка власної RPC-інфраструктури та платіжної системи на її основі є актуальним завданням, яке дозволяє створити незалежний, масштабований і безпечний інструмент для роботи з блокчейн-транзакціями у режимі реального часу.

Метою роботи є розробка та впровадження платіжної системи на основі власної RPC-інфраструктури, яка забезпечує моніторинг, обробку та збереження блокчейн-транзакцій у реальному часі, а також порівняння її ефективності з існуючими ринковими рішеннями.

Об’єкт дослідження – блокчейн-орієнтовані платіжні системи та їх інфраструктурні компоненти.

Предмет дослідження – RPC-інфраструктура, механізми моніторингу транзакцій та архітектура платіжної системи на основі блокчейну Ethereum.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– проаналізувати теоретичні основи блокчейну, структуру Ethereum та принципи роботи RPC-інтерфейсів;

– дослідити можливості та обмеження хмарних RPC-провайдерів (Infura, Alchemy, Ankr) та визначити потребу у власному рішенні; спроєктувати та розгорнути серверне середовище для власного Ethereum-вузла з використанням Erigon;

– розробити RPC-інфраструктуру, яка забезпечує швидкий та безпечний доступ до блокчейну;

– реалізувати асинхронний сканер транзакцій та модуль моніторингу блоків у реальному часі;

– створити платіжну систему з унікальними депозитними адресами, обробкою інвойсів та автоматичним визначенням транзакцій;

– провести експериментальне порівняння продуктивності власного вузла з хмарними провайдерами;

– виконати порівняння створеної платіжної системи з комерційними сервісами CoinGate, Coinbase Commerce, BitPay та іншими аналогами;

– визначити переваги, обмеження та можливості подальшого вдосконалення системи.

Наукова новизна отриманих результатів полягає у розробці інтегрованої RPC-орієнтованої платіжної системи, що поєднує власний Ethereum-вузол, асинхронний сканер транзакцій та сервіс обробки інвойсів у єдину інфраструктуру. У роботі вперше комплексно досліджено питання продуктивності локального вузла Erigon у контексті фінансових сервісів та проведено порівняння з хмарними RPC-провайдерами та готовими платіжними рішеннями. Створена система забезпечує прямий доступ до даних блокчейну без зовнішніх залежностей, що відсутнє в більшості існуючих рішень на ринку.

Практичне значення одержаних результатів полягає у створенні високопродуктивної, масштабованої та автономної платіжної системи, здатної працювати без сторонніх провайдерів. Реалізована інфраструктура може бути використана в комерційних фінтех-проєктах, крипто-платформних сервісах, мерчант-платежах та системах автоматизації розрахунків. Наявність власного RPC дозволяє суттєво знизити витрати, підвищити швидкість обробки транзакцій, отримувати доступ до mempool, а також забезпечити адаптивну логіку моніторингу відповідно до потреб бізнесу.

Особистий внесок полягає у повному проєктуванні, розробці та впровадженні всієї системи, включаючи розгортання Ethereum-вузла Erigon,

налаштування RPC-інфраструктури, створення асинхронного сканера транзакцій, проєктування бази даних, розроблення серверної логіки обробки інвойсів, реалізацію API сервісу та проведення експериментальних досліджень продуктивності й порівняння з ринковими аналогами.

Апробація результатів. Основні результати дослідження були представлені на технічних семінарах та внутрішніх інженерних обговореннях. Матеріали дослідження опубліковано в науковому альманахові: СГ НТМ «Новий курс» (Харків, 2025) [26].

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО БЛОКЧЕЙН ТЕХНОЛОГІЇ ТА ЇЇ СКЛАДОВІ

1.1 Основи блокчейн-технологій

Блокчейн-технологія становить базову інфраструктуру сучасних децентралізованих платіжних систем і є фундаментом для побудови прозорих, незмінних та відмовостійких реєстрів транзакцій [5]. Її ключовою особливістю є відсутність централізованого контролю: управління історією операцій здійснюється колективно, великою кількістю незалежних вузлів [5]. Це дозволяє створювати системи, в яких неможливо непомітно змінити історію або підробити інформацію, що особливо важливо для фінансових застосунків [21].

У контексті цієї роботи блокчейн розглядається як технологічна основа для реалізації платіжної системи з власною RPC-інфраструктурою, оскільки саме блокчейн визначає правила формування транзакцій, моделі консенсусу, вимоги до обміну даними та принципи валідації стану мережі [16].

1.1.1 Суть та структура блокчейну

Блокчейн визначають як розподілений цифровий реєстр транзакцій, згрупованих у блоки, які пов’язані між собою за допомогою криптографічних хешів [5]. Кожен блок містить хеш попереднього блоку, мітку часу та дані транзакцій, зазвичай організовані за допомогою дерева Меркла [21]. Завдяки тому, що кожний блок містить посилання на попередній, утворюється безперервний ланцюжок блоків, у якому зміна даних у вже записаному блоці стає практично неможливою без внесення змін у всі наступні блоки та погодження цих змін більшістю мережі. Таким чином, блокчейн забезпечує незмінність даних: транзакції, записані у ланцюжку, не можуть бути відкориговані заднім числом без порушення цілісності всього реєстру.

Блокчейн-мережа функціонує у децентралізованому середовищі, тобто без єдиного центру зберігання чи контролю. Копії ланцюжка блоків зберігаються на багатьох вузлах мережі, і нові блоки додаються лише після колективної валідації

учасниками згідно з протоколом консенсусу. Після додавання нових блоків до ланцюжка всі вузли оновлюють свої копії реєстру, а можливі конфлікти (наприклад, ситуації одночасного додавання двох блоків) вирішуються автоматично за наперед визначеними правилами консенсусу. Така архітектура робить блокчейн стійким до відмов і зловмисних змін: для успішної фальсифікації запису зловмиснику знадобилося б одночасно змінити велику кількість копій реєстру на різних вузлах, що практично неможливо у великій мережі [21].

1.1.2 Роль децентралізації у забезпеченні цілісності платіжної системи

Однією з ключових переваг блокчейн-технології є відсутність центрального керуючого органу, що в традиційних платіжних системах відповідає за підтвердження та зберігання транзакцій. У централізованих системах усі дані зберігаються й обробляються на одному сервері або під контролем однієї установи (наприклад, банку чи державного реєстру). Натомість у блокчейні підтримання реєстру розподілено між багатьма незалежними вузлами мережі, жоден з яких не має повного контролю над системою. Така децентралізація усуває єдину точку відмови і зменшує ризик цензури або махінацій з боку окремого учасника.

Децентралізована модель забезпечує цілісність платіжної системи за рахунок колективної перевірки транзакцій. Усі вузли повинні дійти згоди щодо правильності чергового блоку, перш ніж включити його до ланцюжка. Це означає, що атака на систему вимагає координації більшості учасників, що є вкрай малоймовірним у великій мережі. Блокчейн-системи вважаються захищеними за дизайном і володіють високою візантійською відмовостійкістю (Byzantine fault tolerance), тобто здатні продовжувати коректну роботу навіть за умов збоїв або зловмисних дій частини вузлів [18]. Досягнення консенсусу значною кількістю незалежних учасників гарантує, що записані транзакції залишаються достовірними, навіть якщо частина мережі зазнає збоїв або спроб компрометації [16].

Децентралізація також створює нову модель довіри: учасники мережі не зобов’язані довіряти один одному або третій стороні (наприклад, банку), а лише протоколу та алгоритмам консенсусу [5]. Завдяки цьому технологія блокчейн дозволяє вибудувати надійний реєстр транзакцій між сторонами, що не знайомі і не мають попереднього довірчого відношення. Фактично, прозорість записів у публічному блокчейні та їх криптографічна захищеність замінюють собою традиційних посередників, забезпечуючи цілісність даних і стійкість до шахрайства [16].

1.1.3 Принципи роботи транзакцій у публічному блокчейні

У публічному блокчейні (такому, як Bitcoin або Ethereum) кожна транзакція являє собою передачу цифрового активу між учасниками, захищену криптографічними методами. Кожна транзакція підписується відправником за допомогою його приватного криптографічного ключа, що підтверджує автентичність транзакції і право відправника розпоряджатися відповідними коштами [5]. Цей механізм цифрового підпису забезпечує, що сторонні особи не можуть підробити транзакцію від імені іншого користувача: тільки власник приватного ключа має змогу ініціювати дійсний платіж. При цьому публічний ключ (адреса) отримувача фігурує у транзакції як адресат, і сам факт підписання транзакції приватним ключем відправника служить «доказом» її легітимності.

Транзакції в мережі блокчейн є прозорими: вони транслюються (розсилаються) з вузла відправника до всіх інших вузлів мережі [16]. Кожен вузол, отримавши нову транзакцію, виконує її перевірку, зокрема, переконується, що транзакція правильно підписана та не дублює вже витрачені кошти (захист від подвійного витрачання). Валідовані транзакції потрапляють до пулу непідтверджених транзакцій (мемпулу), звідки їх беруть майнери (або валідатори) для включення до наступного блоку.

Після того, як майнер сформував новий блок транзакцій і додав його до ланцюжка, відповідні транзакції вважаються підтвердженими. Всі вузли оновлюють у себе копію реєстру, включаючи цей новий блок. З цього моменту транзакції, що увійшли до блоку, отримують одне «підтвердження»

(confirmation). З додаванням кожного наступного блоку кількість підтверджень зростає, підвищуючи рівень довіри до незворотності транзакції. В публічних блокчейнах, де можливі розгалуження ланцюга, загальноприйнятою практикою є очікування кількох підтверджень (наприклад, шести у біткоїні) перед тим, як вважати транзакцію остаточно завершеною і непіддатливою до відкату [1].

Важливо підкреслити, що у публічному блокчейні всі записи транзакцій доступні для перегляду будь-якому учаснику або сторонньому спостерігачу (хоча сторони транзакції зазвичай виступають під псевдонімними адресами). Ця відкритість, поєднана з криптографічним захистом, забезпечує високий рівень довіри до системи: будь-хто може перевірити, що транзакція справді включена до блокчейну, і не була змінена після цього.

Через децентралізований характер блокчейн-мережі та можливість існування декількох конкуруючих гілок ланцюга (так званих форків) підтвердження транзакції у блокчейні має ймовірнісний характер. Це означає, що навіть після включення транзакції до блоку існує мала ймовірність, що цей блок може бути відкинутий (наприклад, якщо інша гілка блокчейну виявиться довшою). Повна гарантія незворотності транзакції настає лише після певної кількості підтверджень, коли ризик її скасування стає практично нульовим. Такий підхід відповідає моделі кінцевої узгодженості (eventual consistency): система досягає консистентного стану не негайно, а після певної затримки, коли всі вузли мережі синхронізуються на одній «версії правди».

У блокчейні на базі Proof-of-Work (наприклад, біткоїн) прийнято вважати транзакцію фіналізованою після отримання кількох блоків зверху (класично 6 блоків для біткоїна, що займає ~1 годину) [8]. Ця емпірична практика ґрунтується на тому, що ймовірність успішного переписування історії експоненціально зменшується з кожним новим блоком, доданим до версії ланцюга, що містить дану транзакцію. По суті, коли транзакція «похована» під достатньою кількістю наступних блоків, можна з високим рівнем впевненості говорити, що вона не буде відкликана. Це і є фіналізація транзакції в умовах

eventual consistency: консенсус досягається остаточно через деякий час, а не миттєво [5].

