Міністерство освіти і науки України

Луцький національний технічний університет

(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій

(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки

(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

ДЕЦЕНТРАЛІЗОВАНА БЛОКЧЕЙН-СИСТЕМА НА БАЗІ МОБІЛЬНОГО ЗАСТОСУНКУ З ВИКОРИСТАННЯМ ШІ

DECENTRALIZED BLOCKCHAIN SYSTEM BASED ON A MOBILE APPLICATION USING AI

спеціальність 123 Комп'ютерна інженерія

(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія

(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Кулакевич Олег Русланович

(підпис)

Керівник:
к.т.н., доцент
Кайдик Олег Леонтійович

(підпис)

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

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. Терлецький
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кулакевичу Олегу Руслановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Децентралізована блокчейн-система на базі мобільного застосунку з використанням ШІ*

Керівник роботи *к.т.н., доцент Кайдик Олег Леонтійович*

затверджені наказом закладу вищої освіти від «17» червня 2025 року №о 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Теоретичні основи роботи з блокчейн-мережею Sui
Програмна реалізація Sui-гаманця
Реалізація та дослідження механізмів транзакцій та голосування
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Принцип роботи блокчейн системи
Обробка транзакцій SUI
Мобільний гаманець на Swift
Прототип електронного голосування на базі блокчейну

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Теоретичні основи роботи з блокчейн-мережею Sui	*Кайдик О. Л., доцент*		
Програмна реалізація Sui-гаманця	*Кайдик О. Л., доцент*		
Реалізація та дослідження механізмів транзакцій та голосування	*Кайдик О. Л., доцент*		
Нормоконтроль	*Багнюк Н. В., доцент*		
Гарант ОП	*Гринюк С. В., доцент*		
Показник запозичень тексту	_____**%**		
Академічна доброчесність	*Міскевич О. І., ст. викладач*		

7. Дата видачі завдання 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури, аналіз сучасних підходів до блокчейн-розробки, дослідження архітектури Sui та існуючих гаманців*	до 01.08.2025 р.	
2.	*Теоретичні основи побудови криптографічних гаманців, механізмів транзакцій, RPC*	до 02.09.2025 р	
3.	*Вибір тех. бази: Swift, SuiSwift SDK, Move, формування архітектури застосунку*	до 15.09.2025 р.	
4.	*Реалізація застосунку: створення та відновлення гаманця, обробка транзакцій, історія операцій, faucet*	до 05.10.2025 р.	
5.	*Висновки та пропозиції*	До 25.10.2025 р	
6.	*Формування списку використаних джерел*	До 27.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 05.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 11.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Кулакевич О.Р.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Кайдик О.Л.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Кулакевич О. Р. Децентралізована блокчейн-система на базі мобільного застосунку з використанням ШІ. Рукопис.

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

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

У першому розділі виконано аналіз сучасних підходів до побудови блокчейн-систем, особливостей архітектури Sui, принципів роботи криптографічних гаманців та механізмів транзакцій. Розглянуто RPC-взаємодію, локальні блокчейн-мережі, принципи консенсусу та можливості застосування Sui у мобільних системах.

У другому розділі обґрунтовано вибір технологічної бази для реалізації застосунку: мови Swift, фреймворків UIKit/SwiftUI, пакета SuiSwift, локальної тестової мережі Sui та інструментів для роботи з транзакціями. Розроблено архітектуру мобільного застосунку, включно зі створенням і відновленням гаманця, обробкою балансу, підключенням до локального RPC та взаємодією з faucet-сервісом.

У третьому розділі реалізовано прототип мобільної системи, що забезпечує можливість формування та відправлення транзакцій, перегляду історії операцій, а також виконання on-chain голосування. Проведено функціональне тестування, досліджено роботу локальної Sui-мережі та проаналізовано ефективність запропонованих рішень.

Ключові слова: Sui, блокчейн, Swift, iOS, транзакції, гаманець, RPC, Move, голосування.

ANNOTATION

Kulakevych O. Decentralized blockchain system based on a mobile application using AI.

Qualification work of the master of the specialty «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

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

The first chapter provides an analysis of modern blockchain architectures, the structure of the Sui network, principles of cryptographic wallet generation, transaction processing mechanisms, and RPC communication models. Special attention is paid to local blockchain networks, consensus methods, and the applicability of Sui in mobile systems.

The second chapter substantiates the choice of technologies for the project implementation, including Swift, UIKit/SwiftUI, the SuiSwift SDK, local Sui test network tools, and transaction-handling mechanisms. The architectural model of the mobile application was developed, featuring wallet creation and restoration, balance processing, RPC configuration, and interaction with the faucet service.

The third chapter presents the implementation of a functional prototype that enables sending transactions, viewing transaction history, and performing on-chain voting. Functional testing was conducted, behaviour of the local Sui network was analysed, and the system’s performance and reliability were evaluated.

Keywords: Sui, blockchain, Swift, iOS, transactions, wallet, RPC, Move, voting.

ЗМІСТ

ВСТУП

Стрімкий розвиток мобільних технологій, децентралізованих систем та блокчейн-інфраструктур відкриває нові можливості для створення безпечних, автономних і масштабованих цифрових сервісів. З появою високопродуктивних блокчейнів нового покоління, таких як Sui, зростає потреба у мобільних рішеннях, що дозволяють користувачам безпосередньо взаємодіяти з децентралізованими застосунками, керувати цифровими активами та брати участь у голосуваннях чи інших on-chain процесах. Це формує актуальну задачу розроблення інтелектуальних мобільних систем, здатних забезпечити зручний, безпечний та швидкий доступ до можливостей блокчейн-мереж.

Традиційні мобільні гаманці та додатки для роботи з блокчейном зазвичай обмежуються базовими функціями перегляду балансу та здійснення транзакцій, тоді як сучасні користувацькі сценарії вимагають більшої гнучкості. Існує потреба у створенні системи, яка поєднує локальну роботу з тестовою мережею, автоматичне формування транзакцій, управління бюджетом газу, перегляд історії операцій, а також інтеграцію механізмів голосування на основі Move-модулів. Це формує актуальність розроблення мобільного застосунку, здатного працювати як із локальною блокчейн-інфраструктурою, так і з віддаленими вузлами.

Метою кваліфікаційної роботи магістра є розробка та дослідження мобільного застосунку для iOS із використанням блокчейна Sui, що забезпечує створення та відновлення гаманця, виконання транзакцій, роботу з локальною мережею, перегляд історії операцій та реалізацію on-chain голосування.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– здійснити аналіз сучасних блокчейн-технологій і особливостей архітектури Sui;

– дослідити методи обробки транзакцій, управління цифровими активами та механізми RPC-взаємодії;

– розробити архітектуру мобільного застосунку з підтримкою створення гаманця, отримання тестових токенів, відправлення транзакцій та обробки газу;

– реалізувати модуль перегляду історії операцій із визначенням вхідних та вихідних трансакцій;

– інтегрувати механізм on-chain голосування на основі Move-контрактів та провести тестування функціональності в локальному середовищі Sui.

Об’єктом дослідження є процес взаємодії мобільного застосунку з блокчейн-інфраструктурою Sui.

Предметом дослідження є методи побудови мобільної системи управління транзакціями, гаманцями та голосуванням у високопродуктивній блокчейн-мережі.

