МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Використання LayerZero та Layer 2 рішень в розробці застосунків, що працюють з блокчейном**

Виконав: студент 2 курсу, групи 8.1214-іпз-2
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)

освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)

Е. А. Якимцов
(ініціали та прізвище)

Керівник доцент А.І. Безверхий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Рецензент директор ТОВ “Дісітел”
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя

2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти другий (магістерський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)

Освітня програма Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Тетяна КРИТСЬКА
“ 01 ” квітня 2025 року

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

Якимцову Едуарду Анатолійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи: Використання LayerZero та Layer 2 рішень в розробці застосунків, що працюють з блокчейном
керівник роботи доцент кафедри електроніки, інформаційних систем та програмного забезпечення, Безверхий Анатолій Ігорович
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від 28.03.2025р. № 361-е
2. Строк подання студентом кваліфікаційної роботи 01.12.2025
3. Вихідні дані магістерської роботи
   - комплект нормативних документів;
   - технічне завдання до роботи.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   - огляд та збір літератури стосовно теми кваліфікаційної роботи;
   - створення програмного продукту та його опис;
   - перелік вимог для роботи програми;
   - дослідження проблеми та розробка висновків та пропозицій
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
   16 слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата
		Завдання прийняв

7. Дата видачі завдання 01.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	01.04.2025 р.	Виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	15.04.2025 р.	Виконано
3	Огляд та збір літератури стосовно теми кваліфікаційної роботи	01.05.2025 р.	Виконано
4	Опрацювання функціональних вимог до системи	15.05.2025 р.	Виконано
5	Проектування архітектури застосунку	01.10.2025 р	Виконано
6	Програмна реалізація застосунку та його апробація	01.11.2025 р	Виконано
7	Оформлення звіту	01.12.2025 р.	Виконано

Студент _____ Якимцов Е.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Безверхий А.І.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок – 77

Рисунків – 17

Таблиць – 1

Джерел – 27

Якимцов Е. А. Використання LayerZero та Layer 2 рішень в розробці застосунків, що працюють з блокчейном: кваліфікаційна робота магістра спеціальності 121 “Інженерія програмного забезпечення” / наук. керівник А. І. Безверхий. Запоріжжя : ЗНУ, 2025. 77 с.

Мета дослідження полягає у аналізі роботи смарт-контрактів розроблених за допомогою LayerZero, для взаємодії між блокчейнами, та вивченні механізмів ефективного обміну даними між Layer 1 та Layer 2 блокчейнами. В роботі особлива увага приділяється розробці застосунків які працюють зі кількома блокчейнами одночасно.

У процесі дослідження була розглянута проблема передачі даних між блокчейнами, а також проблеми масштабованості блокчейнів та високої вартості транзакцій. Як результат, було створено смарт-контракт за допомогою технології LayerZero, використано інтеграцію з Enso Shortcuts API, для побудови оптимальних шляхів передачі ліквідності між блокчейнами, та розроблено мобільний застосунок. Застосунок є прикладом використання вищевказаних технологій для міжмережевої взаємодії, зменшення витрат та збільшення швидкості обробки транзакцій.

Ключові слова: блокчейн (чейн), консенсус, Proof of Work (PoW), Proof of Stake (PoS), LayerZero, Layer 1 (L1), Layer 2 (L2), Zero-knowledge Rollups, Optimistic Rollups, Sidechain, State Channels, Decentralized Application, токен, транзакція, смарт-сонтракт.

ABSTRACT

Pages – 77

Figures – 17

Tables – 1

Sources – 27

Yakymtsov E. A. Utilizing LayerZero and Layer 2 solutions in blockchain-based applications development: master's thesis in the specialty 121 “Software Engineering” / scientific supervisor: A. I. Bezverkhyi. Zaporizhzhia: ZNU, 2025. 77 pages.

The aim of this study is to analyze the performance of smart contracts developed using LayerZero for cross-chain interoperability and to examine mechanisms for efficient data exchange between Layer 1 and Layer 2 blockchains. Particular attention is paid to the development of applications capable of operating across multiple blockchains simultaneously.

The research addresses the challenges of cross-chain data transfer, as well as issues related to blockchain scalability and high transaction costs. As a result, a smart contract was created using LayerZero technology, and integration with Enso Shortcuts API was implemented to establish optimal liquidity transfer routes between blockchains. Additionally, a mobile application was developed to serve as a practical example of utilizing these technologies for cross-chain interoperability, aiming to reduce costs and increase transaction processing speed.

Keywords: blockchain (chain), consensus, Proof of Work (PoW), Proof of Stake (PoS), LayerZero, Layer 1 (L1), Layer 2 (L2), Zero-knowledge Rollups, Optimistic Rollups, Sidechain, State Channels, Decentralized Applications, token, transaction, smart contract.

ЗМІСТ

ВСТУП 8
1 ДОСЛІДЖЕННЯ ОБМЕЖЕНЬ СУЧАСНИХ БЛОКЧЕЙНІВ 15
1.1 Огляд проблем які притаманні Layer 1 блокчейнам 15
1.1.1 Механізм консенсусу 15
1.1.2 Міжмережева взаємодія 17
1.2 Підсумки розділу 1 18
2 ДОСЛІДЖЕННЯ ТЕХНІК ПОДОЛАННЯ ОБМЕЖЕНЬ БЛОКЧЕЙНІВ 20
2.1 Огляд Layer 2 рішень 20
2.1.1 State Channels 20
2.1.2 Sidechains 21
2.1.3 Rollups 23
2.2 Протокол LayerZero 25
2.3 Міжмережева взаємодія через Enso Build 27
2.4 Підсумки розділу 2 29
3 РОЗРОБКА ДЕЦЕНТРАЛІЗОВАНОГО ОМНІЧЕЙН ЗАСТОСУНКУ 33
3.1 Бізнес-обґрунтування та вимоги 33
3.2 Вимоги до застосунку 33
3.2.1 Функціональні вимоги 33
3.2.2 Нефункціональні вимоги 35
3.3 Архітектура застосунку 36
3.4 Розробка бази даних 39
3.5 Розробка мобільної частини застосунку 40
3.5.1 Налагодження локального середовища 42
3.5.2 Особливості програмної реалізації 43
3.6 Розробка бекенд частини застосунку 47
3.6.1 Налагодження локального середовища 48

3.6.2 Особливості програмної реалізації 49
3.7 Підсумки розділу 3 52
4 АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ ДЕЦЕНТРАЛІЗОВАНОГО ЗАСТОСУНКУ 54
4.1 Аналіз міжмережевого трансферу ліквідності 54
4.2 Аналіз інтеграції Enso Shortcuts API 63
4.3 Аналіз продуктивності Layer 2 рішень у порівнянні з Layer 1 67
4.4 Економічний аналіз розробленого застосунку 71
4.5 Підсумки розділу 4 72
ВИСНОВКИ 74
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 75

ВСТУП

Актуальність теми

Блокчейн – це технологія, яка дозволяє зберігати інформацію в розподіленому та незмінному вигляді в мережі. Блокчейн складається з блоків даних. Кожен блок може бути пустим, або включати в себе транзакції. Кожен наступний блок додається до попереднього і містить посилання на нього, що робить неможливим змінити чи підробити інформацію у вже створених блоках.

Технологія набирає популярності, тож багато розробників використовує її у своїх застосунках. Вона дозволяє створювати децентралізовані застосунки і відкриває можливості для розробки надійних, прозорих рішень. Попри свої переваги, блокчейн має також свої обмеження. Основними проблемами є низька масштабованість (транзакції займають багато часу), висока вартість транзакцій, а також складність інтеграції між різними блокчейн-мережами. У зв’язку з цим з’являються рішення, що спрямовані на вирішення цих проблем, такі як Layer 2 блокчейни, кросс-чейн мости, та протокол LayerZero.

Layer 2 блокчейни, мають механізми масштабування, які зменшують навантаження на основний блокчейн (Layer 1), та дозволяють виконувати транзакції швидше, дешевше та з більшою пропускною здатністю. LayerZero, у свою чергу, є протоколом, який дозволяє здійснювати передачу даних між блокчейнами. Це важливо в умовах, коли застосунок, який використовує смарт-контракт, повинен гарантувати безперервну взаємодію та обмін інформацією між кількома блокчейнами одночасно (наприклад переказ токенів з однієї мережі до іншої). Також цей протокол дозволяє нам не будувати кроссчейн рішення вручну. LayerZero реалізує концепцію “мультичейновості”, відкриваючи новий спосіб створення контрактів які працюють в обраних розробником мережах. Для оптимальної міжмережевої роботи також важливо знаходити найвигідніший шлях передачі даних і в цьому на допомогу приходить Enso Shortcuts API.

Виходячи з вище сказаного, актуальною є розкобка застосунку який працює з блокчейнами і першого, і другого рівня, а також дозволяє нам обмінюватися ліквідністю між блокчейнами без додаткових витрат на кастомні рішення.

Мета дослідження

Мета дослідження полягає у тому, щоб зрозуміти чи доцільне використання LayerZero смарт-контрактів та Enso Shortcuts API для вирішення проблем блокчейн-технологій в розробці децентралізованих мультичейн застосунків. Під час дослідження проаналізовано проблему обмеженої масштабованості та високих комісій у блокчейнах першого рівня (L1), а також розглянуто складнощі які виникають під час взаємодії між кількома блокчейнами. Як результат, створено LayerZero контракт, використано інтеграцію з Enso Shortcuts API, для побудови оптимальних шляхів передачі ліквідності між блокчейнами, а також розроблено програмний застосунок, що дозволяє користувачам взаємодіяти з кількома блокчейнами різного рівня, забезпечуючи цілісність даних та без необхідності реалізації власних механізмів взаємодії.

Об’єкт дослідження

Об’єктом дослідження є процес взаємодії між блокчейнами різного рівня.

Предмет дослідження

Предметом дослідження є використання технологій LayerZero, Enso Shortcuts API, та взаємодія між блокчейнами.

Методи дослідження

Для проведення даної дослідницької роботи було використано наступні методи:

1. Аналітичний метод. Дослідження існуючих рішень для виявлення їх переваг та недоліків, визначення оптимальних підходів для реалізації міжмережевої взаємодії.

2. Порівняльний аналіз. Оцінка переваг та недоліків використання смарт-контракту на основі LayerZero замість частково централізованих та інших кроссчейн рішень.

3. Прототипування. Розробка MVP для перевірки концепції перед реалізацією застосунку.

4. Експеримент. Розробка та тестування застосунку, аналіз його продуктивності, швидкості, та вартості виконання операцій.

5. Економічне моделювання. Оцінка вартості виконання транзакцій виконаних за допомогою LayerZero та Enso Shortcuts API.

Наукова новизна одержаних результатів

Наукова новизна дослідження полягає у комплексному аналізі використання протоколу LayerZero, та Enso Shortcuts API для забезпечення міжмережевої взаємодії в блокчейн-екосистемах, з акцентом на наступні аспекти:

1. Аналіз продуктивності Layer 2 рішень у порівнянні з Layer 1.

2. Дослідження прямої інтеграції LayerZero контрактів у мультичейнові застосунки для забезпечення прямої взаємодії між блокчейнами першого та другого рівня без необхідності створення централізованих або частково децентралізованих мостів.

3. Дослідження ефективності інтеграції з Enso Shortcuts API для реалізації обміну токенів в мережі та між мережами.

4. Розробка рекомендацій щодо використання LayerZero та Enso Shortcuts API у децентралізованих застосунках, які враховують фактори ефективності та продуктивності.

5. Дослідження можливості зниження комісійних витрат та оптимізації ресурсів для розробників і користувачів блокчейн-застосунків.

Практичне значення одержаних результатів

Практичне значення роботи полягає в оцінці ефективності LayerZero та Enso Shortcuts для міжмережевої взаємодії. Проведений аналіз визначає підходи до інтеграції цього протоколу та API, у сфері децентралізованих фінансів. Розроблений застосунок демонструє використання LayerZero для переміщення власного токену між блокчейнами різного рівня, та використання Enso Shortcuts API у міжмережевій взаємодії, що дозволяє знизити витрати та забезпечити стабільність роботи без створення кастомних мостів. Результати дослідження включають економічну оцінку використання LayerZero та Enso Build, що дозволяє визначити його доцільність у порівнянні з традиційними підходами. Це дослідження відкриває нові можливості для розробки надійних мультичейнових застосунків в короткі строки, з акцентом на самому продукті та швидкості.

Апробація одержаних результатів

Результати дослідження представлені на II міжнародній науково-практичній конференції Інженерного навчально-наукового інституту ім. Ю.М. Потебні (15-16 травня 2025 року, м. Запоріжжя) [27].

Глосарій

Блокчейн (чейн) – розподілена база даних, що складається з ланцюга блоків, кожен з яких містить інформацію про зміни стану системи, та пов’язаний з попереднім. Використовується для зберігання та перевірки даних без потреби в центральному органі управління.

Консенсус – механізм узгодження стану блокчейну між учасниками мережі, який дозволяє досягти єдиного правдивого стану системи без центрального управління.

Proof of Work (PoW) – модель консенсусу, в якій майнери розв'язують криптографічні задачі, витрачаючи обчислювальні ресурси, щоб підтвердити транзакції та додати новий блок у блокчейн (наприклад, Bitcoin).

Proof of Stake (PoS) – модель консенсусу, де учасники блокчейну блокують (стейкають) свої токени як гарантійний внесок, щоб отримати право створювати нові блоки та отримувати винагороди, що зменшує потребу в складних обчисленнях (наприклад, Ethereum 2.0).

Майнер – учасник мережі, який використовує обчислювальні ресурси для перевірки та додавання нових блоків у PoW-мережах.

Валідатор – учасник блокчейн-мережі, який підтверджує транзакції та створює нові блоки в PoS-мережах.

Блок – одиниця даних, що може містити набір підтверджених транзакцій, хеш попереднього блоку, часову мітку та інші технічні параметри. Блоки зв'язуються в незмінний ланцюг, що забезпечує цілісність даних у мережі.

Smart Contract – програмний код, що автоматично виконує умови угоди на блокчейні.

Транзакція – операція передачі даних або цифрових активів між учасниками блокчейну. Транзакція містить інформацію про відправника, отримувача, hash-суму та цифровий підпис для підтвердження автентичності. Після верифікації транзакція додається до блоку.

Токен – цифровий актив, що існує на блокчейні та може виконувати різні функції: бути засобом обміну, правом на управління, частиною DeFi-системи або підтвердженням власності. Токени зазвичай працюють на основі смарт-контрактів та можуть відповідати стандартам, таким як ERC20.

ERC20 – це стандарт, який використовується для створення та випуску токенів на блокчейнах які сумісні з Ethereum. Він визначає набір функцій, таких як передача, баланс та схвалення, яким повинні слідувати всі токени, щоб вони могли взаємодіяти з гаманцями, біржами та іншими контрактами.

Coin (монета) – цифрова валюта, що має власний блокчейн і використовується як засіб платежу або збереження вартості (наприклад, Bitcoin (BTC), Ethereum (ETH), Solana (SOL)). Коіни відрізняються від токенів тим, що функціонують у своїй незалежній мережі.

Layer 1 (L1) – головний рівень побудови блокчейну як системи, який виконує всі транзакції та забезпечує безпеку мережі.

Layer 2 (L2) – технології, що працюють поверх Layer 1, для підвищення масштабованості та зменшення вартості транзакцій.

Zero-knowledge Rollups (zk-Rollups) – тип Layer 2-рішення, що агрегує велику кількість транзакцій в один пакет, використовуючи Zero-Knowledge Proofs (докази з нульовим розголошенням) для перевірки їхньої коректності без розкриття деталей.

Optimistic Rollups – тип Layer 2-рішення, що агрегує транзакції та відправляє їх у основний блокчейн, використовуючи механізм "оптимістичних" доказів, припускаючи їхню правильність, поки не буде подано заперечення.

State Channels – механізм, що дозволяє учасникам проводити транзакції поза основним блокчейном, записуючи лише кінцевий результат у мережу. Використовується для швидких і дешевих мікротранзакцій.

LayerZero – протокол взаємодії між блокчейнами, який дозволяє різним мережам обмінюватися даними без посередників, забезпечуючи високу безпеку та децентралізацію.

Decentralized Applications (dApps) – це застосунки, що працюють в мережі блокчейн. Вони можуть бути у вигляді смарт-контрактів, ігор, фінансових сервісів тощо.

Omnichain Applications (Омнічейн застосунки) – це децентралізовані застосунки, які можуть працювати та взаємодіяти з даними одразу кількох блокчейн-мереж.

Enso Build – платформа, яка спрощує розробку децентралізованих додатків. Вона позиціонує себе як “blockchain shortcuts” та надає API, що дозволяють розробникам зосередитися на створенні продукту, а не на складнощах, пов’язаних з інтеграцією різних блокчейнів та смарт-контрактів.

1 ДОСЛІДЖЕННЯ ОБМЕЖЕНЬ СУЧАСНИХ БЛОКЧЕЙНІВ

1.1 Огляд проблем які притаманні Layer 1 блокчейнам

Всі великі блокчейни, такі як Ethereum, Binance Smart Chain (BSC), Bitcoin, Cardano, Tron та інші, стикаються з такими проблемами як масштабованість, вартість транзакцій та комунікація. Ці обмеження викликані їх механізмами роботи які буде висвітлено нижче.