Слід зазначити, що існують інші алгоритми консенсусу (переважно у системах з обмеженим доступом або новітніх публічних ланцюгах), які забезпечують майже миттєву або детерміністичну фіналізацію блоків. Прикладом є консенсуси на основі візантійської відмовостійкості (BFT) Algorand, Tendermint (Cosmos) чи практична візантійська домовленість у Hyperledger Fabric, де блок вважається остаточно підтвердженим, щойно за нього проголосувала кваліфікована більшість валідаторів [6]. У таких системах модель узгодженості ближча до сильної (eventual consistency зводиться до мінімальної затримки). Проте у класичних публічних блокчейнах на кшталт Bitcoin чи Ethereum (до переходу на PoS) зберігається саме принцип поступової узгодженості: система завжди обирає найдовшу (або «найскладнішу» з точки зору витраченої роботи) гілку як актуальну, і лише з часом всі чесні вузли мережі гарантовано сходяться на єдиному стані реєстру [11].

Модель eventual consistency впливає на те, як користувачі та додатки працюють з блокчейном. Наприклад, біржі криптовалют часто чекають кілька підтверджень перед зарахуванням депозиту, страхуючись від можливого реорганізації ланцюга. З точки зору архітектури розподілених систем, блокчейн жертвує негайною консистентністю заради високої доступності та толерантності до мережевих розділень. Це означає, що система завжди доступна для прийому нових транзакцій і продовження роботи, навіть якщо деякі вузли відключені або сполучення між ними уповільнене, але узгодженість (єдність даних) досягається із затримкою. В підсумку, eventual consistency блокчейн-мережі гарантує, що за відсутності нових збоїв усі вузли зрештою матимуть однаковий журнал транзакцій, і підтверджені транзакції будуть зберігатися в ньому назавжди [10].

1.1.4 Порівняння блокчейн-технологій із централізованими системами

Блокчейн-технологія суттєво відрізняється від традиційних централізованих систем зберігання даних і проведення платежів. В централізованій системі (наприклад, у банківській платіжній платформі або в

класичній базі даних) існує єдиний центр, що контролює всі транзакції: центральний сервер або організація відповідає за перевірку правомірності операцій, ведення балансу рахунків та захист даних [21]. Користувачі такої системи змушені довіряти цьому центру, який виступає посередником і арбітром у всіх транзакціях. У блокчейні, навпаки, усі ці функції розподілені між багатьма учасниками мережі, а сам процес верифікації транзакцій здійснюється колективно за допомогою алгоритму консенсусу. Відсутність єдиного контролера означає, що жоден суб'єкт не може одноосібно змінити записи або скасувати операції; для цього знадобилася б згода (або компрометація) значної частини мережі.

Перевагою децентралізованого підходу є підвищена стійкість і прозорість. Дані в блокчейні доступні для перевірки усіма учасниками, що унеможливлює приховування або несанкціоновану правку інформації. Крім того, відсутність посередників знижує витрати та бар'єри при здійсненні платежів: транзакції можуть відбуватися напряму між сторонами, без банків або процесингових центрів, що особливо важливо для міжнародних переказів та інших випадків, де традиційно присутні значні накладні витрати. Як зазначають фахівці, блокчейн забезпечує довіру та цілісність даних без залучення «традиційних посередників, таких як банки чи державні органи», зменшуючи ризик шахрайства та людських помилок.

З іншого боку, централізовані системи наразі перевершують блокчейн за показниками продуктивності та масштабованості. Банківські платіжні мережі або платіжні шлюзи (на кшталт Visa/MasterCard) здатні обробляти тисячі транзакцій за секунду з мінімальною затримкою, тоді як публічні блокчейни (наприклад, біткоїн) обмежені пропускною спроможністю у кілька транзакцій на секунду через необхідність виконання складних криптографічних обчислень і розповсюдження кожного блоку по всій мережі [21]. Централізована база даних може миттєво підтвердити операцію, тоді як у розподіленому реєстрі потрібен час на досягнення консенсусу та фіналізації блоку (як розглядалось у попередньому підрозділі). Таким чином, eventual consistency блокчейну є платою

за децентралізацію: система виграє у надійності та незалежності від центральної довіри, проте поступається у швидкодії та зручності негайного підтвердження.

Ще один аспект – це можливість централізованого втручання та гнучкість. У традиційній системі уповноважений адміністратор може відкотити помилкову транзакцію, заморозити рахунок чи виправити дані у разі збою або за рішенням суду. В блокчейні ж записи незмінні: якщо кошти були надіслані помилково або стали результатом компрометації ключа, їх неможливо повернути без згоди отримувача (за винятком хіба що форс-мажорного рішення спільноти про «хардфорк», що трапляється вкрай рідко і означає створення нової версії ланцюга). Таким чином, відповідальність за безпеку та правильність операцій більше лежить на самих користувачах. З одного боку, це підвищує автономність (кожен повністю контролює свої кошти, немає центру, який міг би їх заблокувати), але з іншого – вимагає більшої обачності та не залишає місця для третьої сторони, що могла б розв’язати спір [5].

Отже, блокчейн-технології пропонують нову парадигму, що вигідно відрізняється в питаннях децентралізації, прозорості та стійкості до несанкціонованих змін, але водночас накладає обмеження на швидкість та потребує зміщення довіри: від централізованих інститутів до самої математично обґрунтованої моделі. У контексті платіжних систем це означає, що блокчейн може забезпечити глобальну, доступну 24/7 інфраструктуру для розрахунків без посередників, з високим рівнем захисту від підробок і збоїв, проте для досягнення рівня продуктивності традиційних систем необхідні подальші технічні вдосконалення. Баланс між цими підходами визначається конкретними вимогами застосування: в деяких сценаріях (особливо там, де на перше місце виходить довіра та незалежність) переваги блокчейну є вирішальними, тоді як в інших (де критична масова швидкість обробки) централізовані рішення поки що зберігають перевагу.

1.2 RPC та його роль у блокчейні

У побудові платіжної системи на основі блокчейну ефективна взаємодія між клієнтськими застосунками та розподіленою мережею має ключове значення. Саме цю взаємодію забезпечує протокол віддалених процедур (Remote Procedure Call, RPC), який виступає базовим інтерфейсом доступу до даних блокчейну та ініціювання транзакцій. Розуміння принципів роботи RPC, його інфраструктури та викликів у багаторівневих або шардованих мережах є необхідною умовою для створення надійної системи моніторингу транзакцій та забезпечення масштабованості. У цьому підрозділі проаналізовано фундаментальні аспекти RPC, стандарт JSON-RPC, порівняно локальні й хмарні вузли, а також розглянуто особливості маршрутизації запитів у сучасних блокчейн-архітектурах [16].

1.2.1 Принципи роботи RPC у децентралізованих системах

RPC-інтерфейси є невід'ємним елементом більшості блокчейн-застосунків. Для платіжної системи на основі блокчейну, розуміння принципів роботи RPC є критично важливим. Remote Procedure Call (RPC) – це протокол, що дозволяє клієнтській програмі викликати процедури на віддаленому сервері так, ніби вони виконуються локально [5]. У розподілених та децентралізованих системах RPC виступає абстракцією, яка приховує складність мережевої взаємодії та робить віддалений виклик невідмінним від звичайного локального виклику процедури.

У контексті блокчейну RPC використовується як основний механізм зв'язку між зовнішніми застосунками (наприклад, гаманцями, сервісами або dApp) та вузлами децентралізованої мережі. Фактично, RPC слугує інтерфейсом, що дає змогу програмно взаємодіяти з блокчейном – отримувати дані з розподіленого реєстру, надсилати транзакції, викликати функції смарт-контрактів тощо [20]. RPC у блокчейн-мережах реалізує класичну модель клієнт-сервер: клієнт (користувацький застосунок) формує запит певного формату до сервера – вузла блокчейну, який містить необхідні дані або може

виконати запитані дії. Вузол отримує та обробляє цей запит (наприклад, шукає баланс адреси чи включає нову транзакцію в блок), після чого надсилає клієнту відповідь з результатом виконання запиту [16]. Завдяки цьому містку, зовнішні системи можуть працювати з блокчейном без потреби запускати повноцінний вузол локально, доручивши виконання операцій віддаленим вузлам.

RPC-запити, які надсилаються до блокчейн-вузлів, як правило, мають формат запиту-відповіді і можуть передаватися через різні транспортні протоколи (наприклад, HTTP(S) або WebSocket). У відповідь на RPC-запит сервер-вузол повертає або потрібні дані (наприклад, стан рахунку, деталі блока, лог подій смарт-контракту), або підтвердження виконання дії (наприклад, хеш транзакції при її успішному відправленні) залежно від типу виклику. Важливо підкреслити, що RPC в блокчейні дозволяє абстрагуватися від деталей реалізації консенсусу чи зберігання даних: розробник взаємодіє із зрозумілим інтерфейсом, не турбуючись про внутрішню структуру вузла. Це спрощує інтеграцію блокчейну в додатки і робить можливим моніторинг транзакцій у режимі реального часу засобами зовнішніх програм. Саме тому RPC вважається «тилом» взаємодії Web3-застосунків із блокчейн-інфраструктурою [21].

1.2.2 JSON-RPC як стандарт у блокчейн-мережах (Ethereum, BNB Chain, Polygon)

У більшості сучасних блокчейнів для зовнішньої взаємодії з вузлами використовується протокол JSON-RPC – легковаговий stateless-протокол, що передає виклики у форматі JSON. Запит містить назву методу, параметри та ідентифікатор, а відповідь – результат або помилку. Завдяки простоті й уніфікованості JSON-RPC став стандартом у багатьох мережах, а Ethereum, що використовує специфікацію JSON-RPC 2.0, фактично сформував галузевий еталон [15].

Ethereum-клієнти (Geth, OpenEthereum) реалізують широкий набір методів, зокрема eth_getBalance, eth_sendRawTransaction, eth_getBlockByNumber, eth_call тощо. Через ці методи зовнішні додатки можуть отримувати стан блокчейну,

надсилати транзакції та підписані дані, а також отримувати події через WebSocket-підписки – усе через єдиний уніфікований інтерфейс [15].

Оскільки Ethereum є основною платформою для смарт-контрактів, інші EVM-сумісні мережі перейняли його RPC-стандарти. Такі блокчейни, як BNB Chain та Polygon, відповідають на ті самі методи eth_*, завдяки сумісності з Ethereum-клієнтами (BNB – форк Geth, Polygon Bor – модифікований Go-Ethereum) [2, 19]. Це дозволяє розробникам працювати з різними мережами через однакові RPC-виклики, значно спрощуючи створення багатомережевих платіжних систем і сервісів моніторингу.

JSON-RPC має текстовий формат і не залежить від транспортного протоколу, що робить його простим для реалізації. Альтернативи на кшталт gRPC пропонують вищу продуктивність завдяки двійковому формату, але є складнішими й менш поширеними в блокчейн-інфраструктурі [14]. Тому JSON-RPC 2.0 залишається домінуючим способом взаємодії з вузлами, забезпечуючи сумісність інструментів (web3.js, ethers.js, web3.py) з будь-яким EVM-вузлом. Для платіжних систем це означає інтероперабельність: модуль моніторингу може працювати з різними блокчейнами без зміни логіки запитів [4, 24, 18].

1.2.3 Інфраструктура RPC: локальні вузли, хмарні провайдери (Infura, Alchemy)

Під час створення власної RPC-інфраструктури для моніторингу платежів важливо обрати підхід: використовувати хмарні провайдери (Infura, Alchemy, QuickNode, Ankr) чи розгортати власні вузли блокчейну. Обидва варіанти мають переваги й обмеження, а вибір залежить від вимог щодо масштабу, бюджету, навантаження та рівня контролю [19].