Наукова новизна роботи полягає у поєднанні локальної тестової мережі Sui, автоматичного управління транзакціями та реалізації on-chain механізмів голосування всередині нативного iOS-застосунку. У роботі запропоновано підхід до комбінованої обробки транзакцій, що включає автоматичний підбір бюджету газу та динамічний аналіз історії операцій.

Практичне значення роботи полягає у створенні прототипу мобільного застосунку, який може бути використаний для тестування та демонстрації можливостей блокчейна Sui, розроблення децентралізованих сервісів, управління цифровими активами та впровадження on-chain голосувань у прикладних системах.

Апробація роботи на Міжнародній науково-практичній конференції молодих вчених та студентів «Програмне та апаратне забезпечення в інформаційних технологіях» (м. Луцьк, 6 травня 2025 р), де було висвітлено основні положення кваліфікаційної роботи [1].

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОБОТИ З БЛОКЧЕЙН-МЕРЕЖЕЮ SUI

1.1 Розвиток блокчейн-технологій у контексті сучасних інформаційних систем

Інтенсивне зростання обсягів даних, ускладнення цифрових сервісів і потреба у підвищенні рівня безпеки в інформаційних системах привели до формування нових парадигм зберігання та обробки інформації. У цьому контексті блокчейн став однією з найзначущіших технологічних інновацій останнього десятиліття. Його фундаментальні властивості – незмінність даних, прозорість транзакцій, розподіленість та криптографічний захист – забезпечили появу нової моделі довіри, у якій зникла необхідність у централізованому контролері або посереднику [2].

Спочатку блокчейн розроблявся як технологія для підтримки криптовалюти Bitcoin, тобто виконання простої функції – реєстрації фінансових операцій у децентралізованому реєстрі. Однак уже на ранніх етапах стало зрозуміло, що потенціал блокчейну значно ширший. Поява Ethereum засвідчила перехід від «блокчейн 1.0» до «блокчейн 2.0», коли мережі почали підтримувати смартконтракти – програмовані логічні модулі, що виконуються у довіреному середовищі й забезпечують автоматизацію бізнес-процесів без участі центрального сервера [3] на відміну від класичної технології «клієнт-сервер», що подано на рисунку 1.1. Саме смартконтракти дали поштовх до розвитку децентралізованих застосунків (dApps), токенізації активів і появи цілих екосистем цифрової економіки.

Сучасний етап еволюції блокчейнів пов’язаний із платформами третього покоління, орієнтованими на масштабованість, низькі затримки, підтримку паралельного виконання транзакцій і взаємодію з великою кількістю цифрових об’єктів. У цих мережах акцент зміщується з обробки фінансових операцій на забезпечення основи для складних цифрових систем, включно з логістичними платформами, IoT-інфраструктурою, реєстрами даних, цифровою

ідентифікацією та інтелектуальними сервісами. Одним із найперспективніших таких рішень є Sui – блокчейн нового покоління, що використовує об’єктно-орієнтовану модель даних та оптимізований механізм паралельного виконання транзакцій [4].

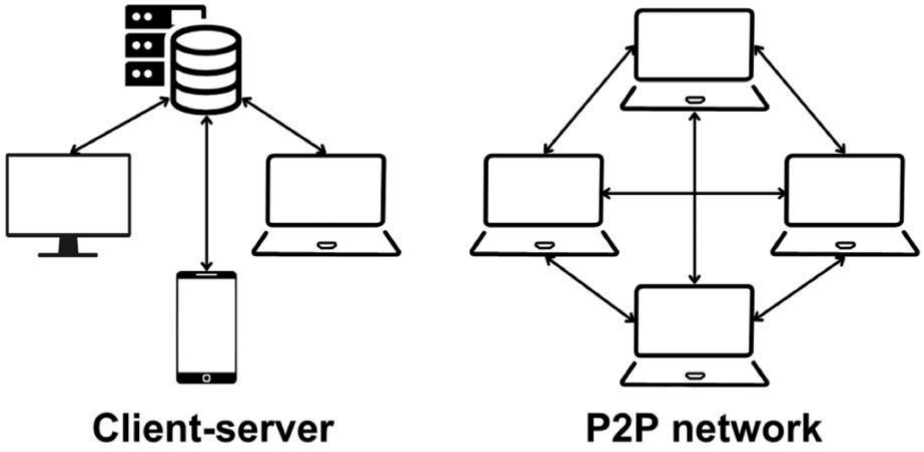

Рисунок 1.1 – Різниця між клієнт-сервер та P2P мережею [2]

Інноваційність підходу Sui полягає в тому, що замість класичної «акаунтної» моделі, характерної для Ethereum, застосовується модель цифрових об’єктів, кожен з яких має унікальний ідентифікатор, властивості та правила взаємодії. Така архітектура дозволяє значно підвищити пропускну здатність мережі та здійснювати обробку транзакцій без необхідності глобальної синхронізації стану. Завдяки цьому Sui здатен обслуговувати велику кількість користувачів у реальному часі та забезпечувати стабільно низькі затримки, що є критично важливим для масових застосунків.

Сучасні інформаційні системи дедалі частіше використовують блокчейн не лише як механізм забезпечення транспарентності, але і як універсальний інфраструктурний рівень для побудови сервісів, у яких гарантії достовірності даних є пріоритетними. До таких сфер належать ланцюги постачання, медичні інформаційні системи, електронні реєстри, ідентифікаційні системи, системи електронного голосування, децентралізовані фінанси (DeFi), NFT-платформи та

різноманітні Web3-сервіси. У кожному з цих напрямів важливу роль відіграє специфіка архітектури конкретного блокчейну, його здатність до масштабування, швидкість обробки транзакцій, безпека смартконтрактів та зручність для розробників, детальніше з основними компонентами архітектури блокчейну можна ознайомитися у таблиці 1.1. В таблиці 1.2 наведено порівняння блокчейн-архітектур.

Таблиця 1.1 – Компоненти архітектури блокчейну [4-7]

Компонент	Опис
Блоки	Основні одиниці даних, які містять трансакції та хеші попередніх блоків
Розподілений реєстр	Децентралізована база даних, копії якої зберігаються в усіх вузлах мережі
Вузли (ноди)	Комп'ютери-учасники мережі які зберігають копії блокчейну, перевіряють трансакції та додають нові блоки
Алгоритми консенсусу	Механізми досягнення «згоди» між вузлами щодо дійсності трансакцій (Proof of Work, Proof of Stake тощо)
Хеш-функції	Криптографічні алгоритми для створення унікального ідентифікатора кожного окремого блоку
Цифрові підписи	Засоби підтвердження справжності трансакцій та ідентифікації учасників
Смарт контракти	Програмні алгоритми які автоматично виконують умови угод (опціонально, залежить від типу блокчейну)

Таким чином, блокчейн сьогодні перестає бути технологією, пов'язаною виключно з криптовалютою, і трансформується у гнучкий та надійний інфраструктурний механізм для побудови розподілених цифрових систем. Особливе місце серед сучасних рішень займає Sui, який поєднує високу продуктивність, інноваційну модель даних та зручність розробки завдяки Move – мові програмування, орієнтованій на безпеку та формальну верифікацію.

Розуміння особливостей таких платформ є важливим етапом у дослідженні можливостей їх використання в сучасних проєктах та інтеграції в інформаційні системи нового покоління.