1.1.1 Механізм консенсусу

Механізм консенсусу забезпечує консистентність даних в мережі. Всі учасники мережі погоджуються з єдиним, незмінним ланцюгом транзакцій без необхідності в центральному органі управління. Він гарантує надійність і довіру до блокчейн-системи, а також захищає від різних видів шахрайства, наприклад, від подвійного витрачання коштів, навіть якщо учасники мережі не довіряють один одному.

Основне завдання механізму консенсусу полягає в узгодженні поточного стану блокчейну, незалежно від того, що кожен учасник може мати лише часткову інформацію про мережу. Механізм гарантує, що кожен учасник отримує однакову інформацію про транзакції та нові блоки в мережі, що дозволяє створювати та зберігати стабільну, незмінну послідовність записів того що відбувалося в системі [1].

Proof of Work (PoW). Цей механізм використовується в системах, таких як Bitcoin, для забезпечення стійкості до атак типу Sybil, у яких зловмисник створює велику кількість фальшивих ідентичностей або вузлів у децентралізованій мережі для отримання непропорційного контролю над системою через витратність майнінгу. PoW ґрунтується на тому, що вузол вираховує хеш значення блоку і коли блок публікується до інших вузлів, то вони мають підтвердити це значення

і лише тоді опублікувати цей блок в себе [2]. Основними недоліками є низька пропускна здатність і високе енергоспоживання, яке стимулює створення великих майнінгових пулів, що підвищує ризики централізації.

Proof of Stake (PoS). Валідатори (учасники, які мають певну кількість монет які блокуються системою і гарантують їх чесність роботи) обираються для створення нових блоків пропорційно їхній частці в мережі [3]. Цей механізм є енергоефективним і має кращу масштабованість порівняно з PoW, однак він залежить від фінансової централізації, оскільки багатші учасники мають більше голосів у консенсусі. Він використовується, зокрема, в Cardano та Ethereum 2.0 та має високу продуктивність (1000+ TPS в залежності від дизайну).

Proof of Authority (PoA). Право на створення блоків належить фіксованому набору вузлів, які мають довіру від мережі [3]. Це забезпечує високу швидкість – до 10000 TPS і низькі витрати, але знижує децентралізацію та робить систему вразливою до компрометації довірених вузлів.

Byzantine Fault Tolerance (BFT). Механізм орієнтований на стійкість до збоїв, навіть якщо частина вузлів поводиться зловмисно. Протоколи, як-от PBFT (Practical Byzantine Fault Tolerance), забезпечують високу пропускну здатність і гарантовану фіналізацію транзакцій. Але такі протоколи погано масштабуються через квадратичну складність обміну повідомленнями між вузлами [3].

Hybrid Consensus. Ці механізми поєднують елементи PoW та BFT для досягнення балансу між безпекою, ефективністю та децентралізацією. Наприклад, гібридні рішення можуть використовувати PoW для вибору вузлів, а BFT – для фіналізації блоків [3].

Механізм консенсусу також є основним вузьким місцем в роботі мережі. Наприклад Ethereum, як одна з найбільших платформ для децентралізованих додатків (dApps) має пропускну здатність яка становить близько 13-15 транзакцій за секунду (TPS), а Bitcoin має пропускну здатність до 7 транзакцій за секунду та високу енергоємність, викликану використанням механізму Proof of Work та

тривалу генерацію блоків (в середньому 10 хвилин). Для порівняння, платіжні системи, такі як Visa, можуть обробляти понад 65 000 TPS [3].

1.1.2 Міжмережева взаємодія

Нові блокчейни з’являються кожен день, тому постає питання взаємодії між ними. Блокчейни ізольовані і прив'язують користувачів до себе, та часто виникає необхідність переміщувати свої активи між чейнами. Ізольованість унеможливлює прямий обмін даними між блокчейнами, що призводить до розділення ліквідності, та складнощів у розробці багатоланцюгових додатків.

Однією з головних причин ізольованості є відсутність нативної міжланцюгової комунікації. Блокчейн був задуманий як система з безпечною та прозорою передачею даних у межах одного ланцюга. Кожна транзакція є підтвердженою та незмінною завдяки консенсусному механізму, що забезпечує цілісність даних. Проте цей підхід був від початку розроблений для внутрішньоланцюгових операцій. Як наслідок, блокчейни не були побудовані з урахуванням можливості взаємодії між собою, що призводить до фрагментації.

На прикладі ринку децентралізованих фінансів (DeFi) ця проблема стає особливо значущою. Кожен блокчейн має свій пул ліквідності та додатки, що створюють обмежений простір для операцій. Користувачі змушені працювати з окремими екосистемами, що обмежує можливості для отримання прибутку чи оптимізації операцій. Наприклад, активи, які перебувають у блокчейні Ethereum, не можуть бути використані в мережі Binance Smart Chain або Solana без додаткових інструментів, таких як мости чи централізовані біржі. Це створює додаткові витрати, затримки та складності.

Щоб подолати цю ізольованість, користувачі зазвичай звертаються до централізованих бірж. Централізовані біржі, такі як Binance, дозволяють обмінювати активи між різними блокчейнами, але вимагають довіри до третьої сторони. Це суперечить фундаментальним принципам блокчейн-технологій, які

повинні забезпечувати бездовірчість і децентралізацію. Такий підхід додає зайві кроки до процесу обміну: спочатку користувач має конвертувати свої активи в проміжний токен, а потім обміняти цей токен на бажаний актив на іншому ланцюзі. У випадку централізованих рішень користувачі ризикують втратити свої активи через можливі збої, зловживання або хакерські атаки. Крім того, централізовані біржі часто мають значні комісії, що робить такі рішення дорогими та менш ефективними. Ті самі проблеми мають і рішення розроблені шляхом створення двох контрактів в різних мережах які керуються на рівні якогось бекенд-сервісу. Якщо додаток з якоїсь причини дасть збій то токени будуть заблоковані в одній мережі і не будуть випущені в іншій.

Децентралізовані альтернативи, такі як крос-ланцюгові мости (Cross-Chain Bridge), створені вирішити проблему ізольованості, але також мають недоліки в плані безпеки і тому є мішенню для хакерських атак, які у 2022 році завдали збитків на суму від 1.5 до 2 мільярдів доларів США [5]. Для передачі активів ці рішення використовують контракти на обох мережах і токени мають бути заблоковані, або спалені на вихідній мережі і випущені, або розблоковані на вхідній [6]. Це ускладнює процес і створює додаткові витрати та затримки. Також, якщо токени будуть спалені в одній мережі і повідомляння не надійде до іншої, то вони будуть просто втрачені.

1.2 Підсумки розділу 1

Основною проблемою швидкості роботи блокчейну є його механізм консенсусу. Хоч він і забезпечує безпеку та незмінність, але часто стає вузьким місцем для пропускної здатності.

Міжмережева взаємодія є другою основною проблемою, оскільки блокчейни за своєю природою ізольовані. Ця ізольованість призводить до фрагментації ліквідності, бо треба робити окремі смарт-контракти в кожній

мережі і ці контракти нічого не знають одне про одного. Все це ускладнює розробку додатків які працюють з кількома мережами одночасно.

Централізовані рішення суперечать основним принципам блокчейну, вимагаючи довіри до третьої сторони та створюючи ризики безпеки.

Децентралізовані мости, хоч і покращують ситуацію, та часто додають зайві кроки та витрати через використання проміжних токенів, ускладнюючи процес та збільшуючи вартість для кінцевого користувача. Звичайно в деяких ситуаціях не можна обійтись без їх використання, та питання їх надійності залишається завжди відкритим.

2 ДОСЛІДЖЕННЯ ТЕХНІК ПОДОЛАННЯ ОБМЕЖЕНЬ БЛОКЧЕЙНІВ

2.1 Огляд Layer 2 рішень

Оскільки основною проблемою блокчейн-технології є масштабованість, то всі рішення цієї проблеми повинні враховувати так звану “трілему блокчейнів” [4]. Її суть полягає в тому, що покращення масштабованості блокчейна негативно впливає або на його безпеку, або на децентралізацію, або на обидва аспекти блокчейну одночасно. Необхідно знайти правильний компроміс для забезпечення надійної і безпечної роботи блокчейну, збалансувавши ці три складові.

Найпоширенішим підходом для досягнення масштабованості є так звані “Layer 2” рішення. Layer 2 є вторинним протоколом, який будується поверх існуючого блокчейна. Основна ідея полягає у створенні додаткового рівня, який обробляє транзакції поза основним блокчейном (тобто поза Layer 1) і передавати на Layer 1 лише “підсумок” цих транзакцій [7]. Це зменшує навантаження на основний ланцюг, дозволяючи досягти вищої швидкості обробки транзакцій. Layer 1 блокчени можуть мати будь-яку кількість блокчейнів другого рівня. Звісно, як уже згадувалося, рух у бік масштабованості, особливо через реєстрацію транзакцій поза основним ланцюгом, призводить до проблем з безпекою та децентралізацією, які потрібно вирішувати за допомогою відповідних контрзаходів.

Щоб краще зрозуміти ситуацію та те як працюють ці рішення, варто розглянути їх основні типи детальніше.

2.1.1 State Channels

Рішення дозволяє створювати позаланцюгові канали зв’язку між учасниками мережі [8] [9]. Вони особливо корисні для сценаріїв, де потрібні часті транзакції між фіксованим набором учасників. Учасники блокують частину своїх

коштів у мультипідписному смарт-контракті на Layer 1 блокчейні. Після цього вони можуть проводити необмежену кількість транзакцій поза блокчейном, обмінюючись підписаними повідомленнями та оновлюючи стан каналу. Коли учасники вирішують закрити канал, фінальний стан передається на основний блокчейн, і кошти розподіляються відповідно до нього (див. Рис. 2.1). Lightning Network для Bitcoin та Raiden Network для Ethereum є прикладами рішень, заснованих на каналах стану.

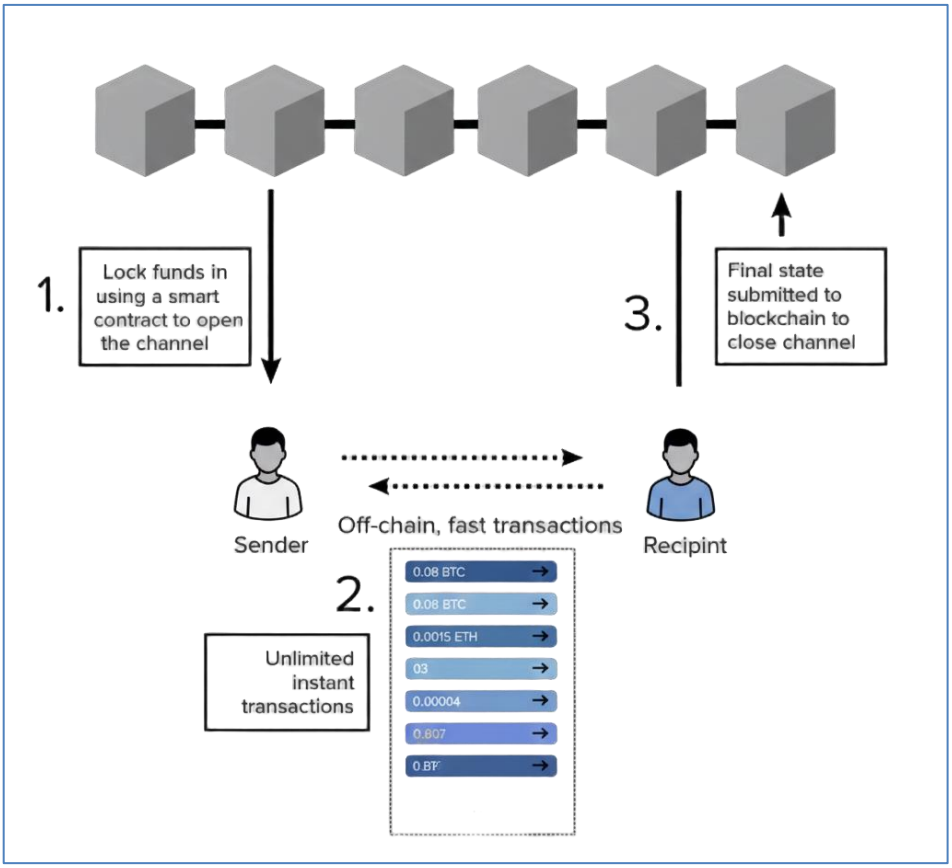


Рисунок 2.1 – Схема зв’язків Layer 2 рішення на основі каналів

2.1.2 Sidechains

Layer 2 у цьому випадку базується на дочірніх блокчейнах, які прив'язані до основного ланцюга й працюють паралельно з ним (див. Рис. 2.2). Ідея в цьому схожа на канали, але головна відмінність полягає в тому, що в сайдчейнах

позаланцюгові транзакції обробляються на блокчейнах, тоді як канали не базуються на блокчейні [8] [9]. Основна перевага перенесення транзакцій на інший блокчейн полягає в тому, що сайдчейни зазвичай включають меншу кількість вузлів і часто мають інший баланс між масштабованістю, безпекою та децентралізацією. Як правило, вони є менш децентралізованими, але швидшими. Транзакції, оброблені сайдчейном, як і у випадку з каналами, реєструються на Layer 1 лише з “відкриттям” та “закриттям”. У цьому контексті окремо розглядається архітектура Plasma. Plasma – це рішення для масштабування блокчейну Ethereum на рівні Layer 2, запропоноване у 2017 році. Воно спрямоване на розширення концепції сайдчейнів з метою зменшення кількості транзакцій, які необхідно обробляти на блокчейні першого рівня (L1), а також зменшення затримок і вартості транзакцій.

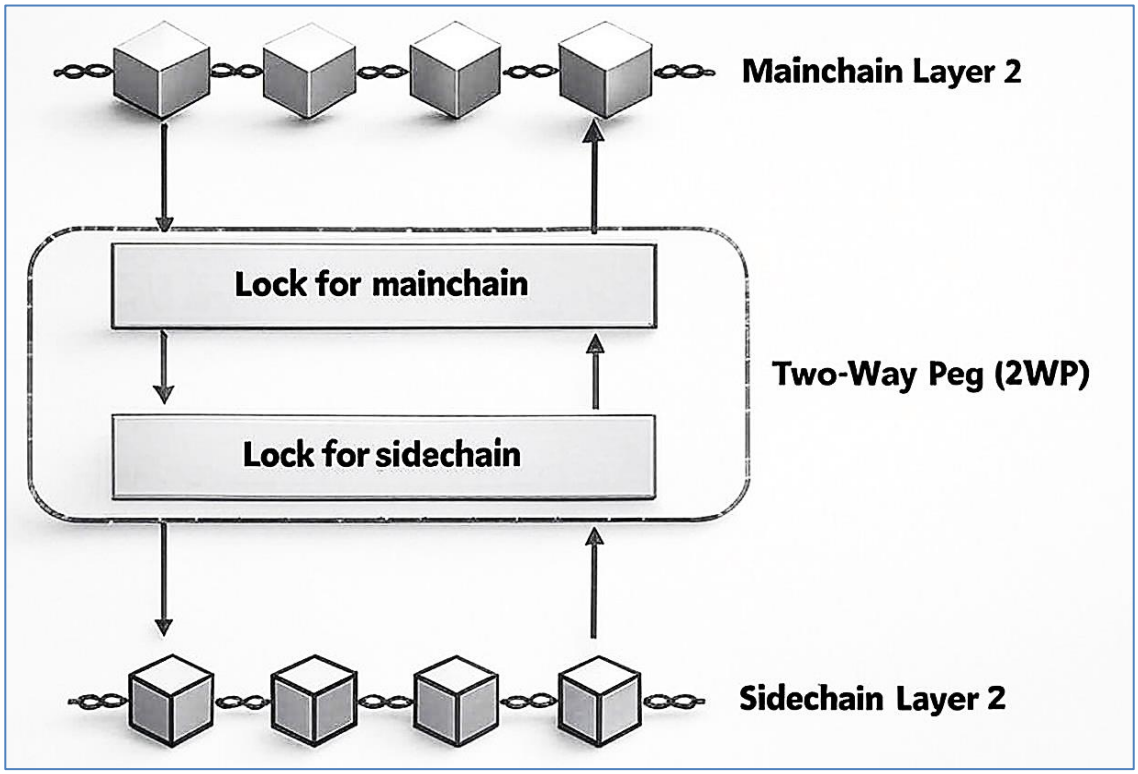


Рисунок 2.2 – Схема зв’язків Layer 2 рішення на основі сайдчейну

2.1.3 Rollups

Рішення базується на загальній концепції виконання транзакцій поза основним ланцюгом, але при цьому обов'язково реєструються дані про них на Layer 1 (див. Рис. 2.3). На практиці, у той час як канали та сайдчейни подають на блокчейн лише підсумок транзакцій, ролапи передають на основний ланцюг менший обсяг даних про кожне оновлення стану. Концепція Rollups реалізується як смарт-контракт, розгорнутий на основному ланцюгу, та відстежує поточний меркл-корінь стану Rollup (на практиці це може бути ланцюг другого рівня) [8]. Обчислення нового стану відбувається поза основним ланцюгом, тоді як root chain використовується для забезпечення доступності даних (шляхом публікації частини даних для кожної транзакції). Такий підхід дозволяє уникнути проблеми приховування даних, оскільки вся інформація завжди доступна на основному ланцюгу (L1), який вважається безпечним і надійним. Будь-який учасник може опублікувати батч – набір стиснутих транзакцій із прикріпленими попереднім та новим меркл-коренями. Перед оновленням стану на новий корінь, Rollup-контракт повинен перевірити, чи відповідає попередній корінь поточному. Унікальною особливістю Rollup є те, що він дозволяє проводити транзакції поза межами самого Rollup-контракту: це створено для підтримки операцій, вхідні-вихідні дані яких надходять ззовні. Основна ми типами роллапів є ZK Rollups та Optimistic Rollups.