Хмарні RPC-провайдери надають готові до використання вузли з доступом через API-ключ і RPC-endpoint. Переваги такого підходу – швидке впровадження, висока доступність, технічна підтримка та додаткові сервіси (аналітика, прискорення транзакцій). Infura використовується як стандартний бекенд у багатьох dApp, включно з MetaMask, а Alchemy пропонує поглиблені

інструменти моніторингу. Це оптимальний варіант для швидкого старту та прототипування.

Недоліки хмарних провайдерів – залежність від третьої сторони, тарифні ліміти (зокрема, кількість запитів у безкоштовних планах), а також вартість, яка зростає зі збільшенням навантаження. Для систем, що здійснюють масове відстеження транзакцій, витрати можуть сягати сотень доларів на місяць. Крім того, передача критичної функції сторонньому сервісу може створювати ризики для безпеки та довіри [19].

Власний RPC-вузол забезпечує повний контроль над даними та інфраструктурою. Немає зовнішніх лімітів чи політик, а дані отримуються безпосередньо з блокчейну, що особливо важливо для платіжних систем. Можна розгортати архівні вузли або налаштовувати власні індекси для потреб моніторингу [14].

Однак запуск повного вузла вимагає ресурсів: великого SSD-сховища, стабільного інтернету, оперативної пам’яті та часу на тривалу синхронізацію. Витрати на обслуговування (хмарні сервери, DevOps-підтримка, оновлення клієнтів) можуть становити сотні доларів на місяць і вимагати постійного технічного нагляду [14].

Гібридний підхід часто є оптимальним: на етапі розробки використовувати Infura чи Alchemy, а в продуктиві – власні вузли з можливістю резервного перемикання на провайдера. Це поєднує контроль та автономність власної інфраструктури з масштабованістю та надійністю хмарних сервісів [19].

Для проєктованої платіжної системи доцільно використовувати власні RPC-вузли як основне джерело даних, залишаючи можливість автоматичного fallback на хмарного провайдера для забезпечення безперебійності моніторингу транзакцій.

1.3 Безпека інфраструктури

З розвитком децентралізованих платіжних систем питання безпеки інфраструктури набуває особливого значення. Вразливість RPC-інтерфейсів, витік ключів чи несанкціонований доступ до ноди можуть призвести до фінансових втрат і порушення роботи всієї системи. У цьому підрозділі розглянуто основні загрози для RPC-рівня, механізми захисту мережевої взаємодії, підходи до автентифікації клієнтів та сучасні методи зберігання ключів і підпису транзакцій. Надійна безпека – критичний елемент будь-якої інфраструктури, що працює з блокчейн-транзакціями.

1.3.1 Типові загрози для RPC-інтерфейсів

Віддалений RPC-інтерфейс блокчейн-вузла є одним із найбільш вразливих елементів інфраструктури, оскільки відкриває можливість прямої взаємодії із системою. Поширеною загрозою є DDoS-атаки, під час яких зловмисник засипає відкритий RPC-вузол великою кількістю запитів або навмисно викликає ресурсоємні методи на кшталт eth_getLogs із великим діапазоном блоків. Такі запити змушують вузол виконувати складні операції, що призводить до значного падіння продуктивності, втрати синхронізації або повного відключення сервісу. Навіть без доступу до внутрішніх компонентів, неправильно захищений RPC дозволяє зловмиснику фактично вивести інфраструктуру з ладу [15].

Ще однією критичною проблемою є несанкціоновані запити. JSON-RPC протокол не містить вбудованих механізмів автентифікації, тому відкритий RPC-порт дає можливість будь-кому виконувати команди. Особливо небезпечні випадки, коли на вузлі активовано адміністративні методи (personal_*, admin_*), що дозволяють розблоковувати акаунти або ініціювати транзакції від імені користувача. Відомі інциденти, коли зловмисники шляхом масового сканування порту 8545 знаходили незахищені Ethereum-клієнти та переводили кошти з гаманців власників. Така помилка конфігурації неодноразово ставала причиною фінансових втрат, що обчислювалися мільйонами доларів [15].

Окрему категорію становлять ризики компрометації ключів через RPC. Якщо вузол виконує роль гаманця і зберігає приватні ключі локально, то недостатньо захищений інтерфейс може дозволити зловмиснику здійснювати несанкціоноване підписування транзакцій або отримувати доступ до конфіденційних даних. Деякі вразливості в реалізації RPC-серверів Ethereum у минулому навіть дозволяли віддалене читання файлової системи чи примусове розблокування акаунтів, що створювало реальну загрозу витоку приватних ключів [22].

Узагальнюючи, найтиповіші загрози RPC-інтерфейсів пов’язані з перевантаженням вузла шкідливим трафіком, виконанням небезпечних методів без контролю доступу та можливістю компрометації приватних ключів у разі помилок конфігурації чи вразливостей. Розуміння цих ризиків є ключовим для розробки безпечної архітектури RPC-рівня в платіжних системах.

1.3.2 Захист мережевого рівня

Захист мережевого рівня відіграє ключову роль у запобіганні несанкціонованому доступу до RPC-ендпоінтів та зменшенні ризику мережевих атак. Першою лінією оборони виступає брандмауер, який обмежує доступ до RPC-порту лише довіреним IP-адресам або внутрішнім мережам. Правильно налаштовані мережеві екрани та групи безпеки унеможливлюють випадкове відкриття RPC в Інтернет, тим самим запобігаючи його скануванню та небажаним підключенням.

Для додаткової ізоляції трафіку часто використовують VPN, що дозволяє здійснювати всі RPC-запити через зашифрований приватний канал. У такій архітектурі зовнішні клієнти спершу проходять автентифікацію у VPN, і лише після цього отримують доступ до вузла. Це фактично приховує RPC-ендпоінт від глобальної мережі та різко зменшує поверхню потенційних атак [9].

Не менш важливим елементом захисту є використання TLS/SSL. Усі зовнішні звернення до RPC повинні передаватися лише через HTTPS, що забезпечує шифрування й захищає від атак типу «людина посередині». Використання довірених сертифікатів у продуктивному середовищі є

обов’язковим, адже відсутність TLS створює ризик перехоплення транзакційних даних, API-токенів та інших чутливих відомостей [9].

Додатковий рівень безпеки забезпечують системи WAF, які аналізують HTTP(S)-трафік і блокують потенційно небезпечні запити. Розміщення RPC за реверс-проксі з увімкненим WAF (наприклад, nginx із ModSecurity, AWS WAF або Cloudflare WAF) дозволяє фільтрувати аномальну активність, зокрема автоматизовані атаки, некоректні виклики методів чи шкідливі сигнатури, ще до того, як такий трафік досягне блокчейн-вузла.

У комплексі ці заходи формують надійний контур мережевої безпеки, що суттєво ускладнює несанкціонований доступ і гарантує стійкість RPC-інфраструктури до зовнішніх загроз.

1.3.3 Аутентифікація і авторизація RPC-клієнтів

Для підвищення безпеки RPC-інтерфейсу мережевої ізоляції недостатньо, необхідно також впровадити надійні механізми аутентифікації та авторизації, які гарантуватимуть, що до вузла звертаються лише довірені клієнти й у межах дозволених повноважень. Найпоширенішим способом контролю доступу є використання API-токенів: кожному сервісу видається унікальний секретний ключ, який він має передавати з кожним запитом. Подібний механізм використовують Infura, NOWNodes та інші провайдери, а сам токен дозволяє не тільки ідентифікувати клієнта, а й накладати ліміти або розмежовувати доступ між різними середовищами. Важливим аспектом є захист самих токенів – вони не повинні зберігатися у відкритому вигляді в коді або у публічних конфігураціях, щоб уникнути компрометації [19].

У складніших та корпоративних сценаріях застосовуються протоколи JWT та OAuth2. У цьому випадку клієнт попередньо отримує токен доступу від авторизаційного сервера й передає його при кожному запиті до RPC. Такі токени можуть містити набір дозволів, що визначають, які саме методи або ресурси клієнт має право викликати. Наприклад, у Quorum реалізована підтримка JWT, де вузол може перевіряти не лише валідність токена, а й його вміст, блокуючи небажані методи незалежно від того, чи надійшов запит від знайомого клієнта.

Багато інфраструктурних провайдерів (зокрема Infura) також підтримують авторизацію на основі JWT як додатковий, більш гнучкий рівень безпеки [23].

Додатковим механізмом контролю доступу є фільтрація за IP-адресами. Вона дозволяє приймати RPC-з'єднання тільки з визначених мереж і тим самим значно зменшує ризики зловживань навіть у випадку викрадення токена. Інфраструктурні платформи на кшталт Chainstack або Infura надають можливість встановлювати allowlist-списки IP чи доменів для кожного проєкту. До мережевих правил також належить відключення невикористовуваних RPC-методів. Заборона модулів, що не потрібні для роботи системи (наприклад, personal у Geth), повністю усуває цілу категорію атак, оскільки небезпечний виклик просто не існуватиме в інтерфейсі [7, 20].

Таким чином, надійна аутентифікація та авторизація – це поєднання секретних токенів, централізованих систем керування доступом і статичних обмежень на рівні IP та RPC-методів. Разом вони створюють ефективний бар'єр, який мінімізує ризики несанкціонованого доступу до вузла навіть у складному загрозовому середовищі.

1.3.4 Захист ключів та підпис транзакцій

Захист приватних ключів є критично важливим елементом безпеки платіжної системи на блокчейні, оскільки компрометація ключа дає повний контроль над коштами. Основні підходи до безпечного зберігання та підпису транзакцій включають:

HSM – це спеціалізовані пристрої для генерації та зберігання ключів у ізольованому середовищі. Приватні ключі ніколи не виходять за межі апаратного модуля, а операція підпису виконується всередині нього. Навіть у разі компрометації сервера зловмисник не зможе отримати ключі. Сучасні HSM (Thales Luna, AWS CloudHSM) відповідають стандартам FIPS 140-2 і можуть інтегруватися з Ethereum-клієнтами через PKCS#11 або спеціальні плагіни. Це найбільш надійний підхід, який застосовується фінансовими установами та біржами [22].

Апаратні гаманці – пристрої на зразок Ledger чи Trezor також виконують підписування в захищеному чіпі, діючи як «міні-HSM». У інфраструктурі їх можна використовувати для підтвердження критичних транзакцій або роботи з резервними фондами. Ethereum-клієнти підтримують їх через зовнішній підписувач Clef. Основний недолік – низька швидкодія та необхідність ручного підтвердження, що робить їх менш придатними для автоматизованих високонавантажених систем [17].

Зовнішні підписувачі – рішення на кшталт Ethereum Clef дозволяють винести зберігання ключів і підпис транзакцій за межі вузла. Вузол передає транзакцію зовнішньому сервісу, який здійснює підписування відповідно до своїх політик. Clef працює як ізольований процес, шифрує сховище ключів і не довіряє зовнішнім командам, що забезпечує додатковий рівень безпеки. Навіть якщо RPC-вузол скомпрометовано, ключі залишаються недоступними [14].

Мультипідписні (multi-sig) схеми та смарт-контрактні-гаманці, смарт-контракти на зразок Gnosis Safe дозволяють використовувати схему M-з-N, де для виконання транзакції потрібно кілька підписів. Це знімає залежність від одного ключа та істотно підвищує безпеку. Навіть якщо один ключ буде викрадено, зловмисник не зможе здійснити переказ без інших підтверджень. Multi-sig особливо корисний для організаційних фінансів та великих платежів, а також зменшує ризики, пов’язані з компрометацією RPC-інфраструктури [3].