Таблиця 1.2 – Види та відмінності блокчейн-архітектур [5]

Властивість	Публічний блокчейн	Консорціум блокчейн	Приватний блокчейн
Прийняття рішень	Усі користувачі	Певна виокремлена кількість користувачів	Користувачі у рамках однієї організації
Інформація про ланцюги	Відкрита	Відкрита або прихована	Відкрита або прихована
Стабільність	Майже неможливо порушити	Можливо порушити	Можливо порушити
Використання ресурсів	Низьке	Високе	Високе
Централізація	Відсутня	Часткова	Існує
Створення нової задачі	Не вимагає дозволів	Потрібен дозвіл	Потрібен дозвіл

Розуміння особливостей таких платформ є важливим етапом у дослідженні можливостей їх використання в сучасних проєктах та інтеграції в інформаційні системи нового покоління.

1.2 Моделі представлення стану в блокчейн-системах

Спосіб організації стану у блокчейн-системі має вирішальний вплив на те, як обробляються транзакції, яким є рівень масштабованості мережі та які типи застосунків можуть на ній функціонувати. Фактично модель стану визначає архітектурну «філософію» блокчейну й формує правила взаємодії користувачів, смартконтрактів і мережевих вузлів.

Перші блокчейни, зокрема Bitcoin, використовували модель невитрачених виходів транзакцій – UTXO (Unspent Transaction Output). У цій моделі стан представлений як множина виходів попередніх транзакцій, які ще не були витрачені. Кожна транзакція створює нові виходи та «списує» старі, формуючи

чіткий причинно-наслідковий ланцюг, даний підхід деталізовано подано на рисунках 1.2 та 1.3.

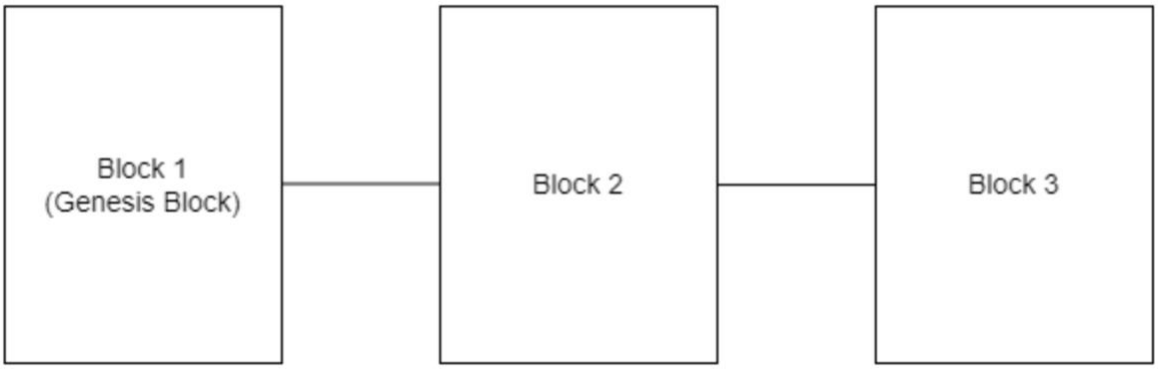

Рисунок 1.2 – Ланцюжок криптографічних блоків [5]

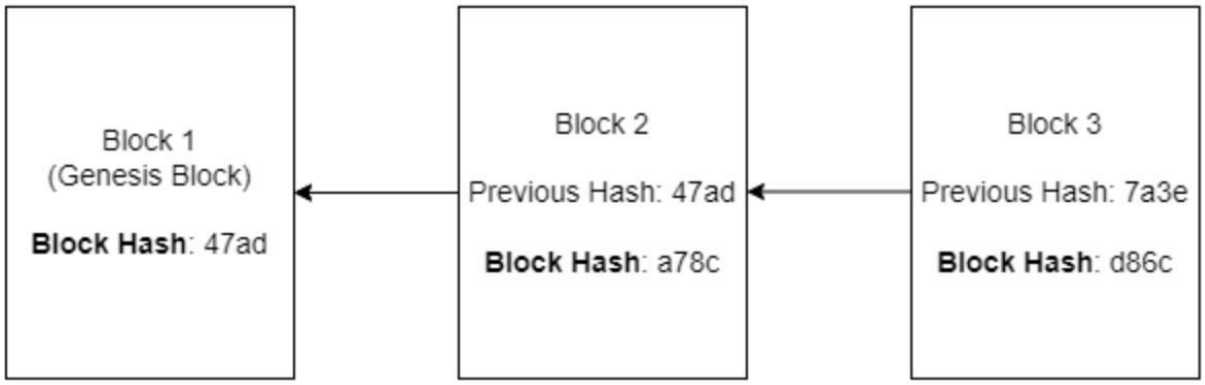

Рисунок 1.3 – Ланцюжок блоків з попереднім хешем [5]

Такий підхід добре підходить для фінансових операцій, де важливо відстежувати рух активів із математичною точністю. Однак UTXO-модель має суттєві обмеження щодо створення складних програмованих сценаріїв, адже вона практично не оперує станом у вигляді структурованих даних. Це ускладнює побудову логічних систем, ігор, токенів з внутрішнім станом, реєстрів та інших динамічних застосунків [3].

З появою Ethereum була запропонована інша парадигма – модель облікових записів (account-based model), у таблиці 1.3 подано порівняння UTXO-моделі та

account-based моделі. У цій системі кожен користувач або смартконтракт має власний стан, що зберігається у глобальному реєстрі. Транзакції змінюють значення полів у відповідних записах, забезпечуючи програмованість і можливість виконувати довільну логіку. Однак така модель значно гірше піддається паралельному виконанню. Якщо два смартконтракти або дві транзакції звертаються до одного й того ж облікового запису, виникає конфлікт стану, і блокчейн змушений виконувати ці операції послідовно. Унаслідок цього вся мережа обмежена глобальною пропускною здатністю, що є критичною проблемою для систем, які прагнуть підтримувати тисячі транзакцій на секунду.

Таблиця 1.3 – Порівняння модель UTXO і account-based [7]

Компонент	Опис
Транзакція вимагає більше місця для зберігання	Транзакція вимагає менше місця для зберігання.
Стан зберігається в транзакціях	Стан зберігається на вузлах
Транзакції обчислювально простіші	Транзакції використовують складні обчислення
Масові транзакції менш ефективні	Масові транзакції ефективніші

У цьому контексті Sui пропонує інноваційну модель представлення стану, що базується на концепції об’єктів. Кожен елемент даних у мережі Sui – це окремий об’єкт із власними полями, власником, типом, життєвим циклом і правилами взаємодії. На відміну від традиційних моделей, стан тут не є «плоским» набором змінних і не прив’язаний до конкретного облікового запису. Об’єкти існують незалежно й можуть бути змінені лише транзакцією, що має до них доступ.

Такий підхід розв’язує ключові проблеми попередніх поколінь блокчейнів. Якщо транзакції працюють із різними об’єктами, їх можна виконувати паралельно, не очікуючи завершення інших операцій. Це дозволяє досягти високої пропускної здатності без складних механізмів шардування або L2 протоколів. Транзакції, що змінюють незалежні об’єкти, не конфліктують між

собою, і валідатори можуть обробляти їх одночасно. Як наслідок, Sui здатний підтримувати потоки операцій у масштабі реальних масових застосунків, таких як ігрові платформи, соціальні мережі, системи електронних документів або децентралізовані фінансові сервіси [6-7].