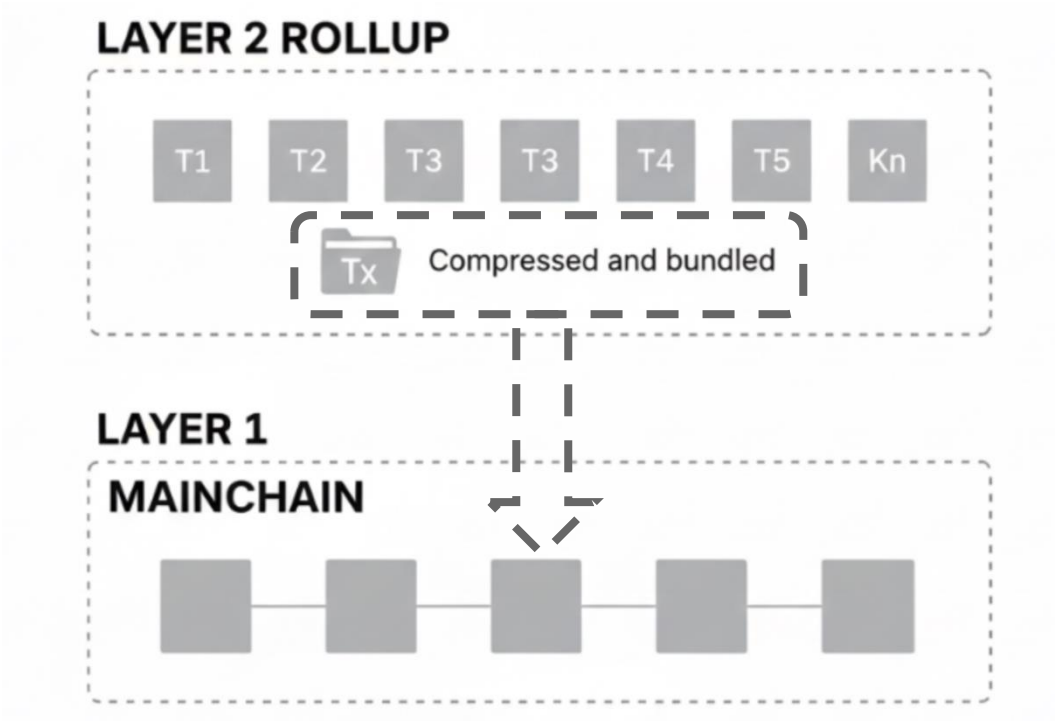


Рисунок 2.3 – Схема зв’язків Layer 2 рішення на основі Rollup

Рішення ZK Rollups базується на концепції доказів валідності (validity proof) та доказів із нульовим розголошенням (zero-knowledge proof). Основна ідея ZK Rollup полягає в тому, щоб додавати до кожного пакета транзакцій доказ їхньої валідності. Цей доказ має бути простим для перевірки, підтверджуючи правильність вмісту пакета. У рамках ZK Rollup використовується доказ SNARK – такий підхід дозволяє спостерігачам негайно підтвердити валідність твердження. Головною характеристикою цього типу доказів є те, що їх перевірка коштує недорого. Однак їх обчислення є дорогим. Таким чином, ZK Rollup є ефективною системою для управління транзакціями, але вона не підходить для виконання складних смарт-контрактів.

Рішення Optimistic Rollups базується на принципі “оптимістичності”. На відміну від ZK Rollups, вони не додають доказів валідності кожного пакета транзакцій під час їхнього публікування на основному ланцюзі. Натомість система передбачає, що всі транзакції є коректними за замовчуванням. Лише у

випадку, якщо хтось запідозрить помилку або шахрайство, ініціюється процес перевірки, який дозволяє оскаржити транзакцію. Складні перевірки можливі лише для окремих, оскаржених транзакцій. Для забезпечення безпеки використовуються механізми доказів шахрайства (fraud proofs), які дозволяють будь-якому учаснику мережі перевірити та довести неправильність певної транзакції. У разі успішного викриття недійсна транзакція анулюється, а винна сторона може бути покарана економічно. Optimistic Rollups потребують певного періоду затримки для виходу користувачів із другого рівня на основний блокчейн, щоб дати можливість оскаржити транзакції. Це є одним із їхніх основних недоліків.

2.2 Протокол LayerZero

LayerZero – це перший протокол передачі повідомлень, який дозволяє здійснювати нативні транзакції між ланцюгами. LayerZero забезпечує прямі крос-ланцюгові транзакції без необхідності використання централізованих, або децентралізованих бірж. LayerZero є першим безпечним омніланцюговим шаром сумісності і підтримує передачу повідомлень між ланцюгами як Layer 1, так і Layer 2 [10] [11].

Крос-ланцюгова транзакція між ланцюгами A та B складається з транзакції t_A на ланцюгу A, комунікаційного протоколу між A та B і повідомлення. Valid delivery означає, що повідомлення доставляється, якщо і тільки якщо t_A є підтвердженою і дійсною. Ключова ідея, яка стоїть за LayerZero, полягає в тому, що якщо дві незалежні сутності підтверджують дійсність транзакції (у даному випадку t_A), то ланцюг B може бути впевнений, що t_A є дійсною (див. Рис. 2.4) [12].

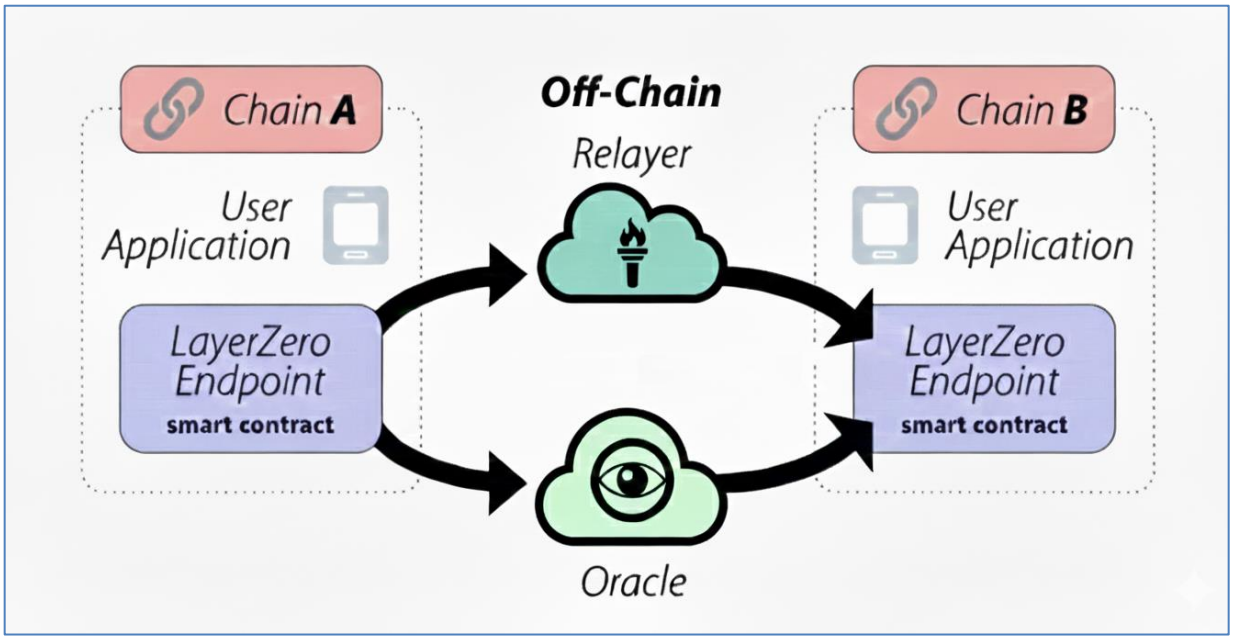


Рисунок 2.4 – Процес підтвердження транзакції в двох чейнах

Якщо одна сутність (Oracle) може надати блок-заголовок для блоку, що містить t_A на ланцюгу A, то інша сутність (Relayer) може незалежно надати доказ для t_A у цьому блоці (доказ транзакції), і коли заголовок та доказ транзакції узгоджуються, то комунікаційний протокол може доставити повідомлення на клієнт ланцюга B з гарантією, що t_A стабільно зафіксовано на ланцюгу A.

Інтерфейс LayerZero – це легкий клієнт на ланцюзі, який називається LayerZero Endpoint. Один LayerZero Endpoint існує на кожному підтримуваному ланцюзі, і будь-який ланцюг з Endpoint може здійснювати крос-ланцюгові транзакції з іншими ланцюгами, що мають Endpoint. Це фактично створює повністю з'єднану мережу, де кожен вузол має пряме з'єднання з кожним іншим вузлом. За допомогою мінімальної шаблонної кодифікації, будь-який блокчейн може бути підтриманий [13].

Здійснення прямих крос-ланцюгових транзакцій з будь-яким іншим ланцюгом мережі відкриває можливості для нових великомасштабних застосунків, які раніше були неможливі, таких як крос-ланцюгові децентралізовані біржі, багатоланцюгові агрегатори доходу та крос-ланцюгове

кредитування. Через LayerZero користувачі можуть вільно переміщувати ліквідність між ланцюгами, дозволяючи єдиному пулу ліквідності брати участь у різних децентралізованих фінансових (DeFi) застосунках на різних ланцюгах і в екосистемах без необхідності проходити через сторонні системи чи проміжні токени.

2.3 Міжмережева взаємодія через Enso Build

Це нове рішення для взаємодії між блокчейнами яке дозволяє обмінюватись токенами між мережами. Enso має інтеграцію з багатьма онлайн-обмінниками та працює на різних блокчейнах (Ethereum, Polygon, Arbitrum, Optimism, Avalanche, Fantom тощо). Цей сервіс прокладає маршрут для обміну і повертає транзакцію. Для розробників він надає можливість швидко будувати web3 застосунки не заглиблюючись у внутрішні процеси.

Платформа позиціонується як провідний постачальник “блокчейн-шорткатів” (blockchain shortcuts) та API middle layer, який уніфікує кожен блокчейн-фреймворк в єдину мережу. Основна мета Enso – надати розробникам єдину точку доступу для читання, запису та взаємодії з будь-яким смарт-контрактом на будь-якому ланцюгу через єдину інтеграцію. Це дозволяє розробляти та запускати застосунки значно швидше, перетворюючи складні багатоетапні транзакції на спрощені “шорткати” [14].

Одним з найбільш значущих аспектів Enso є орієнтація на кінцевий результат. Замість того, щоб вимагати від розробників написання складних, покрокових процесів для взаємодії зі смарт-контрактами на різних блокчейнах, Enso дозволяє їм просто визначати “наміри”. Система Enso потім бере на себе всю технічну складність виконання цих намірів, автоматично знаходячи та реалізуючи оптимальний шлях для досягнення поставленої мети. Наприклад, якщо розробник хоче здійснити крос-чейн обмін токенів, йому не потрібно

вручну прописувати кожен етап – від виведення ліквідності до мосту, та повторного розгортання на цільовому ланцюгу. Достатньо лише вказати бажане, і Enso визначить та виконає найефективніший шлях, оптимізуючи його за такими параметрами, як комісії за газ та кінцева винагорода. Enso розширює цю концепцію, пропонуючи три типи абстракції:

Явні наміри (Explicit Intent): Користувач чітко вказує протокол операції, наприклад, “Я хочу використовувати кредитне плече на Aave”.

Напів’явні наміри (Semi-Explicit Intent): Користувач визначає лише цільове кредитне плече або ставку прибутковості, дозволяючи системі автоматично обрати оптимальний шлях між різними протоколами.

Повністю абстрактні наміри (Fully Abstract Intent): Штучний інтелект обирає найкраще рішення на основі історії користувача та ринкових умов, вимагаючи мінімальних явних інструкцій від користувача. Ця модель значно спрощує складність взаємодії для користувача.

Інновацією в підході Enso є його спільний стан мережі, який централізує дані з різних джерел в єдиний реєстр. Ця уніфікація даних має вирішальне значення, оскільки вона зменшує складність, підвищує точність та значно покращує взаємодію між блокчейнами і використовується для побудови “шорткатів”.

Цей підхід відображає перехід від прямого кодування до декларативної взаємодії. Це зміна, яка робить блокчейн доступним для значно ширшого кола розробників, включаючи тих, хто не є глибокими експертами в тонкощах смарт-контрактів. Такий підхід підвищує зручність для нетехнічних користувачів та розробка Web3 починає відчуватися як розробка Web2, що є кроком до мейнстріму. Зрештою, це є ключовим для подолання складності побудови реальних продуктів, що вважається найбільшим бар’єром для масового впровадження Web3.

2.4 Підсумки розділу 2

Загальний принцип усіх рішень Layer 2 полягає в розвантаженні основного блокчейна, щоб сприяти його масштабуванню тому ці рішення мають кращі характеристики (див. табл. 2.1). Наслідком цього є вища швидкість обробки транзакцій і як правило, нижчі комісії. Це є прямим результатом збільшення пропускної здатності (TPS). Тому Layer 2 є оптимальним рішенням для виконання мікроплатежів. Додатковими перевагами Layer 2 є те, що ніякі зміни на основному ланцюгу не потрібні, а управління транзакціями поза блокчейном є незалежним від Layer 1. Насправді певна залежність існує, оскільки на Layer 1 потрібно реєструвати “підсумок” транзакцій, але, окрім цього, блокчейн не має уявлення про те, що відбувається на Layer 2.

Окремо слід згадати шардінг як ще одне рішення для масштабування блокчейнів. Ця техніка базується на розділенні основного ланцюга на підмножини вузлів, кожна з яких відповідає за певну частину мережі [4]. Кожен вузол обробляє лише інформацію, що належить його шарду. Це допомагає масштабувати блокчейн, оскільки розподіл навантаження між різними підмножинами вузлів призводить до швидшої обробки транзакцій. Проте, на відміну від Layer 2, шардінг є Layer 1 рішенням, оскільки всі транзакції обробляються на основному блокчейні. При цьому шардінг, як і Layer 2 рішення, зменшує децентралізацію та безпеку, підкреслюючи важливість тріади масштабованості.

Таблиця 2.1 – Порівняння Layer 1 та Layer 2 рішень

Характеристика	Layer 1	State Channels	Sidechains	Rollups
Комісії	Високі	Майже нульові. Оплата лише за відкриття та закриття каналу	Залежить від консенсусу, зазвичай низька	Помірні, через агрегацію даних
Швидкість транзакцій	Висока через час підтвердження блоків	Низька, майже миттєва	Помірна. Залежить від механізму консенсусу	Низька для локальних транзакцій, але при Optimistic Rollups є затримка на L1 для оскарження
Безпека	Гарантована самим блокчейном	Залежить від довіри між учасниками каналу	Вимагає довіри до оператора сайдчейну	Висока: ZK Rollups гарантують доказ валідності; Optimistic Rollups покладаються на оскарження

Продовження таблиці 2.1

Характеристика	Layer 1 (L1)	State Channels	Sidechains	Rollups
Масштабованість	Обмежена	Підходить для невеликих груп учасників	Підходить для спеціалізованих застосунків	Висока, придатна для широкого використання
Механізм консенсусу	Proof-of-Work або Proof-of-Stake	Узгодження між учасниками	Proof-of-Authority або інші механізми	Залежить від L1; Proof-of-Fraud у Optimistic Rollups, SNARK у ZK Rollups
Обмеження	Низька продуктивність, високі комісії	Потреба в постійній онлайн-присутності	Централізація та довіра до оператора	Високі витрати на обчислення (ZK), затримки на оскарження (Optimistic)

Важливо зазначити, що різні Layer 2 протоколи можуть бути об'єднані поверх одного і того ж основного блокчейна для максимізації його масштабованості. Це можливо завдяки тому, що блокчейн залишається незміненим і не зазнає впливу від використання Layer 2 рішень.

Використовуючи блокчейни другого рівня та протокол LayerZero разом з Enso, можна побудувати застосунок який зможе працювати з кількома блокчейнами надійно та швидко.

3 РОЗРОБКА ДЕЦЕНТРАЛІЗОВАНОГО ОМНІЧЕЙН ЗАСТОСУНКУ

3.1 Бізнес-обґрунтування та вимоги

Створення децентралізованого омнічейн застосунку з використанням LayerZero, Enso Build та Layer 2 є перспективним бізнес-рішенням. Мета такого застосунку – забезпечити користувачам можливість проводити швидкі, дешеві та безпечні транзакції між різними блокчейнами.