РОЗДІЛ 2
АНАЛІЗ СЕРВЕРНОГО СЕРЕДОВИЩА ТА РОЗРОБКА ОСНОВИ ПЛАТІЖНОЇ СИСТЕМИ

2.1 Вибір серверного середовища та підготовка AWS-інфраструктури

Ефективне функціонування RPC-вузла у платіжній системі безпосередньо залежить від правильного вибору серверного обладнання та конфігурації інфраструктури. Оскільки вузол Ethereum працює з великим обсягом даних, виконує складні операції з бінарними структурами та обслуговує зовнішні запити, необхідно забезпечити достатній запас ресурсу як на рівні обчислень, так і на рівні дискової продуктивності. У межах цієї роботи розгортання здійснюється на платформі Amazon Web Services (AWS), що дозволяє досягти високого рівня доступності та масштабованості інфраструктури.

2.1.1 Вимоги до апаратного забезпечення

У процесі вибору апаратних характеристик важливо враховувати специфіку роботи клієнта Erigon, який використовується як основний вузол блокчейну. Це високопродуктивний клієнт Ethereum, оптимізований для ефективного використання процесора, оперативної пам’яті та, особливо, дискової підсистеми. Під час синхронізації вузол завантажує й обробляє десятки гігабайт даних, виконує індексацію блоків і транзакцій у власному форматі зберігання та активно використовує операції читання і запису на диск.

Для стабільної роботи Erigon потребує інстансу з достатньою кількістю ресурсів. У плані процесорної продуктивності оптимальним вважається використання від чотирьох до восьми vCPU, оскільки клієнт активно застосовує багатопотоковість і виграє від високої продуктивності кожного окремого ядра. Обсяг оперативної пам’яті також відіграє важливу роль: незважаючи на оптимізацію споживання RAM, робота зі state-даними та обробка блоків потребують щонайменше 16-32 GB пам’яті.

Найважливішим компонентом є дискова підсистема. Erigon показує найкращу продуктивність на NVMe-накопичувачах із високою пропускною

здатністю, де швидкість читання та запису сягає від 3 до 7 GB/s. Стандартні SSD EBS не здатні забезпечити необхідний IO-рівень для повноцінної роботи. З огляду на обсяг даних Ethereum та індексів, що створюються під час синхронізації, рекомендований розмір диску становить близько 2-4 TB. Окрім цього, для вузла критично важливе швидкісне мережеве підключення, оскільки в процесі синхронізації він інтенсивно обмінюється блоками та транзакціями з мережею. Практичним мінімумом є канал зі швидкістю не менше 1 Gbps.

Такі апаратні вимоги визначають подальший вибір інстансу в AWS та параметри конфігурації файлової системи, які забезпечують збалансовану роботу вузла й можливість його стабільної довготривалої експлуатації.

2.1.2 Вибір AWS-сервісів та EC2 для розгортання RPC-вузла

Для розгортання RPC-вузла в середовищі AWS у рамках даного проєкту було обрано сервіс Amazon EC2, який надає можливість створення віртуальних серверів із точно визначеними параметрами продуктивності. Це дозволило адаптувати інфраструктуру під специфічні потреби клієнта Erigon, що використовується як основний вузол Ethereum і потребує високої швидкості дискових операцій та стабільної роботи CPU.

У процесі вибору типу інстансу було враховано кілька ключових вимог. Насамперед, Erigon вимагає наявності локального NVMe-сховища, оскільки лише такі диски забезпечують необхідну пропускну здатність під час обробки великого потоку IO-операцій та індексації блоків. Тому інстанси, що використовують виключно EBS-диски, одразу були виключені з розгляду.

Другим критичним параметром став обсяг оперативної пам’яті. Для стабільної обробки state-даних та зниження затримок у роботі RPC потрібний інстанс із RAM не менше 60 GB, що дозволяє уникнути частих swap-операцій і забезпечує стабільність системи під навантаженням. Також, приділено увагу стабільності процесорної продуктивності, тому burstable-інстанси не розглядалися, оскільки вони можуть знижувати частоту CPU залежно від накопичених кредитів.

З урахуванням цих критеріїв були проаналізовані кілька сімейств EC2-інстансів, що потенційно могли підходити для задач вузла. Порівняння наведено в таблиці 2.1.

Таблиця 2.1 – Порівняльні характеристики EC2

Сімейство	Опис	Переваги	Недоліки
m6i / m7i	Збалансовані інстанси загального призначення	Достатнє співвідношення CPU та RAM	Відсутність локальних NVMe-дисків
c6i / c7i	Оптимізовані для обчислювальних навантажень	Висока продуктивність CPU	Також не мають NVMe, залежать від EBS
i3 / i4i	Інстанси, оптимізовані під інтенсивні IO-навантаження	Локальні NVMe-диски, висока швидкість читання/запису	Вища вартість порівняно з іншими варіантами
r6i	Інстанси зі збільшеним обсягом пам’яті	Підходять для RAM-інтенсивних процесів	Відсутність NVMe та недостатня IO-швидкість

Після порівняння всіх доступних варіантів було прийнято рішення обрати інстанси сімейства i3 / i4i, оскільки саме вони повністю відповідають вимогам проєкту. На відміну від інших сімейств, ці інстанси мають локальні NVMe-накопичувачі із дуже високою пропускною здатністю, що критично важливо для швидкої первинної синхронізації Erigon та подальшої роботи вузла під навантаженням.

Крім того, інстанси i3 та i4i містять достатній обсяг оперативної пам’яті (понад 60 GB), що дозволяє ефективно працювати з великими state-областями Ethereum, а також забезпечують стабільну процесорну продуктивність без будь-яких обмежень чи механізмів «burst mode».

У рамках реалізації системи був обраний інстанс із такою конфігурацією:

- тип інстансу: i3.2xlarge або i4i.2xlarge;
- процесор: 8 vCPU;
- оперативна пам’ять: 61-64 GB RAM;
- локальне NVMe-сховище: 1,9-2,3 TB.

Завдяки такому вибору інфраструктура отримала високу швидкість IO-операцій, достатню продуктивність для індексації та обробки блоків, а також стабільну роботу RPC-сервісу на тривалому проміжку часу. Інші сімейства EC2 не могли забезпечити потрібної швидкості дискової підсистеми або мали занадто обмежені характеристики, тому не були прийняті до розгляду як основа для даного проєкту.

2.1.3 Налаштування EC2 та операційної системи

Після запуску EC2-інстансу розпочинається процес його підготовки до роботи як основного вузла RPC-інфраструктури. Насамперед виконується налаштування мережевого оточення. Інстанс розміщується в межах заздалегідь створеної VPC, що забезпечує ізольоване середовище та дає змогу контролювати маршрутизацію, трафік і доступ до вузла. Одним із ключових елементів є коректна конфігурація Security Groups, які виконують роль мережевого фільтра. У межах цього проєкту відкрито лише ті порти, які необхідні для функціонування вузла: порт 22 для SSH-доступу адміністратора, порти 30303 TCP/UDP для роботи Ethereum P2P-протоколу та порти 8545/8546 для RPC-взаємодії, доступ до яких дозволений виключно з приватної мережі або через VPN (рис. 2.1). Такий підхід дозволяє мінімізувати поверхню атаки та забезпечує контрольований доступ до інфраструктури на всіх етапах роботи.

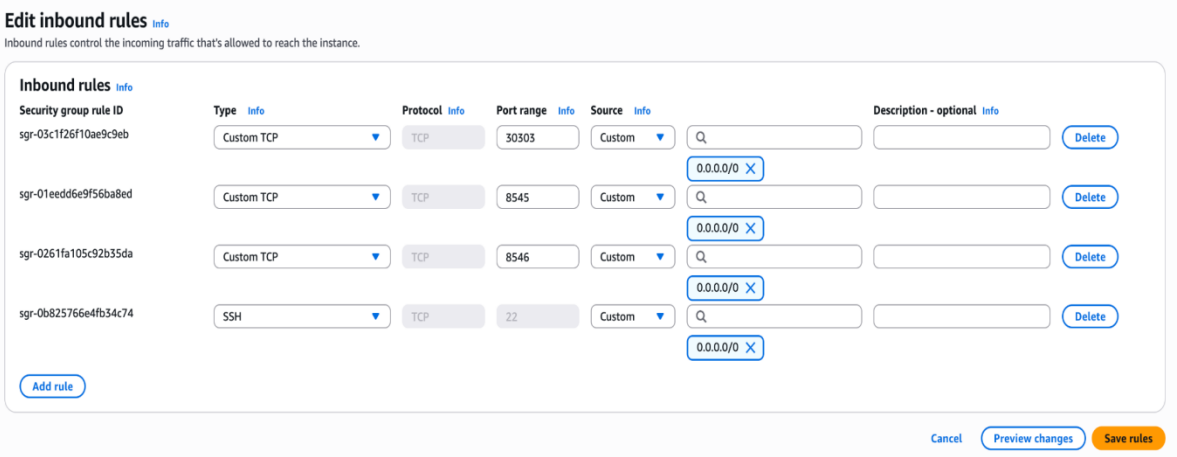

Рисунок 2.1 – Відкриття портів

Після налаштування мережевого середовища виконується підготовка операційної системи. Для сервера використано Ubuntu Server 22.04 LTS як стабільну й довготривало підтримувану платформу, оптимальну для побудови високонавантажених сервісів. Перший вхід через SSH супроводжується оновленням системних пакетів, встановленням останніх патчів безпеки та необхідних інструментів для адміністрування. На цьому етапі також створюється окремий системний користувач, під яким буде запускатися клієнт Erigon. Такий підхід дозволяє ізолювати процеси вузла від адміністративних привілеїв і підвищити загальний рівень безпеки. Додатково конфігурується SSH-доступ: вимикається автентифікація за паролем, забороняється прямий вхід root-користувача, а список дозволених IP-адрес обмежується лише адміністративними вузлами. Усе це формує надійний базовий рівень захисту сервера.

Окрему увагу необхідно приділити дисковій підсистемі, оскільки продуктивність вузла Erigon безпосередньо залежить від швидкості IO-операцій. EC2-інстанси сімейств i3/i4i містять локальні NVMe-диски, які забезпечують пропускну здатність, недосяжну для стандартних EBS-томів. Після запуску інстансу перевіряється наявність NVMe-пристроїв за допомогою системних утиліт, після чого диск форматують у файлову систему EXT4 та монтують у каталог /data. Використання EXT4 пов’язане з його стабільністю, низькою латентністю та передбачуваною поведінкою під високим навантаженням, що є особливо важливим при індексації блоків та роботі зі state-даними.

Для подальшої роботи створюється організована структура каталогів, яка розділяє дані Erigon, журнали та конфігураційні файли. У каталозі /data/erigon розміщуються основні дані вузла та база blockchain-стану, /data/logs використовується для зберігання логів синхронізації та RPC-викликів, а /data/config – для конфігураційних файлів та параметрів запуску. Чітка структуризація дає змогу спростити обслуговування системи, полегшити моніторинг і забезпечити зручність подальшого масштабування або міграції.

Завершення всіх описаних дій формує повністю готове серверне середовище для розгортання клієнта Erigon. Інстанс отримує оптимальне поєднання безпеки, високої продуктивності та ізольованості, що є необхідною основою для стабільної роботи RPC-інфраструктури. У подальших етапах налаштовується сам Erigon, його параметри запуску, синхронізація та інтеграція з іншими компонентами системи.

2.2 Вибір технологічного стеку

Функціонування платіжної системи базується на взаємодії між блокчейном Ethereum, внутрішньою базою даних, RPC-інфраструктурою та сервісами бізнес-логіки. Тому вибір технологічного стеку відіграє ключову роль у забезпеченні стабільності й масштабованості всієї системи.