Об’єктна модель також природно відображає логіку сучасних програмних систем. Наприклад, бюлетень голосування, цифровий сертифікат, NFT-квиток, елемент гри або токен доступу можуть бути представлені у вигляді окремих об’єктів із власною структурою та правилами взаємодії. Мова Move, яка використовується в Sui, жорстко контролює життєвий цикл об’єктів, не дозволяючи створення нелегальних станів чи некоректних операцій. Це підвищує безпеку смартконтрактів і значно зменшує ризики критичних помилок, таких як повторне використання ресурсів, некоректне знищення об’єктів або переповнення стану.

Для розробників такий підхід означає можливість створювати модульні, стійкі до помилок системи, у яких взаємодії між об’єктами визначаються мовою та гарантуються виконанням на рівні протоколу. Move забезпечує формальну перевірку типів і ресурсів, що дозволяє запобігти більшості традиційних вразливостей, характерних для мов загального призначення, як-от Solidity. Це робить екосистему Sui привабливою для створення фінансових інструментів, систем токенізації, логістичних платформ, голосувальних модулів і будь-яких сервісів, де важлива коректність стану [4, 5, 8].

У підсумку модель об’єктів у Sui поєднує високу продуктивність, безпеку та гнучкість. Вона забезпечує органічне масштабування за рахунок паралелізації транзакцій, мінімізує ризик помилок під час програмування та дозволяє створювати складні цифрові системи з чітко визначеною логікою. Архітектурна інновація, закладена у модель стану Sui, стала однією з причин швидкого розвитку екосистеми та зростання її популярності серед розробників Web3-застосунків.

1.3 Архітектурні особливості платформи Sui

Архітектура блокчейн-платформи Sui вирізняється поєднанням кількох технічних рішень, які були спеціально розроблені для усунення недоліків типових структур Web3-систем. На відміну від блокчейнів попередніх поколінь – де обробка транзакцій часто обмежена послідовністю, а доступ до ресурсу блокується глобальним станом – Sui пропонує принципово іншу логіку взаємодії із даними. Її фундаментом є об’єктна модель стану, побудована так, щоб кожна транзакція мала змогу працювати лише з тими елементами, які вона реально змінює, що подано на рисунку 1.4. Такий підхід істотно змінює уявлення про масштабованість і пропускну здатність блокчейна, роблячи Sui одним із найбільш продуктивних рішень на ринку.

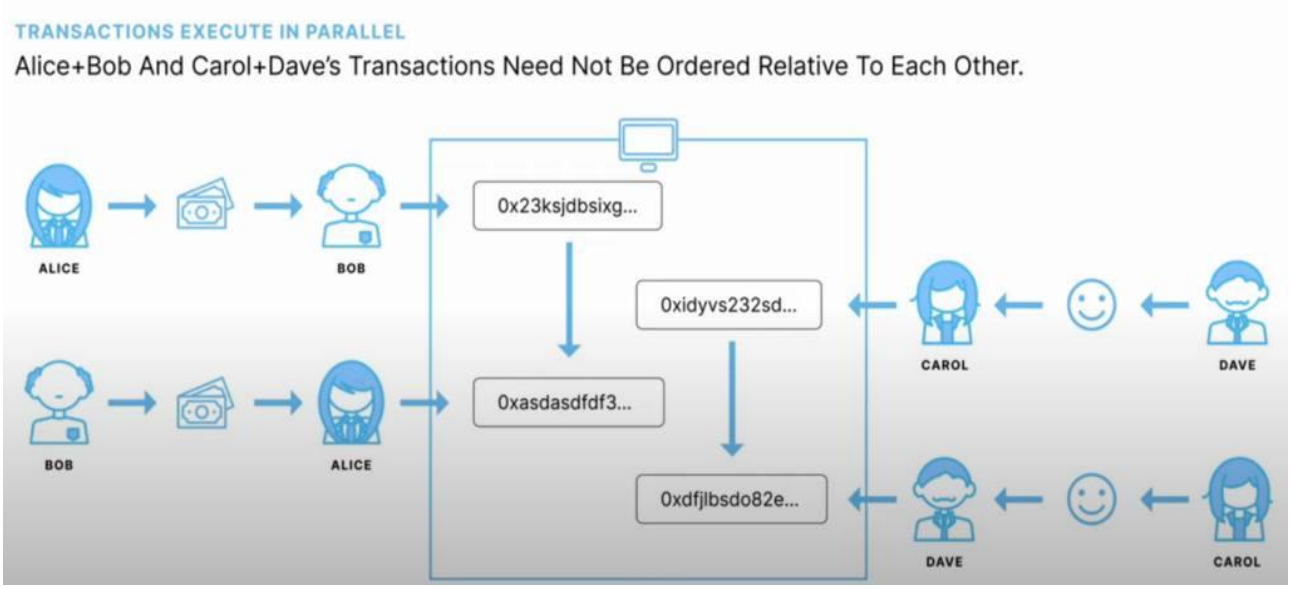

Рисунок 1.4 – Об’єктна модель стану транзакцій [9]

Об’єктна модель дає змогу описувати стан системи не як єдиний глобальний реєстр змінних, а як набір незалежних об’єктів, що існують автономно та взаємодіють між собою за чітко визначеними правилами. Цей підхід суттєво полегшує аналіз транзакції, адже система може точно встановити – які саме об’єкти будуть змінені, а які залишаться недоторканими. Завдяки цьому стає можливим паралельне виконання операцій, що суттєво зменшує

навантаження на вузли й забезпечує швидке підтвердження транзакцій. Особливо важливою є здатність Sui виконувати велику кількість незалежних транзакцій одночасно, що забезпечує практичну масштабованість платформи та дозволяє використовувати її у сценаріях з високою інтенсивністю операцій.

Одним із ключових компонентів архітектури Sui є спеціально розроблена мова Move. Вона створена з орієнтацією на безпечну роботу із цифровими активами – тому кожен ресурс у системі розглядається як такий, що має власника, обмежений життєвий цикл і набір дозволених операцій, еволюція мови Move зображена на рисунку 1.5. Move реалізує концепцію «лінійних типів», яка унеможливлює копіювання або випадкове знищення ресурсу без чіткого дозволу. Це означає, що навіть на рівні компіляції виявляються та блокуються критичні помилки, які у смартконтрактах інших платформ можуть призводити до втрати коштів або порушення логіки системи. Завдяки Move архітектура Sui отримує формально перевірюваний шар безпеки – що надзвичайно важливо для систем, які працюють із цифровими активами, токенами, правами доступу або бюлетенями голосування.