Застосунок дозволить користувачам здійснювати обмін токенів як в межах одного блокчейну, так і між різними мережами. Завдяки використанню LayerZero забезпечується кроссчейн сумісність основного токену додатку, що дозволяє ліквідності циркулювати між різними блокчейнами без необхідності створювати окремі пули для кожної мережі, а Enso Build вирішує проблему обміну токенів як всередині однієї так і між мережами. Це знижує ризик фрагментації ліквідності та покращує досвід користувачів. Використання Layer 2 технологій, таких як ZK Rollups, надає високу пропускну здатність і значно знизить вартість транзакцій, що буде привабливим для залучення масового користувача.

З точки зору бізнесу, застосунок має на меті зайняти нішу в сегменті DEX, орієнтуючись на користувачів, які цінують швидкість та низькі комісії, а також на тих, хто активно працює з кількома блокчейнами.

3.2 Вимоги до застосунку

3.2.1 Функціональні вимоги

Спочатку користувач має створити свій гаманець. Для цього можна скористатися MetaMask або опціями які пропонує застосунок. MetaMask є популярним кроссплатформенним рішенням і тому його інтеграція є необхідною [26].

Після реєстрації, користувач підключає свій криптогаманець до застосунку та вибирає блокчейн, на якому хоче працювати. Система перевіряє з’єднання з обраною мережею та дозволяє вибрати інші доступні мережі для подальшої взаємодії. Якщо користувач бажає здійснити обмін токенів, він вибирає пару токенів для свопу, вводить суму та отримує попередню інформацію про курс обміну, комісію та очікуваний результат операції.

Коли користувач підтверджує транзакцію, застосунок надсилає запит до Enso Shortcuts API та повертає готову транзакцію, яку залишається лише виконаити. У разі, якщо обмін потребує взаємодії з іншим блокчейном і використовується внутрішній токен, LayerZero обробляє крос-чейн запит, надсилаючи повідомлення про транзакцію між мережами. Після цього в іншій мережі виконується відповідний смарт-контракт, забезпечуючи переказ активів. Інформація про статус транзакції повертається користувачу, а токени надходять на гаманець. Якщо відбувається свап інших токенів, то запит обробляється Enso та далі повертається транзакція яку користувач має виконати. Інформація про транзакцію та її результат записується до БД.

Якщо користувач бажає додати ліквідність, він вибирає кількість токенів і монету для їх купівлі. Користувач підтверджує операцію, і система надсилає транзакцію до смарт-контракту який переводить необхідну кількість токенів на гаманець покупця.

У разі бажання користувача вивести ліквідність, система дозволяє обрати спосіб та відсоток ліквідності які користувач може отримати назад. LayerZero забезпечує синхронізацію між мережами, якщо ліквідність розподілена крос-чейн.

Для зручності використання застосунок має надавати інструменти моніторингу. Користувач може переглядати історію своїх транзакцій, токени та монети якими він володіє в приблизному еквіваленті до фіатних валют, поточні обмінні курси.

Усі дії користувача супроводжуються прозорим процесом обробки даних. Layer 2 рішення забезпечують швидке підтвердження транзакцій із мінімальними витратами, тоді як LayerZero гарантує надійну передачу даних і активів між мережами.

Цей процес також включає функції безпеки: перевірка дозволів на використання токенів (approval), відображення потенційних ризиків транзакцій і оповіщення про зміни в пулах ліквідності. Усе це створює максимально зручний та ефективний досвід для користувачів, які працюють у багатоланцюговому середовищі.

3.2.2 Нефункціональні вимоги

Застосунок повинен бути здатним обробляти одночасно велику кількість транзакцій без значних затримок.

Архітектура застосунку має підтримувати горизонтальне масштабування, дозволяючи додавати нові функції та підключати нові блокчейни без шкоди для продуктивності.

Система повинна забезпечувати мінімальний час відгуку, особливо для критичних операцій, таких як обмін токенів і внесення ліквідності.

Застосунок повинен забезпечувати стабільну роботу за будь-яких умов. Важливо впровадити резервні механізми для збереження даних і завершення транзакцій у разі збою.

Інтерфейс має бути інтуїтивно зрозумілим, навіть для новачків у DeFi. Застосунок повинен підтримувати кілька мов і можливість швидкого перемикання між блокчейнами. Важливо інтегрувати навчальні інструкції для користувачів.

Застосунок повинен підтримувати роботу з популярними криптогаманцями, такими як MetaMask або Trust Wallet. Система має бути побудована на стандартах, які дозволяють легко інтегрувати нові рішення або блокчейни в майбутньому.

Платформа повинна мати вбудовані інструменти для відстеження продуктивності, виявлення помилок і створення звітів про роботу системи. Це включає відстеження транзакцій, статусу ліквідності в пулах та активності користувачів.

3.3 Архітектура застосунку

Для бекенд частини застосунку обрано Event Driven архітектуру на AWS, що забезпечує стабільність та високу швидкість роботи (див. Рис. 3.1). API Gateway виступає шлюзом для мобільного додатка. Використання Lambda-функцій дозволить масштабувати систему відповідно до навантаження а також додавати нові компоненти до системи частинами. Використання EventBridge дозволяє керувати подіями, такими як транзакції чи оновлення даних, і направляти їх до відповідних функцій. S3 використовується для зберігання ресурсів, наприклад, зображень токенів чи інших статичних даних. Використання RDS Postgres забезпечує надійне зберігання даних користувачів, та журналу транзакцій. Redis використовується для кешування та тимчасового збереження інформації про сесії користувачів.

Мову програмування для бекенду обрано Node.js [18] що дозволяє легко працювати з форматом JSON та має швидший час запуску і є більш ефективною для короткочасних або частих викликів Lambda функцій. Також ця мова дозволяє легко використовувати Serverless фреймворк для менеджменту цих функцій.

Блокчейн-шар реалізує основні функції обміну та управління ліквідністю. LayerZero виступає інтероперабельним протоколом, який забезпечує зв’язок між блокчейнами. Він дозволяє взаємодіяти між Layer 1 і Layer 2 мережами. Також для свопів токенів використовується Enso. Layer 2 Network обрано для зниження вартості транзакцій та підвищення продуктивності. LayerZero забезпечує обмін активами між мережами, що дозволяє зберегти цілісність ліквідності.

Мобільний додаток реалізований за допомогою React Native, що забезпечує його кроссплатформенність і використання однієї мови і для нього і для бекенду. Інтеграція криптогаманців за допомогою Reown у мобільний додаток створює прямий зв’язок між користувачем і блокчейн-шаром. Це спрощує процес авторизації та підписання транзакцій, дозволяючи користувачам зберігати контроль над своїми коштами без необхідності довіряти центральному серверу.

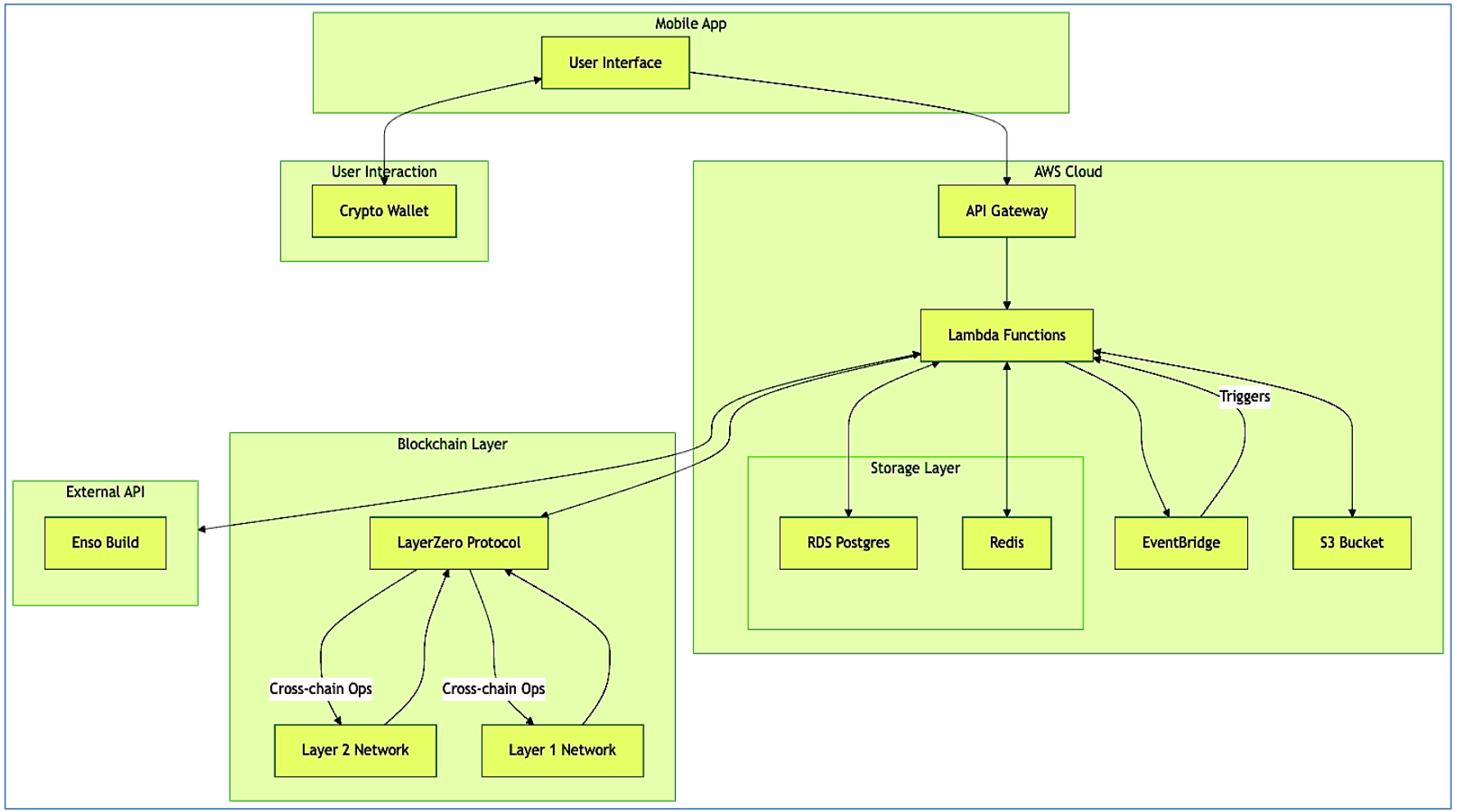


Рисунок 3.1 – Схема взаємодії між архітектурними компонентами

3.4 Розробка бази даних

Для зберігання користувацької інформація було обрано PostgreSQL. PostgreSQL дотримується принципів ACID (Atomicity, Consistency, Isolation, Durability), що гарантує цілісність даних що важливо для зберігання журналу транзакцій. PostgreSQL є повністю відкритим проектом – він безкоштовний у використанні, модифікації та розповсюдженні. Велика спільнота постійно працює над його удосконаленням, виправленням помилок та додаванням нових функцій, що забезпечує розвиток і підтримку. Ця база має потужний механізм для виконання складних SQL-запитів і підтримує різноманітні типи даних, включаючи JSON. Це дозволяє зберігати структуровані та неструктуровані дані в одній базі даних. В подальшому такі дані буде легше аналізувати і створювати звіти на основі них [20].

Базова структура БД та зв’язки між таблицями наведено на рисунку (див. Рис. 3.2).

1. **Wallet**: Таблиця для зберігання інформації про гаманці. Вона містить унікальний ідентифікатор, ID користувача, тип гаманця (внутрішній або зовнішній), адресу гаманця та інформацію про підключення.

2. **Transaction**: Таблиця для зберігання інформації про транзакції. Вона містить унікальний ідентифікатор транзакції, ID гаманця відправника та отримувача, тип активу, кількість активу, мережі відправлення та отримання, а також дату та час транзакції.

3. **User**: Таблиця для зберігання інформації про користувачів. Вона містить унікальний ідентифікатор користувача, ім’я користувача та електронну адресу.

4. **Asset**: Таблиця для зберігання інформації про активи. Вона містить унікальний ідентифікатор активу, його назву та символ.

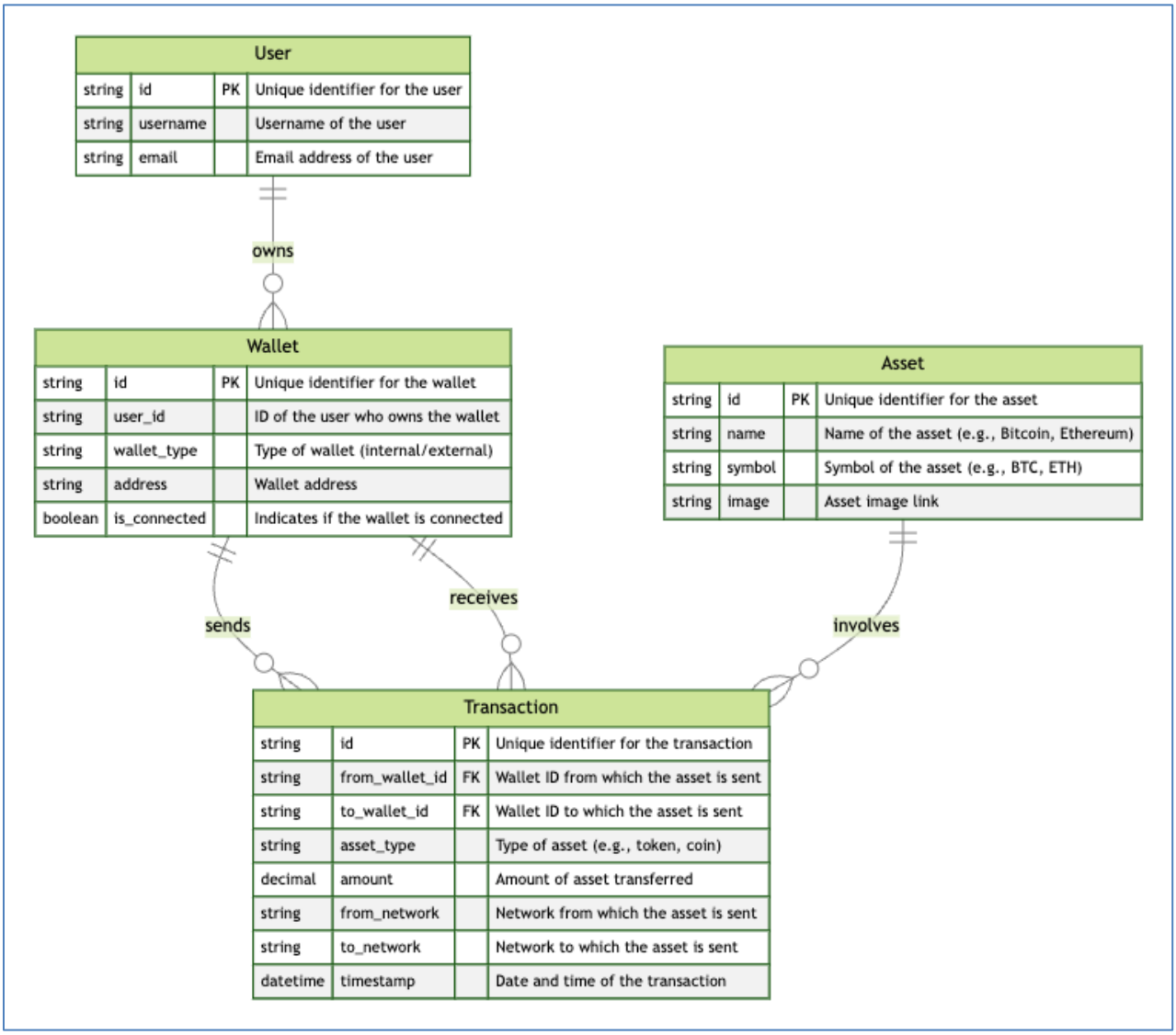


Рисунок 3.2 – Схема бази даних застосунку

3.5 Розробка мобільної частини застосунку

React Native, розроблений Facebook (тепер Meta), залишається одним з найпоширеніших для крос-платформної мобільної розробки. Він дозволяє розробникам використовувати JavaScript для створення мобільних застосунків на iOS та Android. JavaScript є основною мовою програмування React Native, що робить його доступним для великої спільноти веб-розробників, які вже володіють JavaScript. Для підвищення якості коду та зручності обслуговування в проєктах

React Native часто використовується TypeScript, який додає до JavaScript статичну типізацію та фішки ООП. Оскільки для розробки бекенду буде використовуватися Node.js, то вибір React Native є обгрунтованим, бо ми використаємо одну мову програмування для обох частин застосунку [22].

Архітектура React Native перекладає JavaScript-код у нативні компоненти для кожної мобільної платформи, діючи як міст між JavaScript та нативними мовами. Це означає що код буде працювати практично однаково на кількох платформах. Останні архітектурні вдосконалення React Native, такі як JavaScript Hermes та JavaScript Interface (JSI), спрямовані на покращення продуктивності та зменшення залежності від традиційного мосту JavaScript-Native, який раніше міг бути джерелом вузьких місць. Використання Hermes, зокрема, призводить до покращення часу запуску, зменшення використання пам’яті та зменшення розміру встановленого застосунку.