Ethereum було обрано як основну блокчейн-платформу завдяки широкій підтримці токенів стандарта ERC-20, стабільності темпу генерації блоків та величезній екосистемі інструментів. Саме Ethereum сьогодні використовується більшістю світових криптопроцесингових сервісів, що підтверджує його придатність для комерційних фінансових застосунків.

Клієнт Erigon став ключовим елементом інфраструктури завдяки своїй високій продуктивності, оптимізованій роботі з NVMe-накопичувачами, низькій затримці RPC-викликів і швидкій синхронізації. Його архітектура дозволяє опрацьовувати блоки значно швидше, ніж це роблять традиційні клієнти на кшталт Geth або Nethermind, що робить його найкращим вибором для платіжних систем з високою інтенсивністю читання даних.

Основна частина програмної логіки реалізована на Python, що забезпечує високу швидкість розробки, наявність асинхронних інструментів (asyncio, AsyncWeb3, SQLAlchemy async) і доступність великої кількості стабільних бібліотек для взаємодії з Ethereum та побудови API [25].

Зберігання структур користувачів, гаманців, інвойсів і транзакцій реалізовано на PostgreSQL, оскільки ця СУБД забезпечує повну підтримку

ACID-транзакцій, складне індексування та високу продуктивність під змішаними навантаженнями. Redis використовується як допоміжний інструмент для збереження поточного стану сканера, що дозволяє відновлювати роботу без втрат даних навіть після аварій.

2.3 Архітектура сканера

Сканер транзакцій Ethereum є центральним компонентом платіжної системи, оскільки синхронізує стан внутрішньої бази даних із блокчейном. Його архітектура побудована на асинхронній моделі, що дозволяє одночасно отримувати нові блоки та обробляти їх у декількох воркерах, забезпечуючи високу продуктивність та стійкість (додаток Б).

Структура охоплює модуль зв’язку з RPC, модуль отримання нових блоків, чергу задач, пул асинхронних воркерів для паралельної обробки транзакцій, а також механізми запису результатів у внутрішні сервіси та базу даних. Така структура дає змогу ізолювати навантаження, уникнути блокувань та масштабувати систему в залежності від кількості вхідних даних (рис. 2.2).

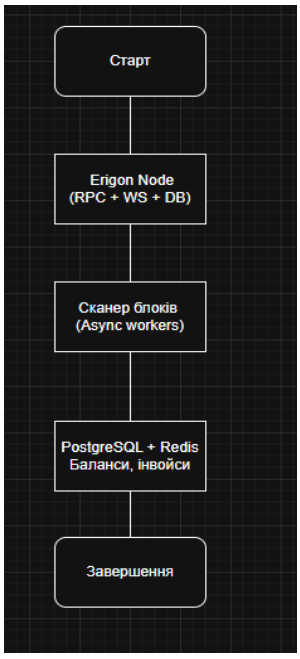

Рисунок 2.2 – Структура комунікації

Сканер взаємодіє з Erigon через RPC-інтерфейс. Головний цикл отримує номер актуального блоку, порівнює його з даними в Redis, визначає пропущені та додає їх у чергу (рис. 2.3).

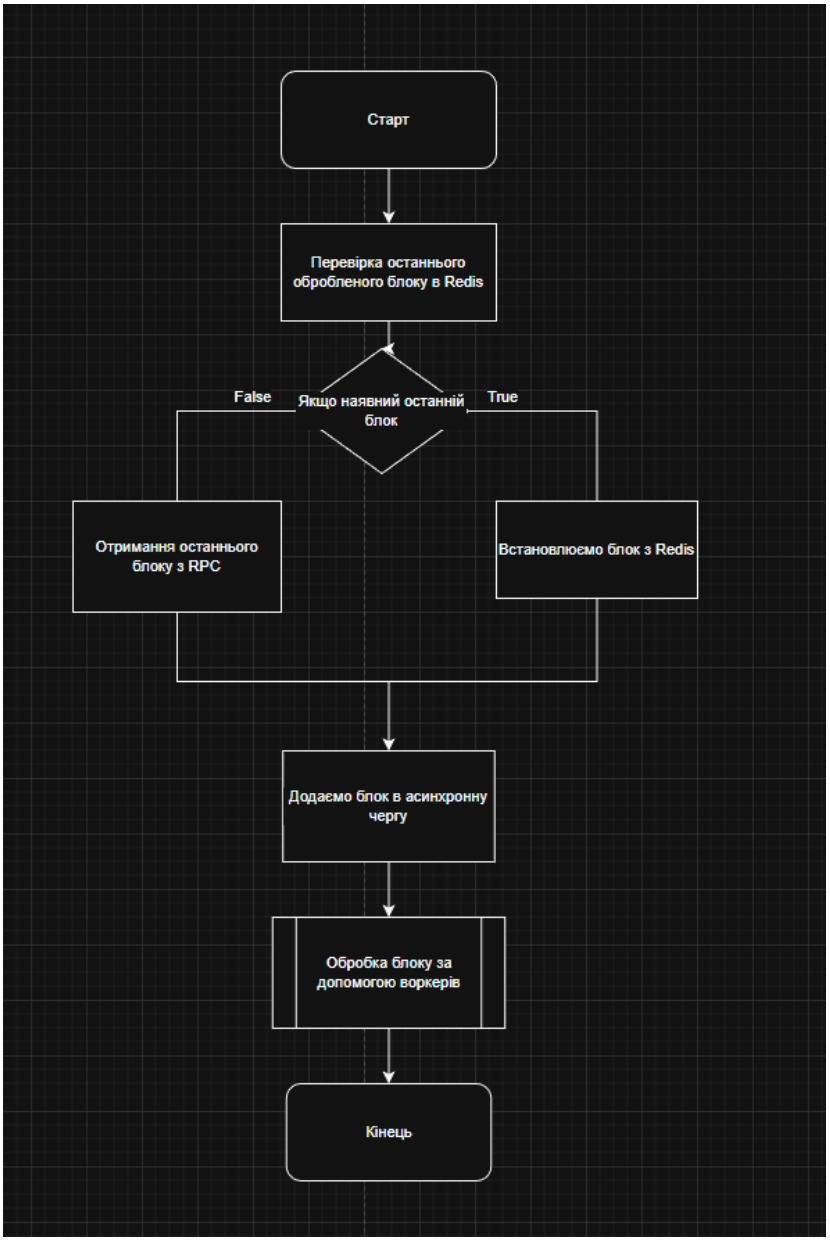

Рисунок 2.3 – Блок схема роботи сканера

Декілька воркерів паралельно беруть блоки з черги, обробляють транзакції, виявляють операції, що стосуються внутрішніх гаманців, оновлюють баланси й статуси інвойсів (рис. 2.4).

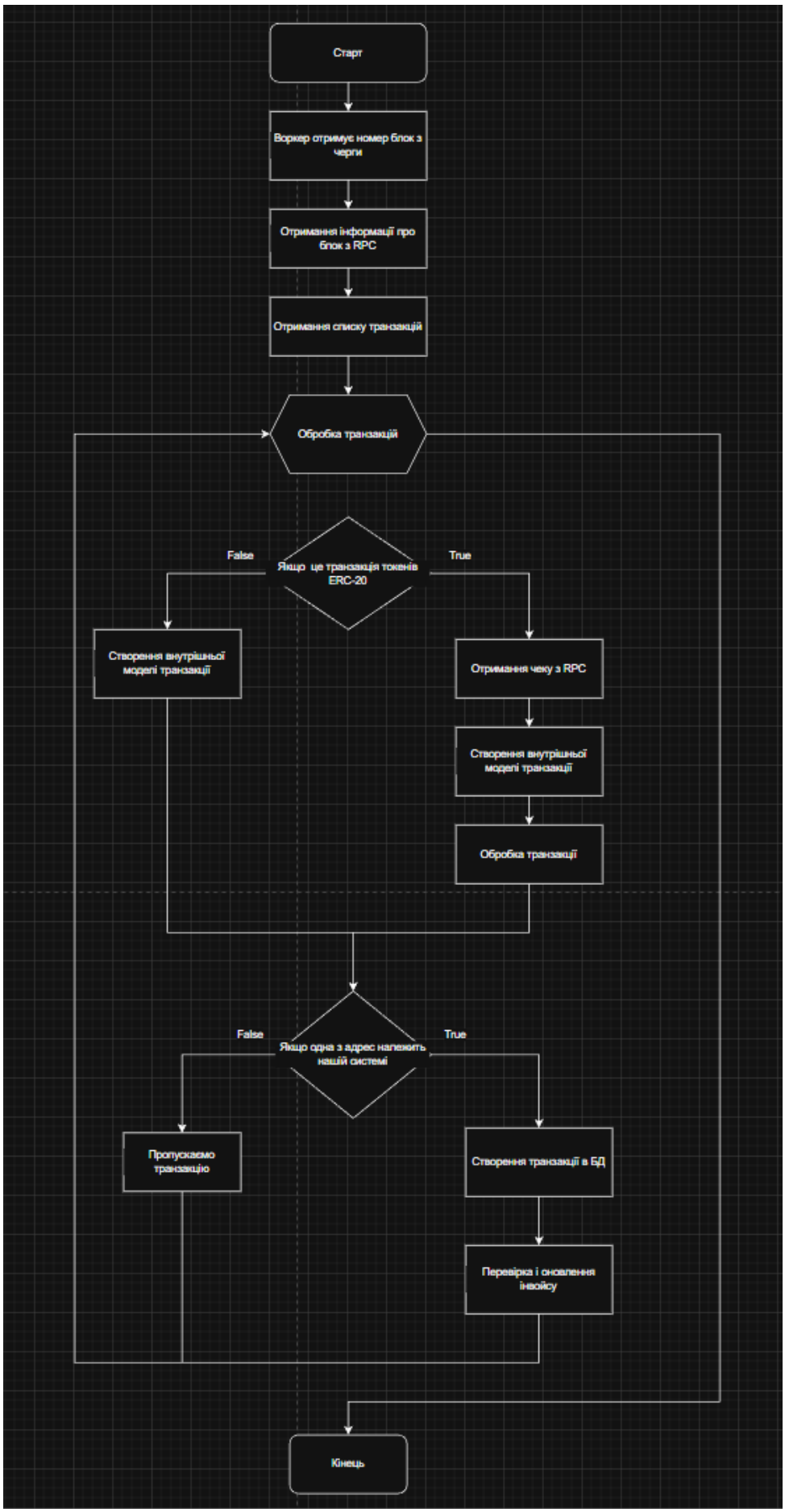

Рисунок 2.4 – Блок схема роботи воркерів сканера

2.4 Реалізація архітектури платіжної системи

Реалізація платіжної системи охоплює повний цикл обробки платежу – від створення інвойсу до його автоматичного закриття після отримання транзакції в мережі Ethereum (додаток В). Уся логіка побудована таким чином, щоб процес відбувався повністю автоматично та не вимагав участі оператора.

Ключовою особливістю розробленої архітектури є застосування принципу розмежування відповідальності між модулем взаємодії з клієнтом та підсистемою моніторингу блокчейну. Використання технології HD Wallet дозволяє генерувати нові платіжні реквізити детермінованим шляхом без необхідності постійного звернення до ноди мережі, що суттєво зменшує затримки при обробці вхідних HTTP-запитів.

Цикл починається з того, що користувач або зовнішній сервіс надсилає запит на створення інвойсу, передаючи адресу електронної пошти, валюту та суму. Система знаходить відповідного користувача в базі даних і визначає валюту платежу. Після цього за допомогою модуля HDWallet генерується унікальна депозитна адреса, яка буде використана виключно для оплати саме цього інвойсу. Адреса зберігається у таблиці wallet із позначенням типу «інвойсний гаманець».