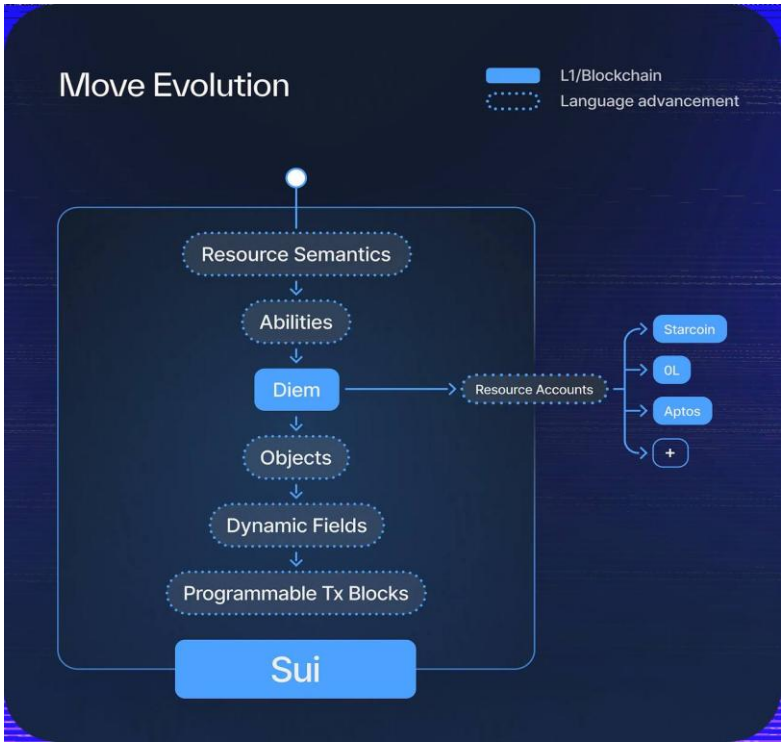

Рисунок 1.5 – Еволюція мови програмування Move [10]

Структура транзакцій у Sui також розроблена таким чином, щоб забезпечити максимальну ефективність виконання. Кожна транзакція супроводжується переліком об’єктів, до яких вона звертається, і це дозволяє мережі одразу визначити – чи є конфлікт між двома операціями. У разі відсутності спільних об’єктів транзакції обробляються паралельно, не створюючи черг і зменшуючи затримки в роботі. Такий підхід суттєво відрізняється від класичних блокчейнів, де глобальний порядок транзакцій вимагає послідовного виконання майже кожної операції. У Sui залежності встановлюються локально, що створює передумови для горизонтального масштабування мережі та забезпечує високу продуктивність навіть за великої кількості користувачів [8, 9].

Важливою особливістю архітектури Sui є те – що вона включає модульну інфраструктуру, яка спрощує розробку та розгортання децентралізованих застосунків. Платформа підтримує RPC-інтерфейси для взаємодії з вузлами, має низку інструментів моніторингу стану мережі, візуалізації транзакцій і аналізу смартконтрактів. Наявність локальних вузлів дозволяє розробникам тестувати свої застосунки без підключення до основної мережі, а блок-експлорери забезпечують прозоре відображення всієї історії транзакцій. Це формує цілісну екосистему, у якій поєднуються мережеві технології, механізми безпеки, інструменти автоматизації та мова програмування Move – що значно підвищує ефективність роботи розробників.

Завдяки своїй архітектурі Sui вже сьогодні розглядається як одна з найбільш перспективних платформ третього покоління. Вона орієнтована на швидку обробку транзакцій, структурну безпеку даних, широкі можливості паралельності та підтримку складних цифрових об’єктів. Такий підхід відкриває шлях до створення високонавантажених реальних систем – від голосувань і електронних реєстрів до ігор, фінансових сервісів, IoT-платформ та інтегрованих корпоративних рішень.

1.4 Криптографічні аспекти та безпека користувача

Безпека блокчейн-систем ґрунтується на застосуванні криптографії з відкритим ключем, яка забезпечує автентифікацію користувача та цілісність транзакцій. Приватний ключ відіграє роль головного ідентифікатора – саме його наявність підтверджує право власності на цифрові активи, об’єкти або смартконтракти. На відміну від традиційних централізованих систем, де ідентифікація здійснюється через логін, пароль або сторонній сервер, у блокчейні користувач самостійно керує своїм криптографічним ключем і не передає його жодній стороні. Це дає високий рівень контролю, але водночас створює можливість незворотної втрати доступу у разі помилки користувача.

Процедура створення ключів зазвичай базується на використанні мнемонічних фраз, які формують детермінований ключовий простір. Такі фрази дозволяють користувачу відновити свою криптографічну пару у будь-який момент і на будь-якому пристрої. Мнемоніка складається з набору слів, які генеруються за певним стандартом і гарантують достатню ентропію для захисту від підбору. У системах Web3 їх роль надзвичайно важлива – саме від правильного збереження мнемонічної фрази залежить безпека цифрових активів. Це робить користувача одночасно і власником, і відповідальним адміністратором свого облікового запису, що є як перевагою, так і потенційним джерелом ризику.

Варто зазначити, що попри складні криптографічні алгоритми, сама криптографія рідко є слабким місцем у блокчейн-системах. Переважна більшість інцидентів пов’язана не з криптографічним зламом, а з логічними помилками, ненадійними реалізаціями смартконтрактів або неправильною поведінкою користувачів. Аналіз найбільших інцидентів Web3 демонструє, що приблизно сімдесят відсотків компрометацій було спричинено помилками в бізнес-логіці контрактів, некоректним керуванням станом або відсутністю захисту від подвійного використання об’єктів та даних. Це свідчить про необхідність впровадження підходів, що мінімізують можливість створення неконсистентних або небезпечних сценаріїв взаємодії всередині системи [10-13].

У цьому контексті Sui пропонує принципово відмінну модель безпеки. Її фундаментом є мова Move, яка створена з урахуванням властивостей ресурсів та їх унікальності. Move не дозволяє копіювати або знищувати ресурс, якщо це прямо не передбачено логікою модуля, що виключає цілий спектр помилок, характерних для смартконтрактів на інших платформах. Кожен об’єкт у Sui має чітко визначений власницький статус, а всі операції з ним підлягають формальній перевірці. Це означає, що система на рівні компіляції блокує операції, які порушують правила володіння або створюють потенційні логічні конфлікти. Подібний підхід істотно знижує ризик появи вразливостей, пов'язаних із некоректною обробкою стану.

Крім того, об’єктна модель Sui дозволяє визначати, які саме об’єкти необхідні для виконання конкретної транзакції. Завдяки цьому транзакція не може зачепити стан, до якого вона не має доступу, що підсилює ізоляцію та робить неможливим неконтрольоване втручання в логіку інших контрактів. Це забезпечує високу прозорість моделі безпеки: розробник чітко знає, які ресурси будуть змінені, а система може гарантувати, що лише транзакції з правильними дозволами будуть виконані. Такий підхід також зменшує кількість потенційних атак, пов'язаних із гонками або паралельними конфліктами, адже транзакції виконуються автономно, якщо їхні об’єкти не перетинаються [14].

У контексті користувацької безпеки платформа Sui значно спрощує роботу з цифровими активами. Оскільки об’єкти мають унікальний життєвий цикл, система може автоматично відстежувати їхній стан і заборонити подвійне використання або несанкціоновану передачу. Це забезпечує додатковий рівень захисту від помилок, які на інших платформах можуть призводити до втрати активів або порушення інваріантів системи. Окрему роль відіграє механізм перевірки транзакцій – кожна транзакція повинна бути підписана приватним ключем, який неможливо відновити зі зворотного боку, а сам підпис перевіряється незалежними валідаторами мережі.

Таким чином, криптографічна безпека платформи Sui спирається на поєднання кількох рівнів захисту – від класичної криптографії з відкритим

ключем до структурної безпеки, яку забезпечує сама архітектура Move та об’єктна модель. Це створює комплексну систему, у якій безпека реалізована не лише на рівні алгоритмів, а й на рівні логіки роботи контрактів, структури стану та поведінки самого блокчейна. Завдяки такому підходу Sui забезпечує значно більшу стійкість до логічних помилок, що робить платформу надійною основою для побудови сучасних децентралізованих сервісів – від систем голосування та електронних реєстрів до цифрових колекцій, токенізованих ресурсів та складних багатокористувацьких застосунків.