Для управління станом застосунку обрано Zustand, який виділяється простотою та високою продуктивністю. На відміну від Redux, який вимагає налаштування редюсерів, екшенів, диспатчерів та іноді саг, Zustand дозволяє створити глобальний стан за допомогою однієї простої функції. Zustand оптимізований для високої продуктивності, він не використовує Context API для зберігання стану, що дозволяє уникнути зайвих ререндерів компонентів. Замість цього, він використовує механізм підписки, який гарантує, що рендер відбувається лише тоді, коли змінюється та частина стану, на яку підписаний компонент. Це особливо корисно у великих додатках з частими оновленнями стану. Бібліотека Zustand має невеликий розмір, що позитивно впливає на загальний розмір бандлу та час завантаження додатку.

Для взаємодії між застосунком та бекендом обрано GraphQL, що має ряд суттєвих переваг. Замість завантаження цілих ресурсів, як це часто відбувається у REST API, GraphQL дозволяє точно вказати поля, які вам потрібні. Це значно зменшує обсяг переданих даних і прискорює завантаження застосунку навіть для

мобільних пристроїв з повільним інтернетом. GraphQL API є типізованим і кожен запит валідується на відповідність схемі, забезпечуючи більшу надійність. На основі схеми також генерується автоматична документація, що значно спрощує розуміння доступних даних для фронтенд-команди. З GraphQL зазвичай використовується Apollo Client. Це комплексна бібліотека для управління станом додатку на основі даних GraphQL. Він розроблений для того, щоб зробити взаємодію з GraphQL API максимально простою та ефективною. Він обробляє все: від надсилання запитів до серверу, до кешування та нормалізації даних [23].

Для взаємодії з криптогаманцями обрано Reown (колишній WalletConnect). Reown виступає як універсальний протокол з відкритим вихідним кодом, що дозволяє популярним мобільним гаманцям безпечно та легко підключатися до dApps, незалежно від платформи [16]. Замість того, щоб React Native застосунок напряму інтегрувався з MetaMask, Trust Wallet та багатьма іншими, Reown пропонує єдиний стандартизований протокол і навіть готові програмні компоненти. Основний принцип роботи полягає у встановленні зашифрованого з'єднання між React Native dApp та обраним користувачем гаманцем який встановлено на мобільний пристрій. Це з'єднання підтримується через релейний сервер, який передає зашифровані повідомлення і дозволяє виконувати транзакції.

3.5.1 Налагодження локального середовища

Розробка мобільного застосунку на React Native вимагає налаштування локального середовища. Для управління програмним забезпеченням, знадобиться Homebrew, популярний менеджер пакетів для macOS. Він спрощує встановлення багатьох необхідних компонентів. Далі необхідно встановити Node.js для виконання JavaScript-коду в React Native, а також менеджер керування залежностями – npm або Yarn.

Для розробки під iOS потрібно встановити Xcode – середовище розробки від Apple. Після завантаження та встановлення Xcode з Mac App Store, важливо

також встановити Command Line Tools, які є набором інструментів командного рядка, необхідних для компіляції та запуску iOS-застосунків.

Розробка під Android вимагає встановлення Android Studio. Це повноцінне IDE, що включає в себе Android SDK, необхідний для розробки, а також емулятор та інші інструменти. Також для Android розробки, треба встановити Java Development Kit (JDK).

3.5.2 Особливості програмної реалізації

Підключення до гаманця було реалізовано як окремий компонент-кнопка. Для підключення до гаманця і мереж було використано Node.js бібліотеки Reown та бібліотеку Wagmi, яка реалізує взаємодію через React Hooks. Компонент доступний на верхній панелі екрану та відкриває модальне вікно з усіма можливими способами під'єднання гаманця. Якщо використати соціальні мережі, або натиснути Continue with Email, то гаманець буде створено автоматично (див. Рис. 3.3).

Після того як гаманець під’єднано, натискання на компонент-кнопку відкриває модальне вікно яке дозволяє змінити мережу, або відєднати його (див. Рис. 3.4). Код даного компоненту наведено в лістингу 1.

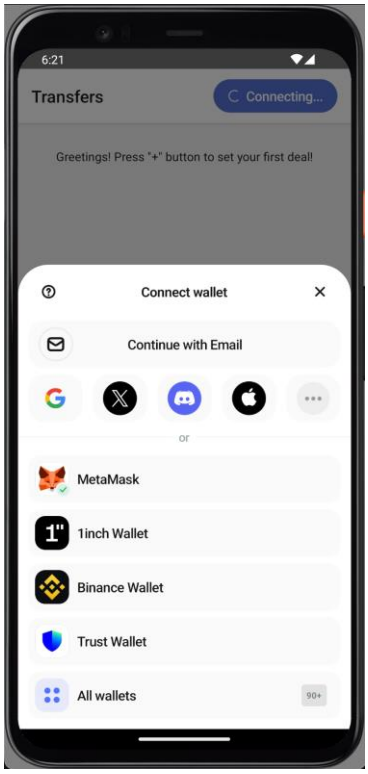


Рисунок 3.3 – Модальне вікно підключення гаманця

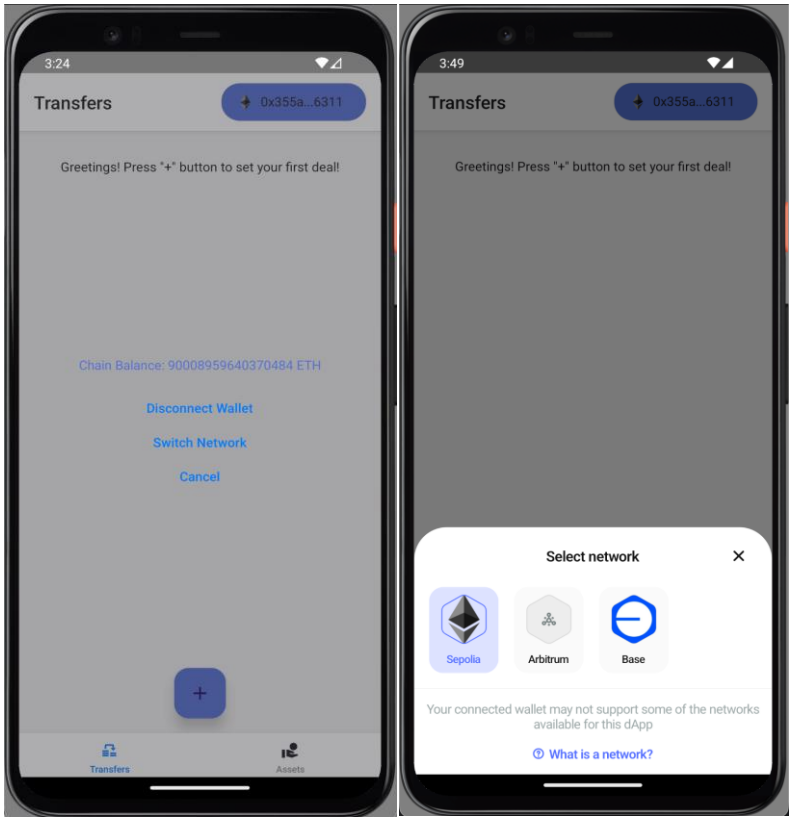


Рисунок 3.4 – Модальні вікна для відключення гаманця та зміни мережі

Лістинг 1 – Компонент-кнопка для роботи з гаманцем:

```
import { AssetController, AssetUtil } from '@reown/appkit-core-react-native';
import {
  AppKitNetwork,
  ConnectButton,
  useAppKit,
} from '@reown/appkit-react-native';
import React, { useState } from 'react';
import { Image, StyleProp, View, ViewStyle } from 'react-native';
import { Button, Icon, Modal, Portal, Text } from 'react-native-paper';
import { Address, Chain } from 'viem';
import { useAccount, useBalance } from 'wagmi';
import { walletInfoStyles } from './WalletConnectButton.styles';

interface WalletConnectButtonProps {
  style?: StyleProp<ViewStyle>;
}

export const WalletConnectButton = ({
  style,
}: WalletConnectButtonProps): React.JSX.Element => {
  const { isConnected, address, chain } = useAccount();

  return (
    <>
      {isConnected && address ? (
        <WalletInfo
          walletAddress={address as Address}
          chain={chain}
          style={style}
        />
      ) : (
        <ConnectButton
          label="Connect Wallet"
          loadingLabel="Connecting..."
          style={[style, walletInfoStyles.sharedButton]}
        />
      )}
    </>
```

```
  );
};

interface WalletInfoProps {
  walletAddress: Address;
  chain?: Chain;
  style?: StyleProp<ViewStyle>;
}

const WalletInfo = ({
  walletAddress,
  chain,
  style,
}: WalletInfoProps): React.JSX.Element => {
  const [modalVisible, setModalVisible] = useState(false);
  const { data: balance, isLoading } = useBalance({
    address: walletAddress,
  });
  const { disconnect, open } = useAppKit();
  const networkImage = AssetUtil.getNetworkImage(
    chain as AppKitNetwork,
    AssetController.state.networkImages,
  );

  const handlePress = () => {
    setModalVisible(true);
  };

  const handleDisconnect = () => {
    disconnect();
    setModalVisible(false);
  };

  const handleNetworkSwitch = () => {
    open({ view: 'Networks' });
    setModalVisible(false);
  };

  const icon = ({ size }: { size: number }) =>
    networkImage ? (
      <Image
        source={{ uri: networkImage }}
        style={{ width: size, height: size }}
      />
    ) : (
      <Icon source={'wallet'} size={size} />
```

```
        );

      return (
        <View>
          <Button
            mode="outlined"
            onPress={handlePress}
            style={[style, walletInfoStyles.sharedButton]}
            icon={icon}
          >
            <Text>{`${walletAddress.slice(0,
6)}...${walletAddress.slice(-4)}`}</Text>
          </Button>

          <Portal>
            <Modal  visible={modalVisible}  onDismiss={()  =>
setModalVisible(false)}>
              <Text style={walletInfoStyles.balance}>
                {isLoading
                  ? 'Loading balance...'
                  :    `Chain    Balance:    ${balance?.value}
${balance?.symbol}`}
              </Text>
              <Button    onPress={handleDisconnect}>Disconnect
Wallet</Button>
              <Button     onPress={handleNetworkSwitch}>Switch
Network</Button>
              <Button               onPress={()              =>
setModalVisible(false)}>Cancel</Button>
            </Modal>
          </Portal>
        </View>
      );
    };
```

3.6 Розробка бекенд частини застосунку

Оскільки бекенд будується на основі хмарної інфраструктури та сервісах AWS, з використанням Node.js, то це дозволить застосунку масштабуватися адаптуючись до навантажень. Якщо застосунок переживає раптовий сплеск

трафіку, AWS автоматично виділяє більше ресурсів, щоб забезпечити стабільну роботу. Коли навантаження спадає, ресурси зменшуються, оптимізуючи витрати.

AWS Lambda – це обчислювальний сервіс AWS, який дозволяє запускати код без необхідності забезпечувати або керувати серверами. Достатньо завантажити свій код, і Lambda автоматично масштабує його виконання, реагуючи на події, такі як HTTP-запити, зміни в базах даних, завантаження файлів у S3 або повідомлення в чергах. Використовуючи AWS Lambda можна легко будувати бекенд та надавати необхідний функціонал для фронтенду. Не потрібно думати про операційні системи, їх оновлення, місткість або доступність серверів. AWS повністю бере на себе це завдання, а розробник може зосередитися на програмному продукті.

Незважаючи на переваги Lambda, управління такими застосунками що складаються з багатьох функцій, API Gateway, та інших ресурсів, може бути складним. Тут на допомогу приходить Serverless Framework.

Serverless Framework – це CLI-інструмент, який спрощує розробку, розгортання та управління Lambda-функціями. Він дозволяє описати архітектуру (функції, події, ресурси) у простому YAML-файлі, та автоматично перетворює цей опис на необхідні CloudFormation шаблони для AWS, що усуває потребу в ручному налаштуванні. Фреймворк надає інструменти для тестування функцій без розгортання в хмарному середовищі. Має багато плагінів що дозволяють розширювати функціональність фреймворку, додаючи можливості для відлагодження, управління, оптимізації бандлів тощо [24].

3.6.1 Налагодження локального середовища

Розробка бекенду потребує налагодження локального середовища, яке б імітувало продакшн-середовище. Для серверлес-бекенду на AWS, використовується поєднання LocalStack для імітації сервісів AWS, Docker для

управління базами даних Redis та PostgreSQL, та Node.js з Serverless Framework для написання і розгортання лямбда-функцій у це локальне середовище.

LocalStack емулює хмарні сервіси, та надає повнофункціональний локальний “хмарний” провайдер для розробки та тестування ваших AWS-застосунків. Він підтримує більшість популярних сервісів AWS, включаючи Lambda та API-Gateway [17].

Docker відіграє головну роль, дозволяючи нам створювати та управляти “контейнерами” для наших баз даних та для LocalStack. Контейнери ізольовані один від одного та від основної операційної системи. Це означає, що сам бекенд (всередені LocalStack) та бази даних (PostgreSQL, Redis) працюють у власних, чистих середовищах, не конфліктуючи з іншими програмами на комп'ютері. Більше того, контейнери є портативними, тож все працюватиме однаково на будь-якій машині, де встановлено Docker. Контейнери є легшими, ніж віртуальні машини, оскільки вони використовують ядро операційної системи хоста. Це дозволяє запускати декілька сервісів одночасно без значного навантаження на ресурси комп’ютера [19].

3.6.2 Особливості програмної реалізації

Для роботи з базою даних було обрано Prisma ORM. Ця ORM дозволяє легко працювати з БД Postgres та автоматично генерує клієнт і типи даних для роботи з БД. Оскільки клієнт для роботи з базою даних використовується в різних лямбда-функціях і має бути доступним в кількох репозиторіях, то було прийнято рішення винести його в окремий git-субмодуль і підключати до кожного репозиторію окремо.

Для менеджменту сесій клієнта було обрано базу даних яка зберігає значення у форматі key-value – Redis [21]. Для кожного клієнта створюється структура даних хеш, яка використовується для зберігання набору пар “поле-значення” під одним ключем Redis. Приклад реалізації наведено в лістингу 2.

Лістинг 2 – Менеджмент сесій користувача:

```
import { keys } from '../keys';
import { getRedisInstance } from '../redis';

const redis = getRedisInstance();

export async function setUserSession(
  userId: string,
  sId: string,
  expiresInSeconds?: number,
) {
  const multi = redis.multi();
  multi.hset(`${keys.token}:${userId}`,                sId,
Date.now().toString());
  if (expiresInSeconds) {
    multi.call(
      'HEXPIRE',
      `${keys.token}:${userId}`,
      expiresInSeconds,
      'FIELDS',
      1,
      sId,
    );
  }
  await multi.exec();
}

export async function getUserSessions(
  userId: string,
): Promise<Record<string, string>>;
```

```
    export async function getUserSessions(
      userId: string,
      sId: string,
    ): Promise<string | null>;
    export async function getUserSessions(userId: string, sId?:
string) {
      if (sId) {
        return redis.hget(`${keys.token}:${userId}`, sId);
      }
      return redis.hgetall(`${keys.token}:${userId}`);
    }

    export async function removeUserSessions(
      userId: string,
      sId?: string,
    ): Promise<number> {
      if (sId) {
        return redis.hdel(`${keys.token}:${userId}`, sId);
      }
      return redis.del(`${keys.token}:${userId}`);
    }
```

Також слід відзначити те, що доступ до бекенду відбувається через лямбда-функцію, яка реалізує GraphQL з Apollo Server. Він допомагає швидко налаштувати робочий GraphQL API, забезпечуючи перевірку запитів. Тож клієнт робить GraphQL-запит до цієї лямбди, а вона вже перенаправляє його до інших лямбд які реалізують необхідний функціонал.

Щоб користувач мав завжди актуальний список токенів, які доступні в різних мережах, було інтегровано API Coingecko. Він надає інформацію про ціни, та охоплює тисячі криптовалют та токенів. Користувач зможе в результаті

отримати вартість своїх активів у різних фіатних валютах (USD, EUR, UAH тощо).

3.7 Підсумки розділу 3

Архітектура застосунку базується на Event-Driven парадигмі та використанні сервісів AWS таких як Lambda та API Gateway, що забезпечує гнучкість, масштабованість та високу швидкість роботи.

Розробка бази даних з використанням PostgreSQL обґрунтована ACID-сумісністю, відкритим вихідним кодом, підтримкою JSON-даних, та можливістю масштабувати БД як вертикально так і горизонтальнододаючи репліки, що важливо для забезпечення цілісності даних та стабільної роботи застосунку.

Мобільна частина застосунку реалізується за допомогою React Native, що забезпечує його крос-платформенність і дозволяє використовувати єдину мову програмування як для фронтенду, так і для бекенду (Node.js). Це значно спрощує процес розробки та підтримки.

Взаємодія між мобільним додатком та бекендом реалізується через GraphQL, що є ефективнішим рішенням порівняно зі звичайним REST API. Цей вибір обґрунтований його здатністю точно вказувати необхідні поля, що зменшує обсяг переданих даних і прискорює їх завантаження, а це важливо для мобільних пристроїв. GraphQL API забезпечують автоматичну документацію. На бекенді та у мобільних застосунках використовується бібліотека Apollo. Вона надає повний набір інструментів, що спрощують роботу з GraphQL. Наприклад, на стороні сервера Apollo Server допомагає швидко налаштувати GraphQL API, а на стороні клієнта Apollo Client забезпечує кешування даних, що ще більше прискорює роботу програми.