Паралельно створюється запис у таблиці invoice, що містить суму в людському форматі, суму у wei, статус «очікує оплати» та ідентифікатор гаманця. Таким чином формується повноцінний платіжний документ, прив’язаний до конкретного користувача й окремого блокчейн-гаманця.

Після створення інвойсу система переходить у режим очікування надходження платежу, і далі вся робота передається сканеру блоків. Сканер безперервно отримує найновіші блоки та завантажує транзакції. Якщо транзакція містить адресу, що належить системі, баланс відповідного гаманця оновлюється. Для нативних ETH-переказів це відбувається за значенням value. Для токенів ERC-20 (додаток Г) система аналізує журнали подій і виділяє суму переказу з логів Transfer (рис. 2.5).

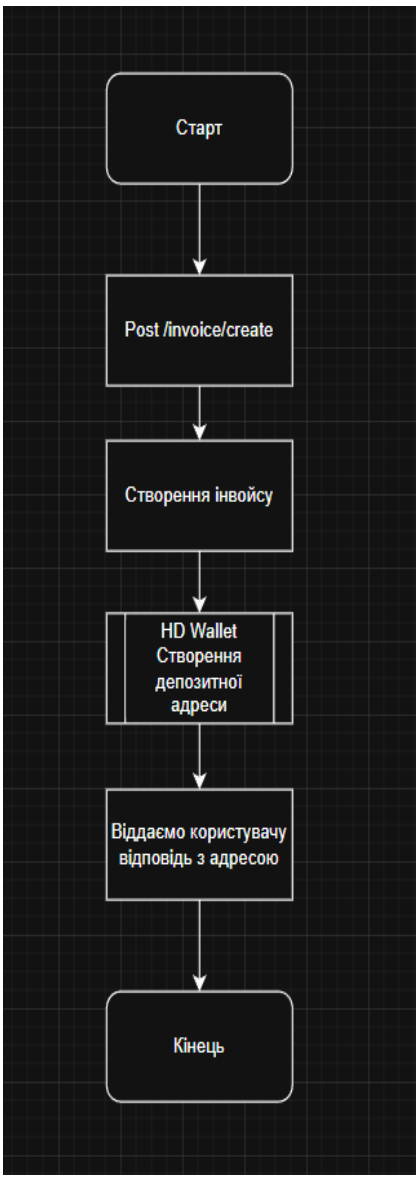

Рисунок 2.5 – Блок схема створення інвойсу

Коли на інвойсний гаманець надходять кошти, система перевіряє, чи досягнуто необхідної суми для закриття інвойсу. Якщо баланс дорівнює або перевищує очікувану суму, інвойс автоматично переходить у статус «оплачено». Одночасно створюється запис у таблиці transaction, що дозволяє відстежувати історію платежів.

Система працює повністю автономно: сканер виявляє платежі, оновлює баланс, створює транзакції та закриває інвойси без будь-якого ручного втручання. Завдяки асинхронній архітектурі Python і використанню Redis для відстеження прогресу сканер може обробляти великі обсяги даних у реальному часі.

РОЗДІЛ 3
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ВЛАСНОЇ RPC ТА ПОРІВНЯННЯ З ІСНУЮЧИМИ СИСТЕМАМИ

3.1 Встановлення та конфігурація клієнта Erigon

Після підготовки серверного середовища та оптимізації апаратної інфраструктури наступним етапом є встановлення й налаштування клієнта Erigon, який використовується як основний вузол для роботи RPC-інфраструктури. Цей процес включає отримання програмного забезпечення, конфігурацію параметрів вузла, визначення директорій для зберігання даних, а також підготовку сервісних механізмів для автоматичного запуску та контролю стану Erigon у виробничому середовищі.

Erigon був обраний як основний Ethereum-клієнт через його архітектурні особливості, орієнтовані на високу продуктивність і ефективність використання ресурсів. Завдяки оптимізованій роботі з диском, паралельній індексації, використанню власних форматів зберігання й багатопотоковій обробці блоків, Erigon значно швидше проводить синхронізацію та забезпечує низьку затримку при виконанні RPC-запитів порівняно з іншими клієнтами.

Встановлення клієнта може виконуватися як з попередньо зібраних бінарних релізів, так і за допомогою Docker-контейнерів. У межах цього проєкту було обрано варіант із використанням Docker, оскільки контейнеризація гарантує відтворюваність середовища, спрощує оновлення, дозволяє стандартно ізолювати залежності й легко інтегрується з майбутньою інфраструктурою моніторингу. Після встановлення Docker створюється окрема робоча директорія в каталозі /data/erigon, де зберігатимуться база даних, завантажені блоки, state-дані та індекси, які генерує клієнт під час синхронізації.

Перед першим запуском формується конфігурація Erigon, яка визначає ключові параметри вузла: шлях до зберігання даних, режими індексації, тип RPC-інтерфейсу, дозволені методи, підтримувані протоколи й структуру логування. Зокрема, важливими є параметри, які визначають рівень індексації

історичних даних та активність модулів rpcdaemon і sentinel (рис. 3.1). Для підтримки власного RPC-сервісу Erigon запускається з увімкненими HTTP та WebSocket-інтерфейсами, а доступ до них обмежується приватною мережею, що відповідає вимогам безпеки й виключає можливість прямого доступу до публічного RPC.

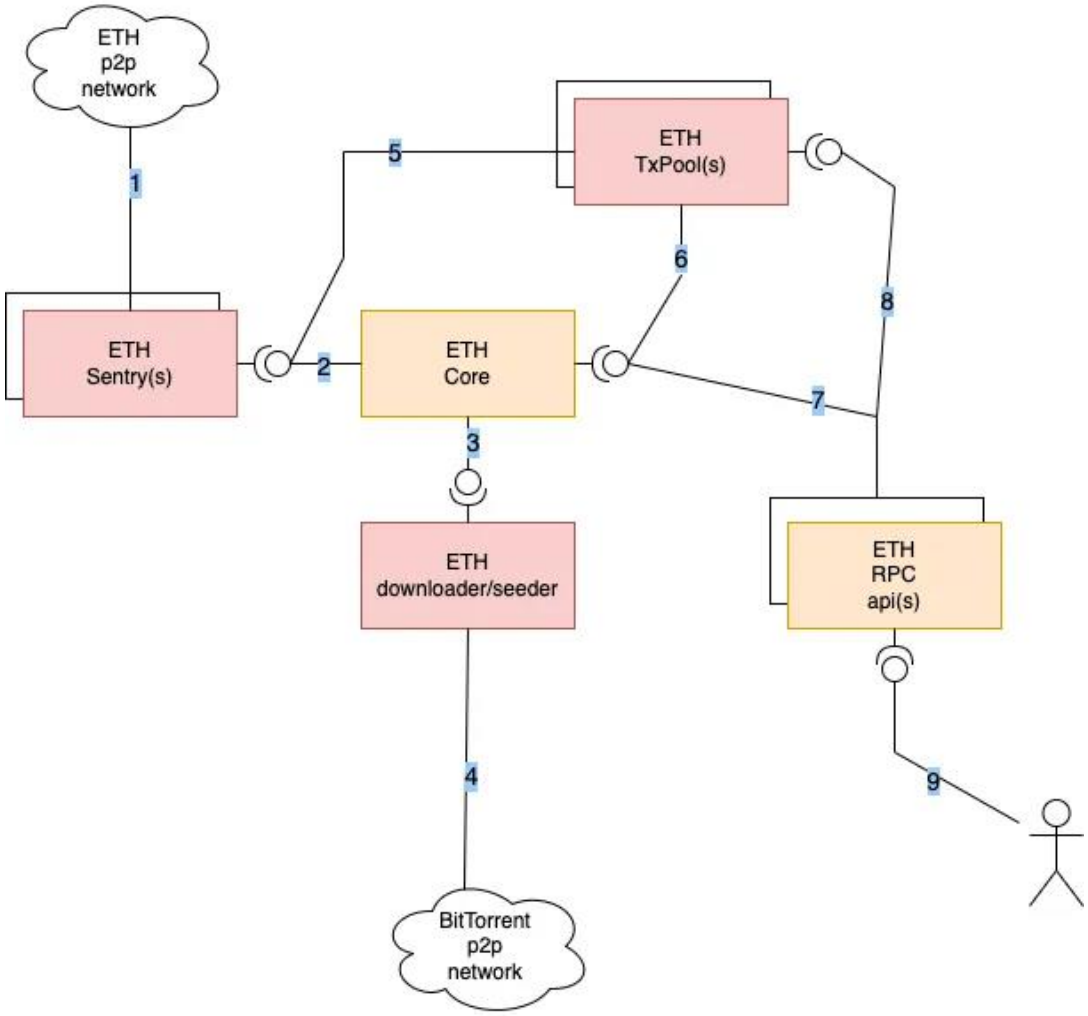

Рисунок 3.1 – Архітектура Erigon вузла [12]

Окреме значення має коректна конфігурація логів. Вузол генерує значну кількість інформації під час синхронізації, тому журнали виводяться у файл у каталозі /data/logs, що спрощує моніторинг, дозволяє оперативно відстежувати помилки та аналізувати динаміку роботи системи. Для уникнення надмірного споживання дискового простору налаштовується log rotation, який автоматично архівує або видаляє старі журнали.

Після цього формується systemd-сервіс, який забезпечує автоматичний запуск Erigon під час старту операційної системи, а також контроль його стану та автоматичне перезапускання у випадку збою. Такий підхід є критично важливим для підтримання безперервної роботи RPC-інфраструктури, оскільки вузол має залишатися доступним навіть після планових оновлень, перезавантажень системи або неочікуваних збоїв.

Під час першого запуску Erigon розпочинає процес синхронізації блокчейну Ethereum, який залежно від конфігурації інстансу, пропускної здатності диска й швидкості мережі може займати від кількох годин до декількох діб. На цьому етапі активно використовується багатопотоковість, а навантаження на локальний NVMe-диск сягає максимальних значень, що підтверджує важливість попередньо виконаної оптимізації дискової підсистеми. Завдяки використанню інстансу з локальним NVMe та достатніми ресурсами CPU/RAM синхронізація проходить стабільно й без деградації продуктивності.

Після завершення процесу синхронізації вузол повністю готовий до обробки RPC-запитів і може інтегруватися з іншими компонентами системи, такими як балансувальник навантаження, кешуючі шари, система логування або моніторинг (рис. 3.2). У такому стані сервер набуває ролі стабільного джерела даних для внутрішніх сервісів платіжної інфраструктури, що взаємодіють з блокчейном Ethereum.

Таким чином, встановлення й конфігурація Erigon складають критичний етап підготовки RPC-інфраструктури. Саме на цьому рівні визначаються продуктивність, швидкість синхронізації, стабільність обробки запитів та подальша масштабованість системи. Створене середовище повністю готове для роботи як вузол, що відповідає за централізований доступ до блокчейну в межах платіжного сервісу, та слугує основою для наступних етапів реалізації всієї архітектури.