1.5 Блокчейн як основа для електронного голосування

Електронне голосування на сьогодні вважається однією з найбільш чутливих та відповідальних сфер цифрової трансформації, оскільки поєднує високі вимоги до безпеки даних, зручності користування та довіри суспільства до результатів процесу. Одночасно система має гарантувати анонімність виборця, прозорість механізму підрахунку голосів, незмінність даних і захищеність від стороннього втручання. Традиційні централізовані голосувальні платформи не здатні забезпечити баланс між цими вимогами, адже централізований сервер завжди виступає потенційною точкою атаки або маніпуляції результатами.

Саме тому блокчейн розглядається як технологічна основа для побудови електронних виборчих систем нового покоління, оскільки його фундаментальні властивості – децентралізація, криптографічна цілісність, відсутність можливості редагувати дані постфактум та публічний аудит – відповідають вимогам безпечного голосування [15-19].

Однією з ключових переваг блокчейна у цьому контексті є те, що кожен виборчий бюлетень може бути представлений як криптографічно захищений запис, який неможливо змінити після подання. Такий запис матиме часову мітку, цифровий підпис і підтвердження мережі, що унеможливлює підробку або дублювання голосів. При цьому анонімність користувача забезпечується тим, що

ідентифікація та виборчий процес розділяються, а публічний реєстр не містить персональних даних виборців [17-20].

У традиційних блокчейнах (таких як Ethereum) голосування реалізовується через смартконтракти, однак подібні системи часто стикаються з проблемою масштабованості та високої вартості обробки транзакцій. Для реального електронного голосування, де одночасно можуть голосувати сотні тисяч користувачів, ці обмеження стають критичними.

У цьому контексті особливе місце займає Sui – блокчейн третього покоління, який реалізує об'єктно-орієнтовану модель стану та можливість паралельної обробки транзакцій. Завдяки цій моделі кожен голос може бути представлений як окремий об'єкт, що має власну структуру, життєвий цикл та правила модифікації. Об'єкт голосу може містити інформацію в зашифрованому вигляді та мати статус «одноразового використання», що унеможливлює повторне голосування конкретним виборцем [21, 22, 23]. Це значно підвищує стійкість системи до маніпуляцій, порівняно з моделлю, де всі голоси записуються в один глобальний реєстр.

Архітектура Sui дозволяє будувати голосувальні системи, де підрахунок результатів виконується детерміновано та прозоро у смартконтракті, а інтерференція між транзакціями мінімізується завдяки тому, що операції з різними голосами не блокують одна одну. Це особливо важливо для масштабних виборів, де система має обробляти тисячі голосів за секунду без підвищення вартості або затримок.

Окрему увагу заслуговують криптографічні методи, що застосовуються у системах електронного голосування. Зокрема, активно використовуються технології сліпого підпису, які дозволяють виборцю отримати підтвердження права голосу без розкриття його вибору. Іншим важливим напрямом є гомоморфне шифрування – метод, що дозволяє виконувати підрахунок голосів без їх розшифровки, забезпечуючи повну анонімність виборця та прозорість результатів. Поєднання цих підходів із моделлю об'єктів Sui створює можливість

для розробки виборчих систем, де всі дані зашифровані, а перевірка їх коректності є публічною [23-27].

Таким чином, блокчейн, і зокрема платформа Sui, формує технологічну основу для високозахищених, масштабованих і прозорих систем електронного голосування. Поєднання децентралізації, об’єктної структури даних та криптографічного захисту дозволяє створювати механізми, у яких виборці довіряють як власному вибору, так і результатам підрахунку, а втручання у виборчий процес стає технічно неможливим.

РОЗДІЛ 2
ПРОГРАМНА РЕАЛІЗАЦІЯ SUI-ГАМАНЦЯ

2.1 Налаштування робочого середовища для розробки на Sui та Move

Розробка інтелектуальної системи голосування та мобільного Sui-гаманця потребує попереднього налаштування комплексного середовища, яке включає інструменти командного рядка, компілятор Move, інфраструктуру локальної мережі та програмні залежності. Оскільки блокчейн Sui базується на новій об’єктній моделі та власній мові програмування Move, налаштування середовища відіграє критично важливу роль для подальшої розробки, тестування та виконання транзакцій.

Першим кроком стало встановлення Sui Command Line Interface (Sui-CLI), який є основним інструментом взаємодії з мережею Sui на рівні розробника. Він дозволяє створювати локальні гаманці, надсилати транзакції, будувати та публікувати Move-модулі, а також запускати локальну тестову мережу. Встановлення Sui-CLI здійснюється через офіційний інсталятор, що базується на перетворенні системи до актуальної версії Rust-компілятора (лістинг 2.1).

Лістинг 2.1 – Команди встановлення «Sui-CLI»

```
1. curl -fsSL https://install.sui.io | sh
2. sui –version
3. sui genesis
4. sui client active-address
5. sui client addresses
6. sui move new LNTUProject
```

кінець лістингу 2.1

Ця команда завантажує останню стабільну збірку Sui та встановлює її у системі. Після завершення інсталяції необхідно перезапустити термінал або

вручну оновити PATH, щоб shell міг коректно знаходити нові виконувані файли. Під час налаштування застосовувалась команда 2 для перевірки версії (лістинг 2.1) та рисунок 2.1.