Для локальної розробки serverless-бекенду було використано LocalStack всередені Docker-контейнеру. Також Docker використовується для

контейнеризації баз даних, забезпечуючи швидкий цикл розробки та тестування роботи застосунку.

Налагодження локального середовища для мобільної розробки вимагає встановлення та налаштування додаткових інструментів таких як Xcode та Android Studio.

4 АНАЛІЗ РЕЗУЛЬТАТІВ РОЗРОБКИ ДЕЦЕНТРАЛІЗОВАНОГО ЗАСТОСУНКУ

4.1 Аналіз міжмережевого трансферу ліквідності

Для даного аналізу було розроблено LayerZero смарт-контракт на мові Solidity, який реалізує ERC20 токен (див. Лістинг 3). В контракті передбачена генерація початкової кількості токенів в мережі у розмірі 100000 одиниць.

Лістинг 3 – Контракт ERC-20 токену **EasyDEXOFT**:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.22;

import { Ownable } from "@openzeppelin/contracts/access/Ownable.sol";
import { OFT } from "@layerzerolabs/oft-evm/contracts/OFT.sol";
contract EasyDEXOFT is OFT {
    constructor(
        string memory _name,
        string memory _symbol,
        address _lzEndpoint,
        address _delegate
    ) OFT(_name, _symbol, _lzEndpoint, _delegate) Ownable(_delegate) {
        // Uncomment the line below to mint test tokens on deployment (for testnet only)
        _mint(msg.sender, 100000 * (10 ** 18));
    }
}
```

Даний контракт було задеплоєно до тестових мереж: **Ethereum Sepolia Testnet** та **Avalanche Fuji Testnet** (блокчейни першого рівня), **Base Sepolia Testnet** та **Arbitrum Sepolia Testnet** (блокчейни другого рівня).

Оскільки ми використовуємо відкриті мережі, то можна скористатися системами які сканують блокчейн і дозволяють відстежити задеплоєні контракти і виконані транзакції [25]. Прикладами таких систем є Layerzeroscan, Etherscan, Avascan, та Basescan. В цих системах ви можете навіть взаємодіяти з контрактами підключивши до них свій крипто-гаманець.

Адреси опублікованих контрактів відповідно до мереж з посиланнями на них:

1. Ethereum Sepolia Testnet: https://sepolia.etherscan.io/address/0x90ffcd283787d47cd2b273c1285896906faf90b6
2. Avalanche Fuji Testnet: https://testnet.avascan.info/blockchain/all/address/0x0D51a87550968cE6b1058699309E873c09AB15fe
3. Base Sepolia Testnet: https://sepolia.basescan.org/address/0xff76f96a8507b2994a99190bc8345c1d59d3b0af
4. Arbitrum Sepolia Testnet: https://sepolia.arbiscan.io/address/0xfF76F96A8507b2994a99190bc8345c1D59D3b0AF

Для деплою та верифікації смарт-контрактів було використано фреймворк Hardhat з налаштуваннями специфічними для LayerZero які дозволяють пов'язати контракти (див. Лістинг 4).

Лістинг 4 – Налаштування LayerZero деплою контрактів до мереж:

```
import { EndpointId } from '@layerzerolabs/lz-definitions'
import { ExecutorOptionType } from '@layerzerolabs/lz-v2-utilities'
import { TwoWayConfig, generateConnectionsConfig } from '@layerzerolabs/metadata-tools'
import { OAppEnforcedOption } from '@layerzerolabs/toolbox-hardhat'

import type { OmniPointHardhat } from '@layerzerolabs/toolbox-hardhat'

const sepoliaContract: OmniPointHardhat = {
    eid: EndpointId.SEPOLIA_V2_TESTNET,
    contractName: 'EasyDEXOFT',
}

const arbitrumTestnetContract: OmniPointHardhat = {
    eid: EndpointId.ARBSEP_V2_TESTNET,
    contractName: 'EasyDEXOFT',
}

const baseTestnetContract: OmniPointHardhat = {
    eid: EndpointId.BASESEP_V2_TESTNET,
    contractName: 'EasyDEXOFT',
}

const avalancheTestnetContract: OmniPointHardhat = {
    eid: EndpointId.AVALANCHE_V2_TESTNET,
    contractName: 'EasyDEXOFT',
```

```
}

// To connect all the above chains to each other, we need the following pathways:
// Sepolia <-> Arbitrum Testnet
// Sepolia <-> Avalanche Fuji Testnet
// Sepolia <-> Base Testnet
// Base Testnet <-> Arbitrum Testnet

// For this example's simplicity, we will use the same enforced options values for sending to all chains
// For production, you should ensure `gas` is set to the correct value through profiling the gas usage of calling OFT._lzReceive(...) on the destination chain
// To learn more, read https://docs.layerzero.network/v2/concepts/applications/oapp-standard#execution-options-and-enforced-settings
const EVM_ENFORCED_OPTIONS: OAppEnforcedOption[] = [
    {
        msgType: 1,
        optionType: ExecutorOptionType.LZ_RECEIVE,
        gas: 80000,
        value: 0,
    },
]

// With the config generator, pathways declared are automatically bidirectional
// i.e. if you declare A,B there's no need to declare B,A
const pathways: TwoWayConfig[] = [
    [
```

```
            sepoliaContract, // Chain A contract
            arbitrumTestnetContract, // Chain B contract
            [['LayerZero Labs'], []], // [ requiredDVN[], [ optionalDVN[], threshold ] ]
            [1, 1], // [A to B confirmations, B to A confirmations]
            [EVM_ENFORCED_OPTIONS, EVM_ENFORCED_OPTIONS], // Chain B enforcedOptions, Chain A enforcedOptions
        ],
        [
            sepoliaContract,
            avalancheTestnetContract,
            [['LayerZero Labs'], []],
            [1, 1],
            [EVM_ENFORCED_OPTIONS, EVM_ENFORCED_OPTIONS],
        ],
        [
            sepoliaContract,
            baseTestnetContract,
            [['LayerZero Labs'], []],
            [1, 1],
            [EVM_ENFORCED_OPTIONS, EVM_ENFORCED_OPTIONS],
        ],
        [
            arbitrumTestnetContract,
            baseTestnetContract,
            [['LayerZero Labs'], []],
            [1, 1],
            [EVM_ENFORCED_OPTIONS, EVM_ENFORCED_OPTIONS],
        ],
    ]
```

```
    export default async function () {
        // Generate the connections config based on the
pathways
        const connections = await
generateConnectionsConfig(pathways)
        return {
            contracts: [
                { contract: sepoliaContract },
                { contract: avalancheTestnetContract },
                { contract: arbitrumTestnetContract },
                { contract: baseTestnetContract },
            ],
            connections,
        }
    }
```

Спочатку виконується деплой за допомогою команди `pnpm hardhat lz:deploy --tags EasyDEXOFT`, а далі консольная команда `pnpm hardhat lz:oapp:wire --oapp-config layerzero.config.ts` яка виконує ряд транзакцій які пов'язують контракти (див. Рис. 4.1). Аналогчно контракти можна пов'язати за допомогою методу `setPeer(eid,peer)` на самому смарт-контракті.

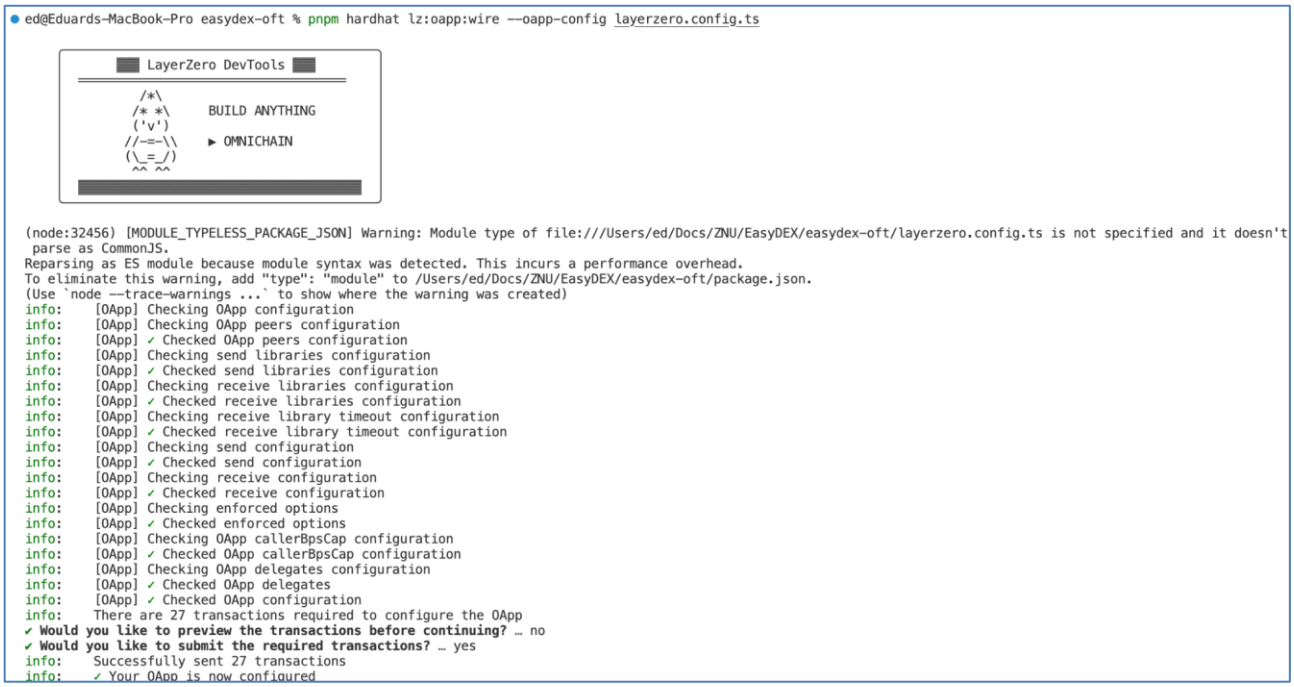


Рисунок 4.1 – Зв’язування задеплоєних контрактів

Після того як контракти були зв'язані, можна перевірити це за допомогою команди `pnpm hardhat lz:oapp:peers:get --oapp-config layerzero.config.ts`, яка виконую метод `peer(eid)` на смарт-контракті і повертає адресу контракту у відповідній мережі (див. Рис. 4.2).



Рисунок 4.2 – Перевірка зв'язків між контрактами

Для підтвердження взаємодії між мережами потрібно скористатися методом `send` смарт-контракту. Відправимо 1 токен з Base Sepolia Testnet до Arbitrum Sepolia Testnet. Для цього нам треба спочатку дізнатися її вартість переказу. Щоб дізнатися вартість, на контракті треба виконати `quoteSend` метод (див. Рис 4.3)

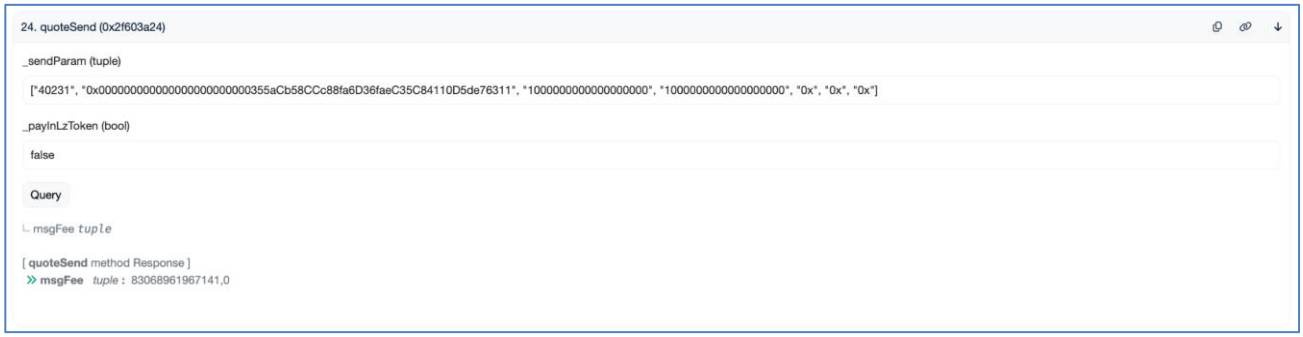


Рисунок 4.3 – Отримання вартості транзакції

Тепер, коли ми маємо цю складову, ми можемо скористатися методом `send` смарт-контракту і переказати 1 токен (див. Рис 4.4). Зверніть увагу що адреса отримувача має туби в `byte32` форматі, а `payableAmount` дорівнювати `nativeFee` в ETH еквіваленті. Після того як ми натиснемо “Write”, відкриється вікно підтвердження та відправки транзакції, яка виконається на початковому ланцюзі. Цю транзакцію можна знайти за адресою:

https://sepolia.basescan.org/tx/0x43c35a23e68068fedfb0867c1f609a6b69462da235b307fd581b2264bdba47cb

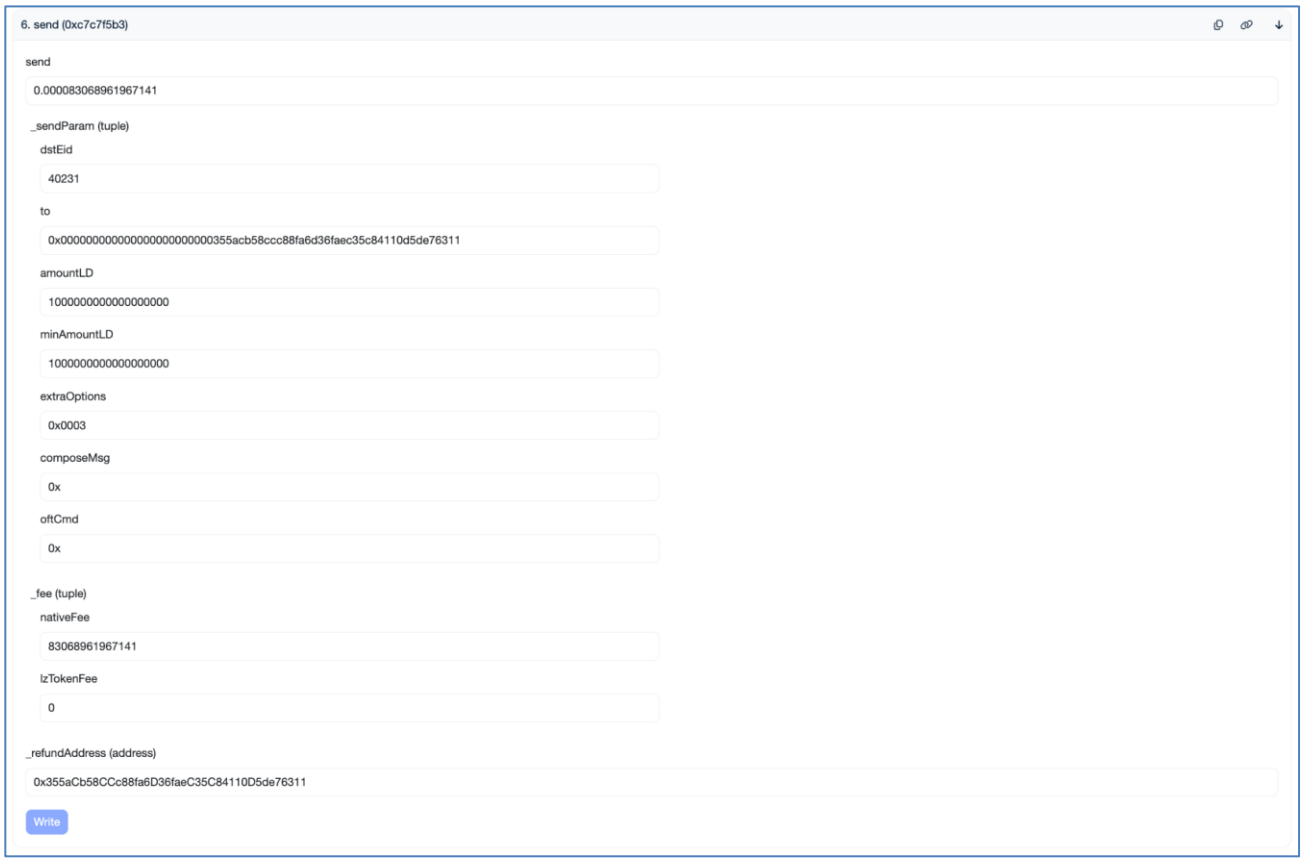


Рисунок 4.4 – Параметри переказу 1 токену між мережами

Виконаний в даному прикладі переказ 1 токену можна знайти на сайті layerzeroscan.com у відкритому доступі за адресою https://testnet.layerzeroscan.com/tx/0x43c35a23e68068fedfb0867c1f609a6b69462da235b307fd581b2264bdba47cb . Далі перевіримо баланс токені у мережі Arbitrum Sepolia Testnet і підтвердимо що тепер маємо на 1 токен більше (див. Рис 4.5)

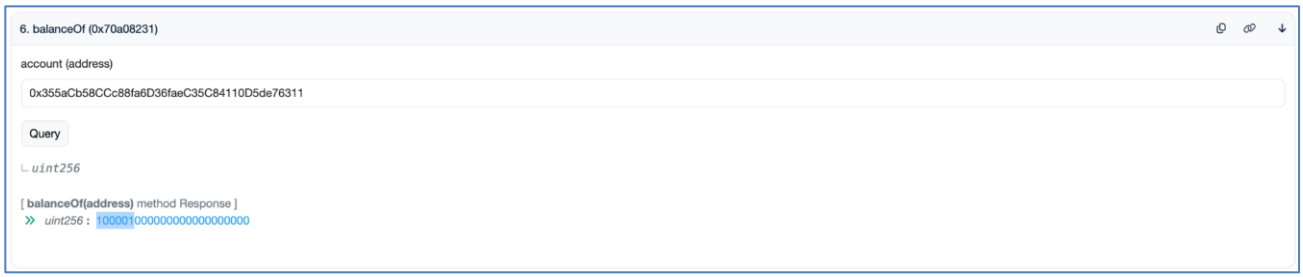


Рисунок 4.5 – Перевірка балансу гаманця отримувача

Відповідно до отриманих результатів, можна підтвердити що передача лівквідності з мережі Base Sepolia Testnet до Arbitrum Sepolia Testnet відбулась успішно і не було порушено умов децентралізації. Отримані контракти пов’язані правильно, працюють в обох мережах і можуть бути використані в застосунку що працює з кількома мережами одночасно.

4.2 Аналіз інтеграції Enso Shortcuts API

Наразі API підтримує більшість популярних мереж серед яких Ethereum (1), Arbitrum (42161), Avalanche (43114), Base (8453), Berachain (80094), Binance (56), Gnosis (100), Hyper (999), Ink (57073), Linea (59144), Optimism (10), Plume (98866), Polygon (137), Sepolia (11155111), Soneium (1868), Sonic (146), Unichain (130), World (480), Zksync (324), Katana (747474), Plasma (9745). На жаль є лише одна тестова мережа – Sepolia [15].

Для роботи з Node.js є бібліотека @ensofinance/sdk яка спрощує процес розробки та інтеграції. Під капотом виконуються запити до API, з яким можна ознайомитися за посиланням https://api.enso.finance/api.

Для того щоб протестувати і продемонструвати як даний API може виконати переказ між мережами було обрано два токени: Tether USD (USDT) з мережі Ethereum (1) і TRON (TRX) з мережі Binance (56). Тепер можна створити запит до API таким чином як вказанов лістингу 5. Оскільки мережі відкриті, то можна легко знайти і підставити реальні адреси гаманців для проведення даного експерименту.

Лістинг 5 – Запит для обміну Ethereum USDT на Binance TRX:

```
curl -X 'POST' \
  'https://api.enso.finance/api/v1/shortcuts/route' \
  -H 'accept: application/json' \
```

```
      -H 'Authorization: Bearer *********' \
      -H 'Content-Type: application/json' \
      -d '{
      "chainId": 1,
      "fromAddress":
"0x8ad292948F71a879c33c6bd798413FB419719714",
      "routingStrategy": "router",
      "receiver":
"0x50f839AC6Ee90664361bA0d70f52409D894F1654",
      "refundReceiver":
"0x50f839AC6Ee90664361bA0d70f52409D894F1654",
      "amountIn": ["1000000"],
      "tokenIn":
["0xdAC17F958D2ee523a2206206994597C13D831ec7"],
      "tokenOut":
["0xCE7de646e7208a4Ef112cb6ed5038FA6cC6b12e3"],
      "destinationChainId": 56
    }'
```

Результатом виконання даного запиту є лише одна закодована транзакція яку залишається лише виконати для того щоб отримати токени в іншій мережі. Приклад відповіді API за запитом з лістингу 5 представлено в лістингу 6.

Лістинг 6 – Відповідь API на запит для обміну Ethereum USDT на Binance TRX:

```
    {
      "amountOut": "3620726",
      "minAmountOut": "3602622",
      "priceImpact": null,
      "gas": "922674",
```

```
"createdAt": 23870602,
"route": [
  {
    "action": "swap",
    "protocol": "odos",
    "tokenIn": [
      "0xdac17f958d2ee523a2206206994597c13d831ec7"
    ],
    "tokenOut": [
      "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48"
    ],
    "chainId": 1
  },
  {
    "action": "bridge",
    "protocol": "stargate",
    "tokenIn": [
      "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48"
    ],
    "tokenOut": [
      "0x8ac76a51cc950d9822d68b83fe1ad97b32cd580d"
    ],
    "sourceChainId": 1,
    "destinationChainId": 56
  },
  {
    "action": "swap",
    "protocol": "odos",
    "tokenIn": [
      "0x8ac76a51cc950d9822d68b83fe1ad97b32cd580d"
    ],
```