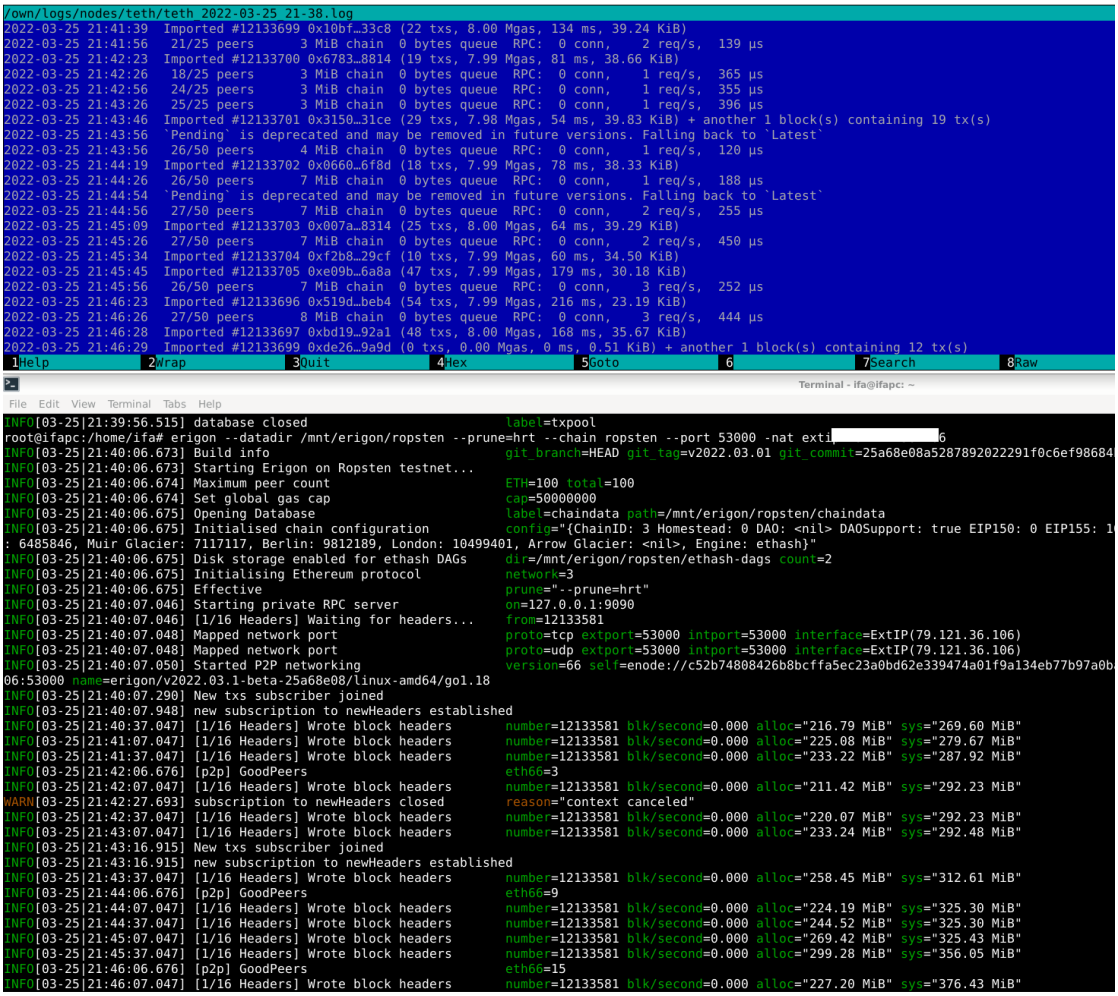

Рисунок 3.2 – Логи від Erigon

3.2 Забезпечення безпеки RPC-інфраструктури

Захист RPC-інфраструктури є критично важливою частиною загальної архітектури платіжної системи, оскільки будь-який неконтрольований доступ до RPC-інтерфейсу може створювати загрози неправомірних запитів, перевантаження вузла або ініціювання несанкціонованих транзакцій. У межах цього проєкту безпека реалізується на трьох рівнях: мережевому, доступу до RPC, а також на рівні захисту приватних ключів, що використовуються для підпису транзакцій. Такий підхід дозволяє ізолювати вузол, контролювати взаємодію внутрішніх сервісів та забезпечувати захищеність критичних операцій.

3.2.1 Мережевий захист

Мережевий рівень є першим захисним бар'єром, який контролює можливість звернення до RPC-інтерфейсу. Вузол розгортається всередині приватної мережі AWS VPC, і всі правила доступу регулюються Security Groups. Єдиний відкритий порт для зовнішнього світу – 30303, який необхідний для роботи Ethereum P2P-протоколу. Порти RPC (8545/8546) доступні винятково з внутрішніх IP-адрес (рис. 3.3).

```
INBOUND:
  TCP 22      → лише адміністративний IP
  TCP 8545    → 10.0.0.0/16 (приватна мережа)
  TCP 8546    → 10.0.0.0/16
  TCP/UDP 30303 → 0.0.0.0/0 (Ethereum P2P)

OUTBOUND:
  ALL → 0.0.0.0/0
```

Рисунок 3.3 Конфігурація Security Group

Доступ до RPC додатково контролюється IP-фільтрацією. Це захищає інфраструктуру навіть у межах приватної мережі: лише конкретні внутрішні сервіси можуть ініціювати виклики до вузла.