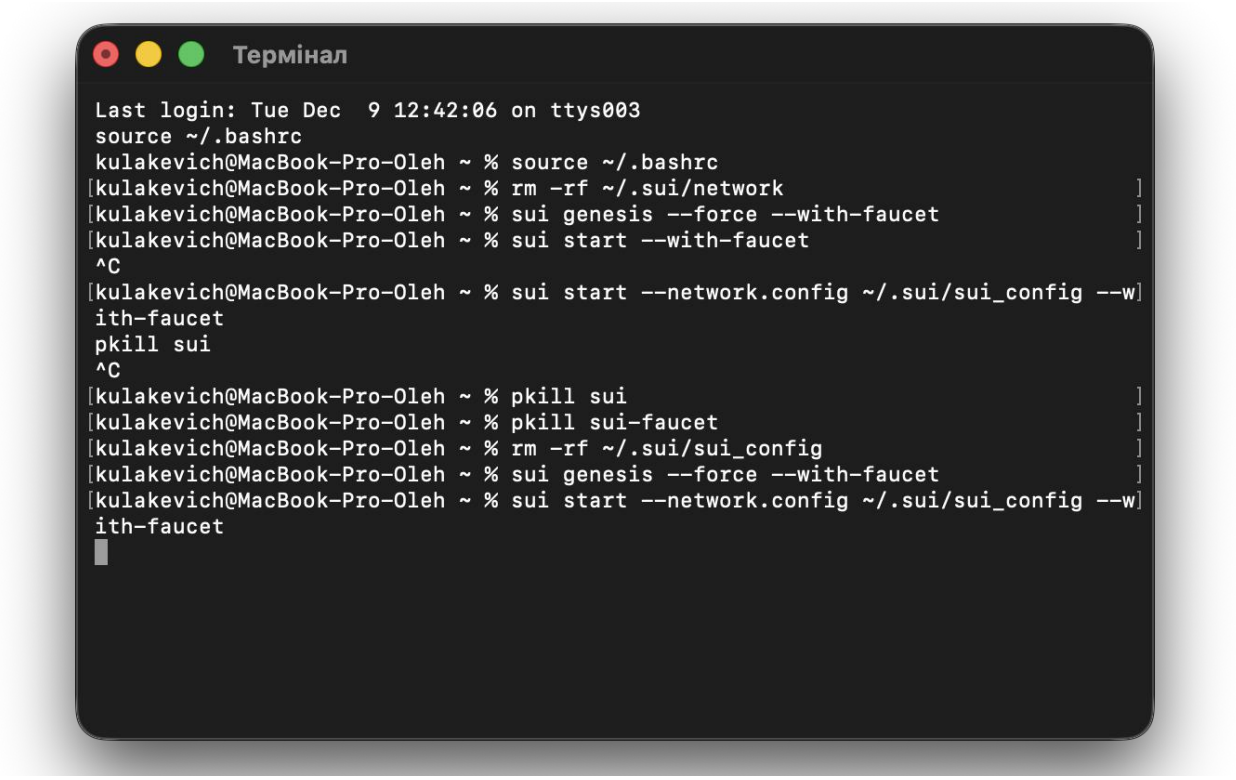

Рисунок 2.1 – Встановлення SUI CLI

Наступним кроком стало створення локального блокчейн-середовища – так званої мережі Localnet. Вона призначена для тестування смартконтрактів і транзакцій без витрат реального газу, у повністю ізольованому середовищі, яке відтворює логіку реальної мережі. Ініціалізація локальної мережі виконується командою 3 (лістинг 2.1)

Команда створює локальний генезис-файл, який визначає початковий стан мережі: список акаунтів, кількість монет, параметри протоколу та конфігурацію вузлів. Це дозволяє розробнику запускати транзакції, розгортати Move-модулі та аналізувати їхню поведінку без ризику вплинути на реальний блокчейн.

Далі виконується перевірка активної адреси гаманця, автоматично згенерованого під час початкової конфігурації за допомогою команди 4 (лістинг 2.1). Ця команда відображає адресу, яка буде використовуватися за

замовчуванням під час виконання транзакцій і публікації модулів. У разі необхідності можна переглянути всі доступні адреси, а також створити нову адресу або імпортувати seed-фразу за допомогою команди 5 (лістинг 2.1).

Щоб забезпечити повноцінний цикл розробки Move-пакетів, необхідно створити нову структуру проєкту. Для цього використовується інструмент командного рядка CLI команда 6 (лістинг 2.1):

Дана команда створює типову структуру каталогу, що включає директорії sources (для модулів Move) та tests (для інтеграційних тестів). Усередині проєкту лежить файл Move.toml, який визначає залежності та конфігурацію пакета. Робота з Move у значній мірі нагадує побудову модулів у Rust, але застосовує власні правила ресурсної безпеки, які забезпечують унеможливлення дублювання або втрати об’єктів.

Для подальшої роботи було необхідно налаштувати RPC-з’єднання з обраною мережею. Під час дослідження застосовувалися як локальна мережа Localnet, так і загальнодоступні тестові мережі Devnet і Testnet. Для перемикання між середовищами використовується команда з лістингу 2.2:

Лістинг 2.2 – Команди взаємодії з «Sui-CLI»

```
sui client switch --env testnet
// для локальної мережі:
sui client switch --env localnet

// перевірка доступності RPC
curl http://127.0.0.1:9000
// перевірка faucet-служби локальної мережі
curl http://127.0.0.1:9123
```

кінець лістингу 2.2

Це дозволяє виконувати транзакції в різних середовищах, оцінювати рекцію RPC-інфраструктури та відстежувати поведінку Move-контрактів в

умовах, максимально наближених до продакшн-середовища. Важливо, що під час розробки мобільного застосунку використовувався саме RPC-доступ, оскільки iOS-додаток взаємодіє з мережею через HTTP-виклики та JSON-RPC протокол.

Після цього локальна мережа запускається командою: sui start (рис. 2.2). Вона ініціює вузол, запускає процес валідатора та відкриває локальний RPC-ендпоінт, доступний зазвичай за шляхом http://127.0.0.1:9000. Під час тестування також використовувались допоміжні команди з лістингу 2.2.

Це дозволяє автоматично отримувати тестові монети SUI на адресу гаманця для подальших експериментів із транзакціями.

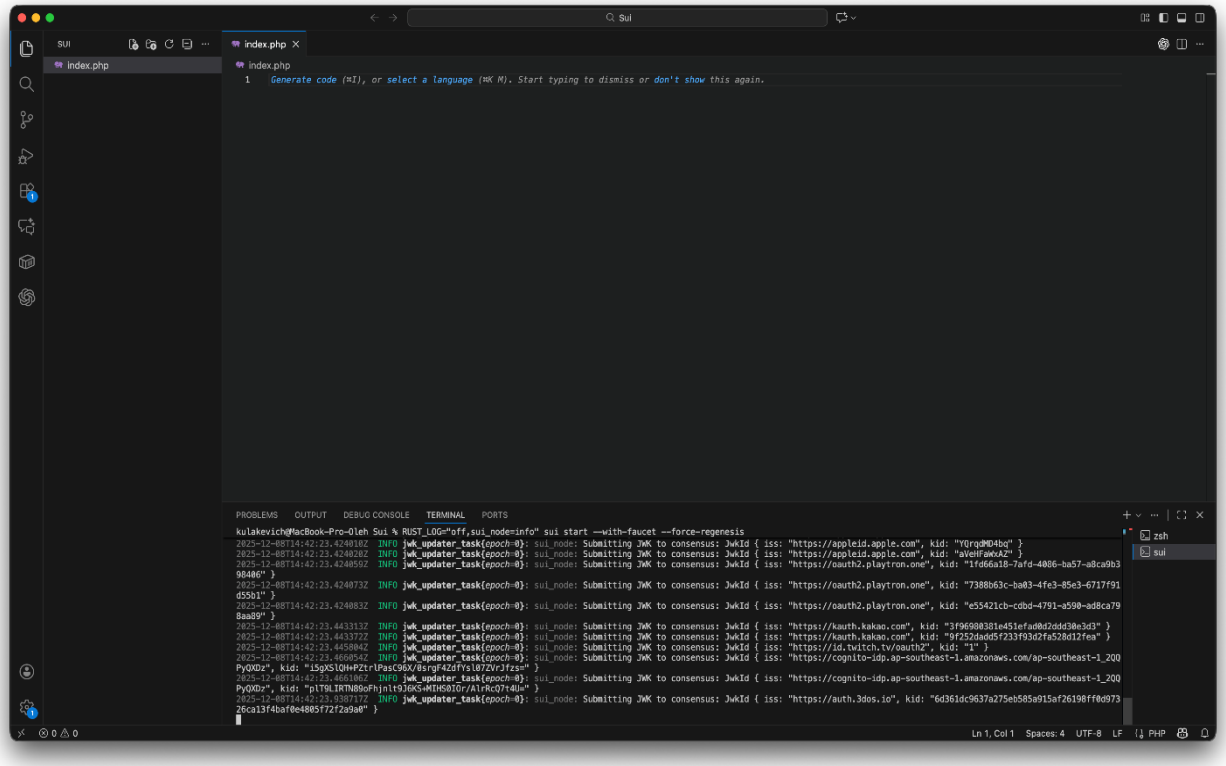

Рисунок 2.2 – Запуск блокчейну SUI

Таким чином, налаштування робочого середовища включало встановлення Sui CLI, запуск локальної мережі, створення Move-проєкту, налаштування RPC-з’єднання та отримання тестових монет. Усі ці кроки є необхідними для подальшої розробки як складної бізнес-логіки на стороні Move, так і клієнтських

застосунків, що взаємодіють з блокчейном через програмні інтерфейси. Завдяки локальній інфраструктурі розробник отримує змогу швидко тестувати функції, відлагоджувати транзакції та симулювати роботу смартконтрактів у контрольованому середовищі, що забезпечує надійність та передбачуваність кінцевої системи.