```
            "tokenOut": [
              "0xce7de646e7208a4ef112cb6ed5038fa6cc6b12e3"
            ],
            "chainId": 56
          }
        ],
        "bridgingEstimates": [
          {
            "fromChainId": 1,
            "toChainId": 56,
            "token":
"0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48",
            "estimatedSeconds": 180,
            "sendConfirmations": 15,
            "receiveConfirmations": 20,
            "sendBlockTimeSeconds": 12,
            "receiveBlockTimeSeconds": 3,
            "bridge": "stargate",
            "source": "defaults"
          }
        ],
        "tx": {
          "data": "0xf52e33f500000…000000",
          "to": "0xF75584eF6673aD213a685a1B58Cc0330B8eA22Cf",
          "from": "0x8ad292948F71a879c33c6bd798413FB419719714",
          "value": "154563089383766"
        }
      }
```


Як бачимо, cross-chain маршрут буде складатися з 3 кроків:

1. Swap на Ethereum. Обмін USDT на USDC через протокол Odos.
2. Bridge з Ethereum в BSC. Міст USDC з Ethereum на Binance Smart Chain через Stargate.
3. Swap на BSC. Обмін USDC на TRON через протокол Odos

Зверніть увагу, що API повертає очікувану суму на виході в полі amountOut, оцінку вартості переказу в полі gas, та те як швидко має пройти переказ в `bridgingEstimates` – приблизно 180 секунд.

4.3 Аналіз продуктивності Layer 2 рішень у порівнянні з Layer 1

Для аналізу продуктивності Layer 2 рішень у порівнянні з Layer 1 використаємо смарт-контракт розроблений за допомогою LayerZero, та виконаємо транзакції для трансферу токенів.

У нас вже підключено дві Layer 2 мережі: Base Sepolia Testnet та Arbitrum Sepolia Testnet і тому можна використати їх і трансфер 1 токену який було виконано раніше, для прикладу (див. Рис 4.6). Для аналізу використовуються дані з Layerzeroscan, оскільки можно чітко зрозуміти скільки часу зайняв трансфер, та чи не виникли проблеми на шляху з однієї мережі до іншої.

Для трансферу між Layer 1 мережами було обрано Ethereum Sepolia Testnet та Avalanche Fuji Testnet. В рамках експерименту було також виконано переказ між цими двома мережами 1 токену (див. Рис 4.7).

Виходячи з даних цих даних транзакцій можна зробити висновки що переказ Layer 2 to Layer 2 виконується набагато швидше за рахунок архітектури цих мереж (29 секунд проти 1 хв 59 секунд).

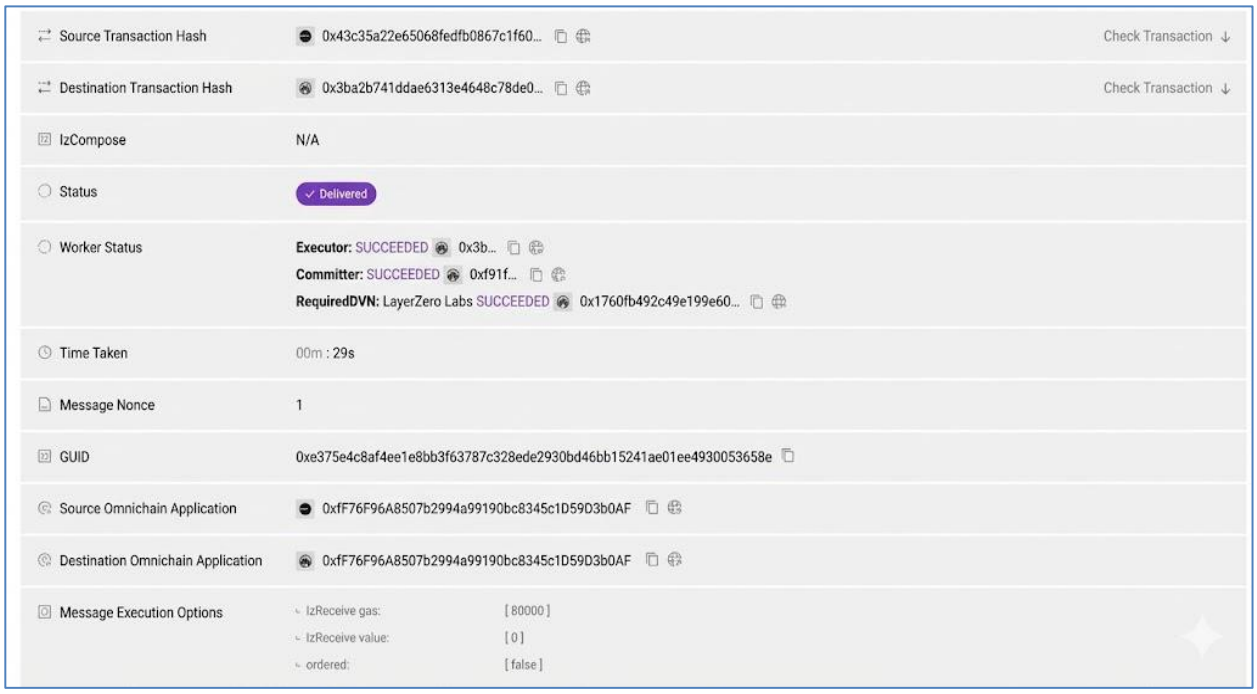


Рисунок 4.6 – Трансфер 1 токену з Base Sepolia Testnet до Arbitrum Sepolia Testnet

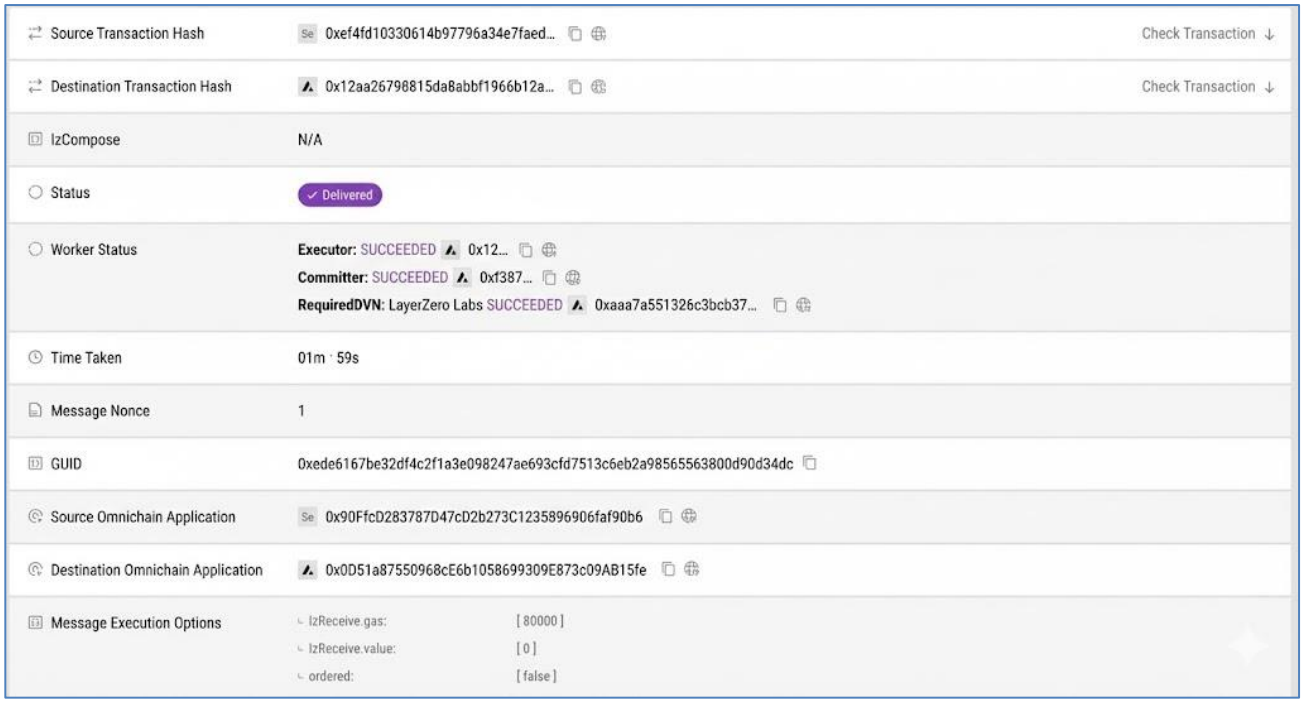


Рисунок 4.7 – Трансфер 1 токену з Ethereum Sepolia Testnet до Avalanche Fuji Testnet

Тепер давайте оцінимо вартість відправки токенів в початкових мережах. Для цього проведемо аналіз цих транзакцій (див. Рис. 4.8 та 4.9). Перерахуємо вартість, використовуючи ці реальні дані з Testnet, і спроектуємо їх на Mainnet.

Припустимо, що середній Gwei на Ethereum Mainnet = 20 Gwei. Зазвичай Ethereum Mainnet потрібно ~150,000 одиниць газу для такої операції, тому приблизна ціна транзакції буде складати 150,000 * 20 Gwei = 0.003 ETH, що становить (при ETH = $3500): ~$10.50$.

На Base Mainnet середній Gwei становить 0.1 Gwei (завдяки Blob transactions). Відповідно фінальна ціна переказу складатиме приблизно $0.01 – $0.05.

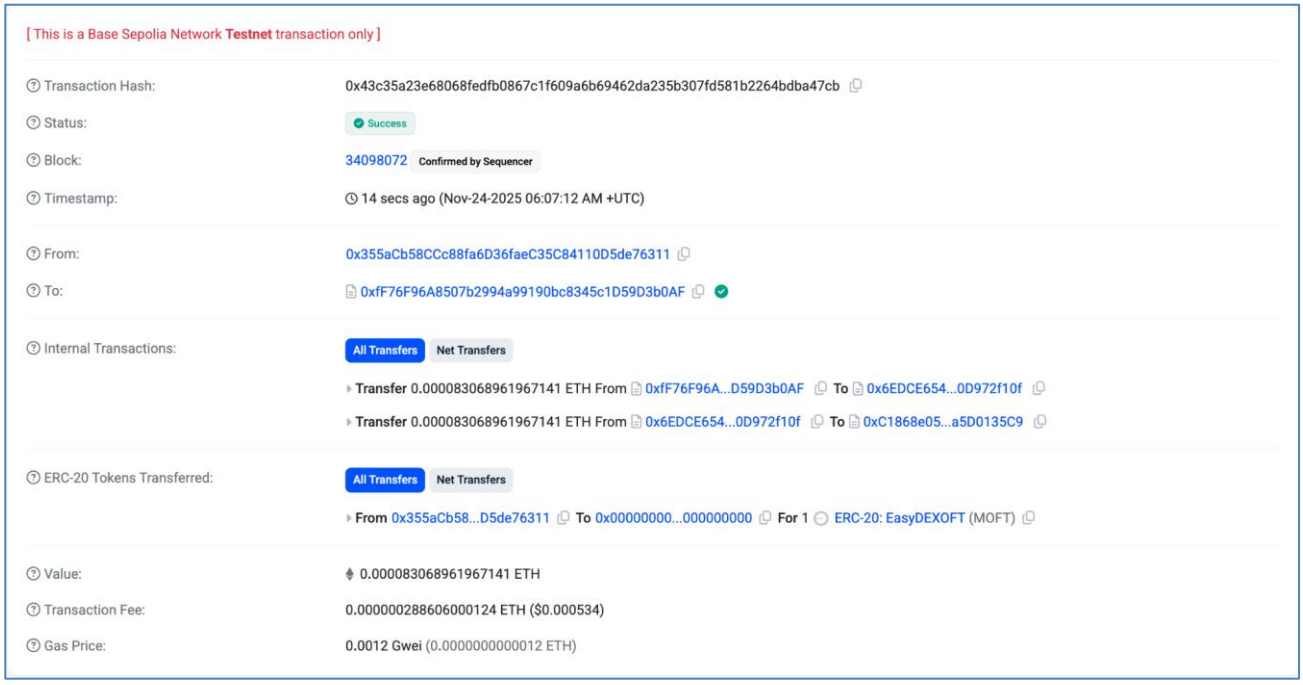


Рисунок 4.8 – Початкова транзакція з мережі Base Sepolia

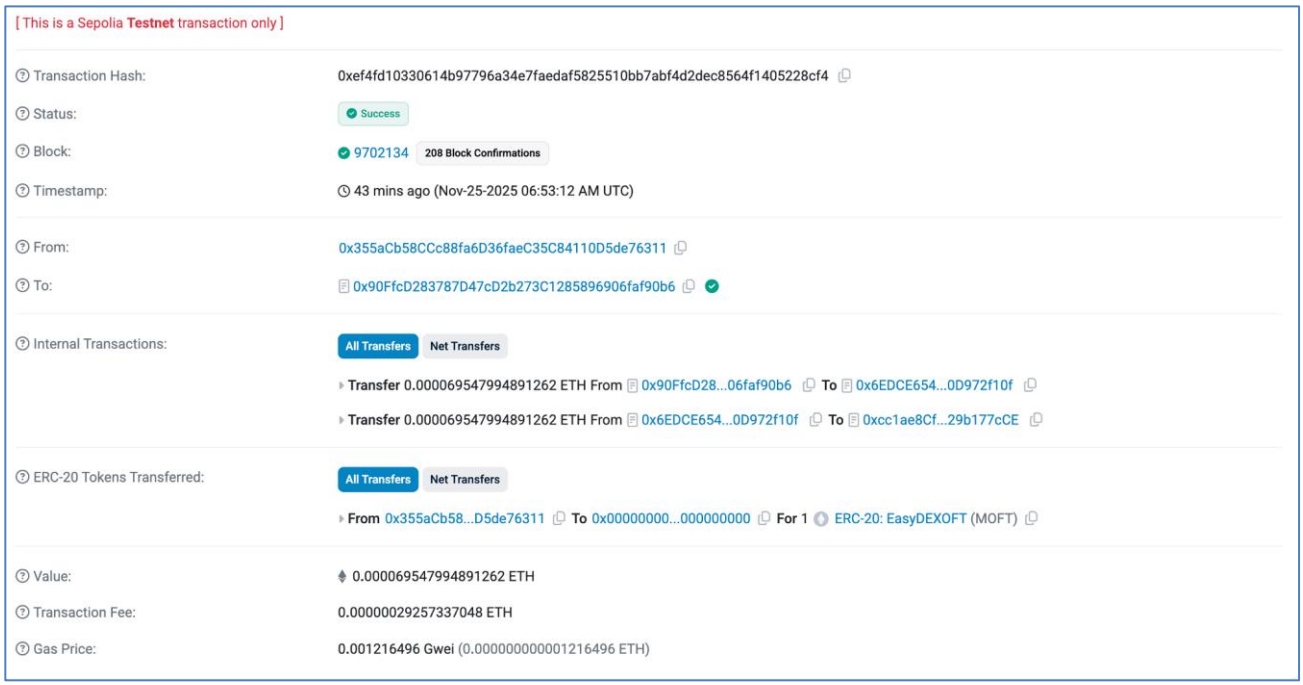


Рисунок 4.9 – Початкова транзакція з мережі Ethereum Sepolia

Варто також оцінити процес передачі токену з Layer 1 мережі до Layer 2 мережі. Для цого перекажемо 1 токен з Ethereum Sepolia Testnet до Arbitrum Sepolia Testnet (див. Рис. 4.10).

Деталі транзакції можна знайти за посиланням: https://testnet.layerzeroscan.com/tx/0x8f679e94ce4c3ec1b884048e47ce08b3f3fe69602236d234a804f50784aee35a.

Як бачимо, за рахунок різниці в механізмі консенсусу, трансфер зайняв більше часу, оскільки на мережі першого рівня транзакція мала підтверджуватися довше.

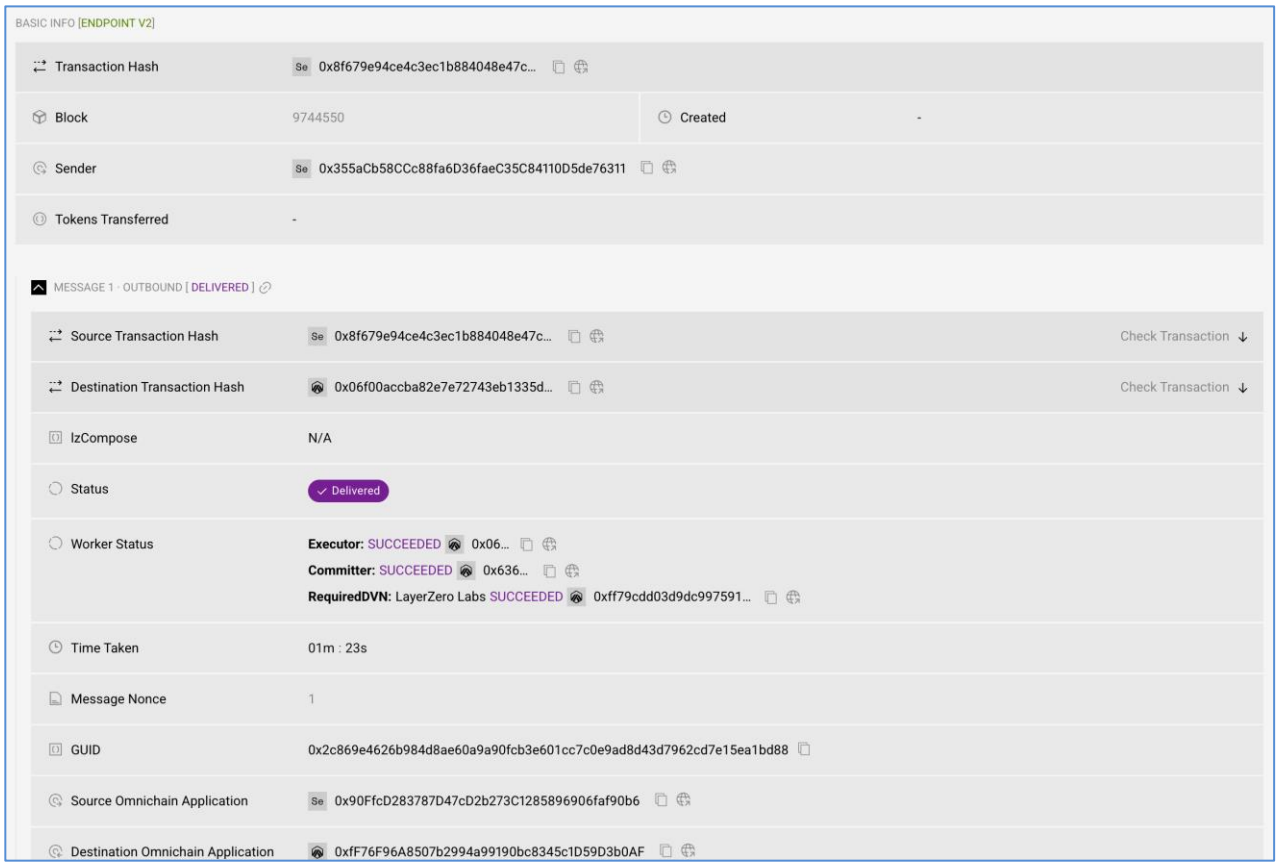


Рисунок 4.10 – Трансфер 1 токену з Ethereum Sepolia Testnet до Arbitrum Sepolia Testnet

Вартість транзакції в цьому випадку буде вищою, оскільки на мережі першого рівня має виконатися транзакція яка ініціює переказ, але можливість скористатися багатонаправленим міжмережевим трансфером ліквідності надає користувачу більше свободи вибору в тому як керувати активами.

4.4 Економічний аналіз розробленого застосунку

З точки зору вибору блокчейн-мереж для роботи з додатком рекомендація і акцент робиться на Layer 2 рішення, що знижує бар'єр входу для користувачів. Layer 2 мережі дозволяють досягти такого досвіду користувача, який наближається до Web 2.0. В прикладах вище (див. Рис 4.8) вартість транзакції на

Base Testnet склала 0.00000288 ETH, що на Mainnet, завдяки Blob-транзакціям, залишається в межах центів. Використовуються Rollups. Транзакції виконуються поза основним ланцюгом і лише їхній криптографічний "підсумок" записується на Layer 1.

Використання Enso Shortcuts API для складних маршрутів. Тут економіка працює за рахунок оптимізації шляху. У прикладі (див. Розділ 4.2) розглянено обмін USDT (Ethereum) на TRX (BSC). Цей шлях включає 3 транзакції. Якщо виконувати обмін вручну, то маршрут би довелося будувати самостійно, а також враховувати коливання курсу криптовалют. Enso Shortcuts API повертає єдиний маршрут (route), який об'єднує всі кроки, знаходить найвигідніший курс серед багатьох протоколів і замість трьох окремих ініціацій транзакцій користувачем, виконується одна складна, що зменшує накладні витрати на газ.

Обрана архітектура на основі лямбда-функцій є економічно обгрунтованим підходом для стартапу, бо оплата здійснюється лише за час виконання коду в хмарі і витрати будуть залежати лише від популярності застосунку. Крім того хмара автоматично виділяє ресурси під час пікових навантажень і зменшує їх, коли активність падає. Використання однієї мови програмування також знижує знижує витрати на розробку, оскільки команда Fullstack розробників може підтримувати весь проект.

4.5 Підсумки розділу 4

Аналіз підтверджує надійність та швидкість крос-чейн рішень, таких як LayerZero та Enso API, а також перевагу використання Layer 2 мереж над Layer 1 з точки зору швидкості та вартості переказу ліквідності.

Контракт розроблений за допомогою LayerZero показав надійність під час виконання передачі токенів як між Layer 2 та і між Layer 1 мережами. Відповідно

це рішення підходить для розробки застосунків що працюють з кількома блокчейн мережами.

Enso Shortcuts API підходить для будування складних маршрутів, але не має високої підтримки Testnet, а на Mainnet підтримує перекази основних найбільш відомих токенів.

Переказ між мережами Layer 2, є значно швидшим (29 секунд) у порівнянні з переказом між мережами Layer 1 (1 хвилина 59 секунд), що пояснюється архітектурними особливостями L2 рішень.

Поекція вартості транзакцій на Mainnet показує перевагу L2 рішень, таких як Base, де вартість переказу токенів може становити лише $0.01 – $0.05, порівняно з приблизно $10.50 на Ethereum Mainnet для аналогічної операції.

ВИСНОВКИ

За даними досліджень проведених під час виконання кваліфікаційної роботи було отримано такі результати:

1. Досліджено основні механізми консенсусу які існують в блокчейн мережах, та визначено їх переваги та недоліки. Поведено аналіз

2. Визначені основні проблеми з якими стикаються користувачі під час роботи з блокченами.

3. В результаті проведеного аналізу обмежень блокчейнів першого рівня було окреслено проблему фрагментації ліквідності та запропоновано Layer Zero, як безпечне та надійне рішення для трансферу власних токенів з однієї мережі до іншої. В якості підтвердження було розроблено та задеплоєно смарт-контракти з тестовим токеном до кількох мереж, та протестовано процес трансферу між ними.

4. Досліджено інтеграцію з Enso Shortcuts API, яка дозволяє отримати оптимальний шлях передачі існуючих токенів між мережами в межах однієї транзакції.

5. Розроблено мобільний застосунок з єдиним інтерфейсом і серверлесс архітектурою, що працює одночасно з кількома блокчейнами і на обох платформах – Android та iOS. Запропонована архітектура: React Native -> GraphQL -> Serverless -> PostgreSQL, довела свою працездатність, та простоту розгортання і роботи в локальному середовищі.

6. Економічне моделювання показало, що застосування комбінації LayerZero з Enso Shortcuts замість мостів та централізованих і частково централізованих рішень дозволяє знизити загальні витрати користувачів на операції в середньому більше ніж на половину.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fu, Xiang, Huaimin Wang and Peichang Shi. A survey of Blockchain consensus algorithms: mechanism, design and applications. Science China Information Sciences 64 (2020): n. pag. DOI: http://doi.org/10.1007/s11432-019-2790-1.

2. Zheng, Zibin, Shaoan Xie, Hongning Dai, Xiangping Chen and Huaimin Wang. “An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends.” 2017 IEEE International Congress on Big Data (BigData Congress) (2017): 557-564. DOI: https://doi.org/10.1109/BIGDATACONGRESS.2017.85.

3. Xiao, Yang, Ning Zhang, Wenjing Lou and Y. Thomas Hou. A Survey of Distributed Consensus Protocols for Blockchain Networks. IEEE Communications Surveys & Tutorials 22 (2019): 1432-1465. DOI: https://doi.org/10.1109/COMST.2020.2969706.

4. Layer 1 Blockchain and scaling solutions (Part 15- Blockchain Series). URL: https://medium.com/techskill-brew/layer-1-blockchain-and-scaling-solutions-part-15-blockchain-series-6c20e94ef96d (дата звернення: 20.01.2025)

5. Notland, Jakob Svennevik, Jinguye Li, Mariusz Nowostawski and Peter Halland Haro. SoK: Cross-Chain Bridging Architectural Design Flaws and Mitigations. ArXiv abs/2403.00405 (2024): n. pag. DOI: https://doi.org/10.48550/arXiv.2403.00405.

6. Ukeje Chukwuemeriwo Goodness. Now, You Understand Blockchain Interoperability. URL: https://dev.to/goodylili/-now-you-understand-blockchain-interoperability-de (дата звернення: 20.01.2025).

7. Sguanci, Cosimo, Roberto Spatafora and Andrea Vergani. Layer 2 Blockchain Scaling: a Survey. ArXiv abs/2107.10881 (2021): n. pag. DOI: https://doi.org/10.48550/arXiv.2107.10881.

8. Mastering Layer 2 Solutions: A Comprehensive Guide for Blockchain Developers. URL: https://www.infuy.com/blog/mastering-layer-2-solutions-a-comprehensive-guide-for-blockchain-developers/ (дата звернення: 20.01.2025)

9. Layer 2 scaling solutions: A beginner's guide to blockchain scalability and efficiency. URL: https://www.starknet.io/blog/layer-2-scaling-solutions (дата звернення: 20.01.2025)

10. Aditya Saraf, Kelvin Koh. Cross-Chain Interoperability: LayerZero. URL: https://www.spartangroup.io/insights/cross-chain-interoperability-layerzero (дата звернення: 20.01.2025).

11. Zarick, Ryan, Bryan Pellegrino, Isaac Zhang, Thomas Kim and Caleb Banister. LayerZero. ArXiv abs/2312.09118 (2023): n. pag. DOI: https://doi.org/10.48550/arXiv.2312.09118.

12. Zarick, Ryan, Bryan Pellegrino and Caleb Banister. LayerZero: Trustless Omnichain Interoperability Protocol. ArXiv abs/2110.13871 (2021): n. pag. DOI: https://doi.org/10.48550/arXiv.2110.13871.

13. LayerZero Documentation. URL: https://docs.layerzero.network/v2 (дата звернення: 20.01.2025).

14. Shortcut to the Enso Network. URL: https://blog.enso.build/shortcut-to-the-enso-network/ (дата звернення: 16.06.2025).

15. Enso Developer Documentation. URL: https://docs.enso.build/home (дата звернення: 16.06.2025).

16. Reown Developer Documentation. URL: https://docs.reown.com/overview (дата звернення: 16.06.2025).

17. LocalStack for AWS Docs. URL: https://docs.localstack.cloud/aws/ (дата звернення: 16.10.2025).

18. Node.js v24.11.1 documentation. URL: https://nodejs.org/docs/latest-v24.x/api/index.html (дата звернення: 20.11.2025).

19. Docker Compose. URL: https://docs.docker.com/compose/ (дата звернення: 10.09.2025).

20. PostgreSQL 17 Documentation. URL: https://www.postgresql.org/docs/17/index.html (дата звернення: 16.09.2025).

21. Develop with Redis. URL: https://redis.io/docs/latest/develop/ (дата звернення: 16.09.2025).

22. React Native Guides. URL: https://reactnative.dev/docs/getting-started (дата звернення: 20.09.2025).

23. GraphQL Learn. URL: https://graphql.org/learn/ (дата звернення: 20.09.2025).

24. Serverless Framework Documentation. URL: https://www.serverless.com/framework/docs (дата звернення: 05.10.2025).

25. What Is Etherscan and How to Use It. URL: https://blog.whitebit.com/en/what-is-etherscan-and-how-to-use-it/ (дата звернення: 05.10.2025).

26. MetaMask: A Detailed Guide for Beginners. URL: https://incrypted.com/en/metamask-a-detailed-guide-for-beginners/ (дата звернення: 10.10.2025).

27. Інженерні інновації та розбудова національної економіки: матеріали II Міжнародної науково-практичної конференції Інженерного навчально-наукового інституту ім. Ю. М. Потебні (15–16 травня 2025 року, м. Запоріжжя) / наук. ред. Н. Г. Метеленко ; Інженерний навчально-науковий інститут ім. Ю. М. Потебні Запорізького національного університету. – Одеса : Видавничий дім «Гельветика», 2025. – 960 с.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Якимцов Едуард Анатолійович, студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти m1214-12@stu.zsea.edu.ua,

- підтверджую, що написана мною кваліфікаційна робота на тему **“Використання LayerZero та Layer 2 рішень в розробці застосунків, що працюють з блокчейном”** відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України “Про освіту”, зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 01.12.2025 Підпис _____ Якимцов Едуард Анатолійович
(студент)

Дата 01.12.2025 Підпис _____ Безверхий Анатолій Ігорович
(науковий керівник)