Для сервісів, що потребують шифрованого каналу, перед вузлом встановлюється зворотний проксі (Nginx або Traefik). Він виконує TLS-шифрування, приймає сертифікати від AWS ACM, фільтрує запити та слугує додатковим контрольним шаром (рис. 3.4).

```
server {
    listen 443 ssl;
    server_name rpc.internal.example.com;

    ssl_certificate     /etc/nginx/certs/cert.pem;
    ssl_certificate_key /etc/nginx/certs/key.pem;

    location / {
        allow 10.0.0.0/16;
        deny all;

        proxy_pass http://127.0.0.1:8545;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Рисунок 3.4 – Конфігурація Nginx як reverse proxy для RPC

Такі обмеження забезпечують те, що RPC може бути викликаний лише з перевірених джерел і виключно через зашифрований канал у тих випадках, коли це необхідно.

3.3 Порівняння власної RPC-інфраструктури Erigon з хмарними сервісами

Для визначення ефективності розробленої інфраструктури було проведено порівняльне дослідження роботи власного вузла на основі клієнта Erigon із наявними ринковими рішеннями, як локальними Ethereum-клієнтами (Geth, Nethermind), так і хмарними RPC-провайдерами (Infura, Alchemy, Ankr). Таке комплексне порівняння дає можливість об’єктивно оцінити продуктивність, надійність, гнучкість та економічну доцільність використання власної інфраструктури у платіжній системі.

Дослідження виконувалося у двох напрямках. Перше – аналіз архітектурних відмінностей між Erigon та іншими клієнтами Ethereum, що дозволило визначити сильні сторони обраного рішення на рівні зберігання даних, виконання транзакцій і структури бази даних. Друге – порівняння локального вузла з хмарними RPC-провайдерами, що дало можливість оцінити різницю у швидкодії, затримці, доступності, масштабованості та вартості експлуатації. Результати обох досліджень стали основою для обґрунтування архітектурного вибору та підтвердили доцільність побудови власної RPC-інфраструктури.

3.3.1 Архітектурні відмінності Erigon від Geth та Nethermind

У межах експериментального дослідження було здійснено детальне порівняння обраного клієнта Erigon з іншими популярними Ethereum-клієнтами Geth та Nethermind. Це дало можливість визначити архітектурні переваги Erigon та обґрунтувати його вибір як основи для побудови RPC-інфраструктури «запропонованої системи».

Ключова відмінність полягає у способі зберігання та обробки стану блокчейну. Geth використовує класичну модель Merkle-Patricia Trie, у якій кожен запит до стану вимагає реконструкції відповідної гілки дерева. Такий підхід створює значне навантаження на диск і збільшує латентність при обробці RPC-запитів. Nethermind частково оптимізує ці операції завдяки застосуванню альтернативних структур зберігання, проте також не позбавлений обмежень, притаманних trie-орієнтованим клієнтам.

Erigon натомість застосовує «площинну» (flat) модель бази даних, у якій стан зберігається у вигляді звичайних таблиць ключ–значення. Це дозволяє виконувати операції читання без необхідності обходу дерева, що зменшує кількість IO-операцій у кілька разів. Такий підхід забезпечує значно вищу швидкість обробки запитів, особливо під час виконання інтенсивних RPC-викликів (eth_getBalance, eth_call, eth_getStorageAt).

Важливою перевагою є багатоступенева pipeline-архітектура, у межах якої процес обробки блоків поділяється на етапи: завантаження даних, виконання транзакцій, хешування, pruning і формування статичних таблиць. Це дозволяє максимально ефективно використовувати багатопотокові CPU і забезпечує значно швидшу повну синхронізацію мережі.

Експеримент показав, що власний вузол на Erigon завершив повний sync Ethereum приблизно за п’ять годин, що у 3-4 рази швидше, ніж Geth, та на 40-60 % швидше, ніж Nethermind. У RPC-тестах середня затримка Erigon становила 1-3 мс, тоді як Geth демонстрував 5-8 мс, а Nethermind – 6-10 мс в аналогічних умовах.

Таким чином, Erigon забезпечує оптимальне співвідношення швидкості, стабільності та гнучкості та є найбільш ефективним рішенням для побудови власної інфраструктури, орієнтованої на високе навантаження та складну транзакційну логіку.

3.3.2 Порівняння Erigon із хмарними RPC-провайдерами

Розгорнута RPC-інфраструктура на основі клієнта Erigon була протестована та порівняна з ринковими хмарними провайдерами Infura, Alchemy

та Ankr. Метою порівняння було визначити ефективність власного підходу до доступу до блокчейн-даних порівняно з централізованими сервісами, що працюють за моделлю «інфраструктура як послуга».

Хмарні сервіси надають доступ до Ethereum-мережі через масштабовані дата-центри, забезпечуючи SLA на рівні 99,9 % і більше. Їхньою перевагою є швидкий старт, відсутність необхідності розгортати власне ПЗ та спрощена інтеграція. Проте вони накладають істотні обмеження, зокрема ліміти запитів, непрозорість внутрішніх алгоритмів, неможливість кастомізації, а також ризик підвищення вартості при зростанні навантаження.

Власна інфраструктура Erigon працює на виділеному обладнанні з локальним NVMe-сховищем, що забезпечує дуже високу швидкість читання стану блокчейну. Під час дослідження вузол завершив повну синхронізацію Ethereum за приблизно п’ять годин – у 3-4 рази швидше, ніж стандартний Geth. Локальні затримки запитів склали лише 1-3 мс, що значно випереджає Infura та Alchemy (близько 38-40 мс). Власна система також позбавлена штучних лімітів: пропускна здатність визначається лише характеристиками заліза.

Хмарні сервіси демонструють високу відмовостійкість завдяки георозподіленим дата-центрам і автоматичному перемиканню на резервні вузли. Натомість власний вузол у базовій конфігурації є потенційною точкою відмови (SPOF). Проте інтеграція кластеризації й конфігурація кількох вузлів з балансувальником повністю нівелює цей недолік, дозволяючи досягти майже такого самого рівня доступності, як у Infura чи Alchemy.

У результаті можна стверджувати, що власна RPC-інфраструктура Erigon є доцільним рішенням у випадках, коли потрібно досягти максимальної продуктивності, контролю над даними, прозорості алгоритмів, а також зменшити витрати при великих обсягах запитів. Хмарні сервіси лишаються зручним варіантом для швидкого старту, але вони обмежують розробника як технічно, так і економічно (табл. 3.1).

Таблиця 3.1 – Порівняння RPC-провайдерів

Метрика	Erigon (власний вузол)	Infura	Alchemy	Ankr
Час повної синхронізації	≈5 годин	-	-	-
Ліміти запитів	Немає	Є (плани + кредити)	Є	Є
Середня затримка RPC	1-3 мс (локально)	~38 мс	~40 мс	~35-45 мс
Пропускна здатність	Лише обмеженням заліза	Обмежено планом	Обмежено планом	Обмежено планом
SLA / Відмовостійкість	Залежить від архітектури	Дуже висока	Дуже висока	Висока
Доступ до low-level API	Повний	Частковий	Частковий	Частковий
Контроль над даними	Повний	Немає	Немає	Немає
Вартість при великому трафіку	Найнижча	Зростає	Зростає	Зростає

3.4 Порівняння власної з ринковими платіжними платформами

Окрім порівняння RPC-інфраструктур, було проведено окреме дослідження платіжних сервісів, що працюють з криптовалютними транзакціями. Розглядалися найбільш популярні представники ринку: CoinGate, Coinbase Commerce, BitPay та CoinPayments [6]. Порівняння виконувалося за технічними, інфраструктурними, бізнесовими та фінансовими параметрами.

Готові платіжні сервіси зазвичай надають мінімальний час інтеграції, автоматичне формування інвойсів, підтримку Webhook, а також можливість конвертації валют [6]. Проте вони накладають суворі обмеження на логіку прийому платежів, кількість підтримуваних монет, модель білінгу й можливість кастомізації. Більшість платформ не дозволяють працювати із сирими блокчейн-даними та повністю приховують низькорівневі транзакції. Це робить

неможливими унікальні сценарії моніторингу, оптимізації, AML-фільтрації та швидкого реагування на події в mempool.

На відміну від них, наша система працює на власній RPC-інфраструктурі та має прямий доступ до mempool, що дає можливість реагувати на транзакції за 1-3 секунди. Завдяки цьому сповіщення про оплату надходить практично миттєво, без залежності від стороннього провайдера. Комісія сервісу становить 0,3 %, і може зменшуватися до 0,2 % чи навіть 0,1 % залежно від обороту. Це значно нижче, ніж у CoinGate (1 %), Coinbase Commerce (1 %), BitPay (1-2 %) або CoinPayments (0,5 %) [12].

Система підтримує роботу з усіма EVM-мережами, Tron, Solana, Arbitrum та TON. Такого покриття немає у жодного з розглянутих сервісів. Додатково система дозволяє в режимі реального часу додавати нові токени, включно з щойно залістеними монетами. Жоден із конкурентів не надає подібної можливості без окремих юридичних і технічних процедур.

Система надає повний white-label, включно з кастомними інвойсами, брендингом, доменом та можливістю інтегрувати власні алгоритми моніторингу. Конкуренти здебільшого пропонують лише частковий white-label або взагалі його не підтримують.

Вивід коштів можливий у USD, EUR та UAH, використовуючи SWIFT або SEPA. Для бізнесів це дає універсальність, якої немає, наприклад, у Coinbase Commerce (не має фіатних виводів) [13].

Запроваджена AML/KYT-перевірка адрес перед прийомом платежу підвищує безпеку й зменшує ризики. KYC потрібен лише для високих оборотів, що спрощує роботу малого й середнього бізнесу, на відміну від обов’язкової повної ідентифікації в CoinGate або BitPay.

Таким чином, наша система поєднує функціональність потужної платіжної платформи, швидкість реагування локальної RPC-інфраструктури, відсутність лімітів, розширені можливості кастомізації та конкурентну модель ціноутворення. Це робить її найбільш гнучким і технологічно просунутим рішенням серед досліджених сервісів (табл. 3.2).

Таблиця 3.2 – Порівняння платіжних систем

Метрика	Запропонована система	CoinGate	Coinbase Commerce	BitPay	CoinPayments
Підтримувані блокчейни	EVM, Tron, Solana, Arbitrum, TON	BTC, ETH, LTC, USDT	До 10 монет	~100 активів	2300+ токенів
Доступ до mempool	Так	Ні	Ні	Ні	Ні
Швидкість підтвердження	1-3 секунди	15-60 сек	15-60 сек	10-60 сек	10-60 сек
Депозитні адреси	Необмежено	Обмежено	Обмежено	Обмежено	Обмежено
Комісія	0.3 %-0.1 %	1 %	1 %	1-2 %	0,5 %
Конвертація	Кросчейн без націнки	Є (з націнкою)	Нема	Є	Є
Вивід у фіат	USD, EUR, UAH (SWIFT/SEPA)	EUR (SEPA)	Нема	EUR/USD	EUR/USD
White-label	Повний	Обмежений	Нема	Нема	Частковий
AML/KYT	Так	Так	Так	Так	Частково
KYC	За потребою	Обов’язковий	Обов’язковий	Обов’язковий	Обов’язковий
Кастомні токени	Так, live-додавання	Нема	Нема	Нема	Обмежено
API	Повний REST API	Так	Так	Так	Так
Ліміти	Немає	Є	Є	Є	Є

Після аналізу видно, що система не лише перевершує конкурентів за функціональністю, а й залишається значно гнучкішою в інтеграції. Її архітектура дозволяє швидко адаптуватися до специфічних бізнес-вимог і масштабувати окремі модулі без залежності від сторонніх провайдерів. Це підсилює надійність та довгострокову ефективність рішення.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було спроєктовано, розроблено та досліджено повноцінну платіжну систему на основі блокчейн-технологій, що включає власну RPC-інфраструктуру, високопродуктивний Ethereum-вузол та комплексний інструментарій для моніторингу й обробки транзакцій у реальному часі. Робота продемонструвала технічну та практичну доцільність застосування локальної RPC-архітектури замість сторонніх хмарних рішень у контексті створення швидких, надійних та масштабованих фінансових сервісів.

У процесі дослідження були проаналізовані теоретичні засади роботи децентралізованих блокчейн-мереж, включно з принципами формування та валідації транзакцій, механізмами консенсусу, структурою блокчейну та роллю RPC-інтерфейсів у побудові сервісів прикладного рівня. Це дозволило сформувати технічні вимоги до системи та визначити оптимальні підходи до обробки даних і організації високопродуктивної інфраструктури.

У ході реалізації було розгорнуто власний Ethereum-вузол на основі клієнта Erigon у середовищі AWS із використанням серверів з NVMe-накопичувачами, що забезпечило мінімальну латентність і високу швидкість синхронізації. Було впроваджено механізми мережевої ізоляції, захищеного доступу, контроль пропускної здатності, а також інструменти моніторингу для забезпечення стабільності роботи системи. Розроблена асинхронна система сканування блоків та транзакцій реалізує багатопотокову обробку, гарантує точність визначення вхідних і вихідних операцій, забезпечує відновлення після збоїв та підтримує інтеграцію з внутрішніми компонентами платіжної платформи.

Експериментальне порівняння власного вузла Erigon із хмарними RPC-провайдерами (Infura, Alchemy, Ankr) підтвердило, що локальна інфраструктура забезпечує кращу продуктивність, повний контроль над даними та можливість реалізації нетипових бізнес-процесів, тоді як сторонні сервіси

накладають обмеження на швидкість, обсяг запитів та доступ до низькорівневих даних. Дослідження платіжних систем (CoinGate, Coinbase Commerce, BitPay, CoinPayments) показало, що готові сервіси є зручними на етапі базової інтеграції, проте не дозволяють створювати власну логіку обробки платежів, не працюють з mempool у режимі реального часу, мають обмеження щодо кількості підтримуваних блокчейнів і токенів та використовують дорожчі моделі ціноутворення.

Розроблена система відрізняється від ринкових аналогів тим, що поєднує високу швидкість реакції (1-3 секунди), доступ до mempool, можливість live-додавання кастомних токенів, широку підтримку блокчейнів (EVM-мережі, Tron, Solana, Arbitrum, TON), гнучку white-label модель та низьку комісію (0,3 % з можливістю зниження). Завдяки цьому вона перевершує наявні рішення за показниками масштабованості, кастомізації, швидкодії та вартості.

Разом із тим у ході роботи було виявлено й низку потенційних обмежень, серед яких – залежність від одного вузла, потреба у складному DevOps-супроводі та необхідність побудови системи відмовостійкості. Для мінімізації цих ризиків запропоновано низку напрямів подальшого вдосконалення: кластеризація вузлів, автоматизація розгортання середовища, впровадження балансування навантаження, резервування даних та розширення системи моніторингу.

Підсумовуючи, робота підтвердила, що створення власної RPC-інфраструктури на базі Erigon та інтеграція її з платіжним модулем відкриває можливість побудови незалежних, високошвидкісних і гнучких криптовалютних сервісів нового покоління. Запропоноване рішення є технологічно перспективним, придатним до масштабування й може бути основою для подальших досліджень та комерційного використання у сфері фінансових технологій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Antonopoulos A., Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. 2023. URL: https://ethereum.github.io/yellowpaper/paper.pdf (дата звернення: 29.10.2025).

2. Alchemy Documentation. URL: https://docs.alchemy.com (дата звернення: 29.10.2025).

3. Amazon CloudHSM Documentation. URL: https://docs.aws.amazon.com /cloudhsm (дата звернення: 30.10.2025).

4. Ankr RPC Services. URL: https://ankr.com (дата звернення: 29.10.2025).

5. Bitcoin: A Peer-to-Peer Electronic Cash System. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 10.10.2025).

6. BitPay Developer Docs. URL: https://bitpay.com (дата звернення: 02.11.2025).

7. Chainstack RPC Documentation. URL: https://docs.chainstack.com (дата звернення: 29.10.2025).

8. Cloudflare WAF Documentation. URL: https://developers. cloudflare.com/waf (дата звернення: 30.10.2025).

9. Coinbase Commerce Documentation. URL: https://commerce.coinbase.com (дата звернення: 02.11.2025).

10. CoinCloud ATM Network. URL: https://coin.cloud (дата звернення: 02.11.2025).

11. CoinGate Overview. URL: https://coingate.com (дата звернення: 02.11.2025).

12. CoinPayments Merchant Tools. URL: https://coinpayments.net (дата звернення: 02.11.2025).

13. Erigon Documentation. GitHub. URL: https://github.com/ledgerwatch/erigon (дата звернення: 25.10.2025).

14. Ethereum Clients Documentation (Geth / OpenEthereum). URL: https://geth.ethereum.org/docs (дата звернення: 25.10.2025).

15. Ethereum JSON-RPC API Documentation. URL: https://eth.wiki/json-rpc/API (дата звернення: 05.11.2025).

16. Ethereum: A Secure Decentralised Generalised Transaction Ledger. URL: https://ethereum.org/en/whitepaper/ (дата звернення: 12.10.2025).

17. Gnosis Safe – Smart Contract Wallets. URL: https://docs.safe.global (дата звернення: 03.11.2025).

18. Hyperledger Fabric Documentation. URL: https://hyperledger-fabric.readthedocs.io (дата звернення: 15.10.2025).

19. Infura Ethereum API Documentation. URL: https://docs.infura.io (дата звернення: 29.10.2025).

20. Ledger Developer Portal. URL: https://developers.ledger.com (дата звернення: 03.11.2025).

21. Mastering Bitcoin: Programming the Open Blockchain. URL: https://github.com/bitcoinbook/bitcoinbook (дата звернення: 15.10.2025).

22. OWASP Top 10 for Blockchain. URL: https://owasp.org/www-project-blockchain-top-10 (дата звернення: 04.11.2025).

23. Quorum Privacy and Authorization. URL: https://docs.goquorum.consensys.net (дата звернення: 15.10.2025).

24. Tendermint Core Documentation. URL: https://docs.tendermint.com (дата звернення: 15.10.2025).

25. Web3.py Documentation. URL: https://web3py.readthedocs.io (дата звернення: 06.11.2025).

26. Мотрій Д., Бортник К. Методологічні засади побудови шардованої RPC-інфраструктури для високонавантажених платіжних систем у EVM-сумісних блокчейнах. Науковий альманах. Харків: СГ НТМ «Новий курс», 2025. ISBN 978-617-7886-81-4. DOI: 10.61718/cros2025. С. 213-214.