2.2 Створення першого Move-модуля

Для демонстрації роботи з об’єктами було створено базовий модуль, що описує структуру цифрового ресурсу – наприклад, голос, токен або інший елемент стану, що має власника та унікальний ідентифікатор.

Структура типового модуля має на ступний вигляд відображений у лістингу 2.3, а також додатках Б, В.

Лістинг 2.3 – Структура типового модуля SUI

```
module voting_system::ballot {
      struct Ballot has key, store {
           id: UID, choice: u8, owner: address
       }
      public entry fun create(choice: u8, ctx: &mut TxContext): Ballot {
           let ballot = Ballot {id: object::new(ctx),choice, owner:
      tx_context::sender(ctx)};
            ballot
       }
      public fun read(ballot: &Ballot): u8 {
          ballot.choice
      }
}
```

кінець лістингу 2.3

У цьому прикладі застосовано ключове поняття Move – ресурс, який не можна довільно копіювати або знищувати. Поле id: UID використовується для того, щоб об’єкт міг існувати в глобальному стані Sui, а ключове слово key визначає можливість його зберігання у сховищі блокчейна. Після написання модуля необхідно виконати компіляцію sui move build, ця команда перевіряє синтаксис, типи, коректність роботи з ресурсами та відповідність модулів правилам Move-безпечності. У разі виникнення помилок компілятор вказує на конкретні порушення, наприклад дублювання ресурсів, недозволене копіювання або неправильну передачу посилань.

Після успішної компіляції модуль може бути розгорнутий у локальній мережі (рис. 2.3) через команду sui client publish --gas-budget 50000000

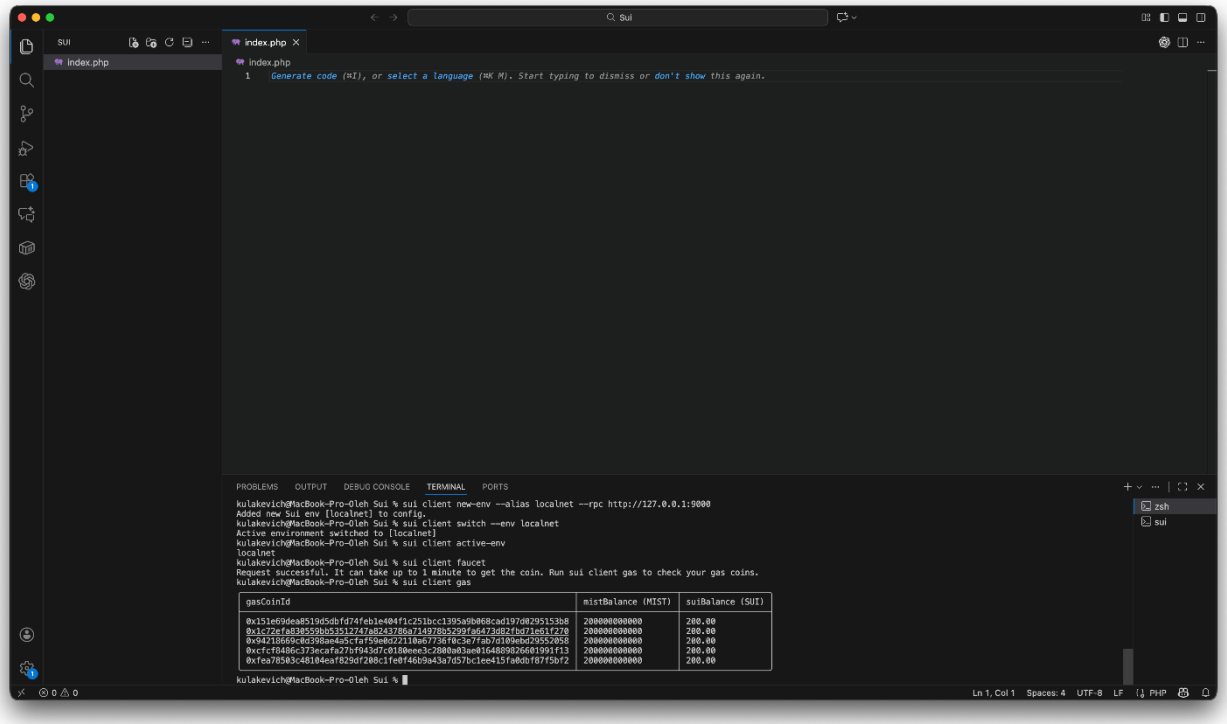

Рисунок 2.3 – Присвоєння gas токенів у мережі

Ця команда створює транзакцію, яка додає модуль в глобальний стан локального блокчейна. Під час публікації система генерує унікальний ID пакета, який є обов’язковим для виклику функцій модуля у подальших транзакціях.

На цьому етапі можна виконати тестові транзакції, наприклад створення об’єкта голосу (лістинг 2.4).

Лістинг 2.4 – Виконання тестової транзакції SUI

```
sui client call \
--package <PACKAGE_ID> \
--module ballot \
--function create \
--args 1 \
--gas-budget 50000000
```

кінець лістингу 2.4

Виконання цієї команди повертає ID створеного об’єкта, який можна переглянути або використати для інших операцій.

Move має власну систему юніт-тестування, яка дозволяє розробнику перевіряти логічну коректність модуля без взаємодії з блокчейном. Тести розміщуються в каталозі tests/ і описуються Move-сценаріями приклад тесту подано у лістингу 2.5.

Лістинг 2.5 – Юніт тестування SUI

```
fun test_create_ballot() {
     let ctx = &mut TxContext::empty();
      let ballot = ballot::create(1, ctx); assert!(ballot.choice == 1,
0);}
// Запуск тестів здійснюється командою
sui move test
```

кінець лістингу 2.5

Під час виконання Move-тестів компілятор створює ізольоване середовище, в якому модуль виконується без взаємодії з мережею. Це дозволяє виявляти логічні помилки та перевіряти коректність процедур ще до публікації пакета в блокчейні.

Для системи голосування було розширено модуль додатковими функціями – наприклад, перевіркою можливості голосувати лише один раз, обмеженням доступу до зміни параметрів об’єкта, а також функцією підрахунку голосів.

Такі функції дозволяють моделювати різні сценарії голосування: редаговані голоси, остаточні голоси або анонімні голоси з подальшим розшифруванням.

Усі зміни щоразу проходять через етапи:

– локальний запуск тестів;

– компіляцію;

– повторну публікацію модуля;

– тестування через CLI або мобільний додаток.

Цей цикл формує повноцінний pipeline розробника для Move-модулів.

2.3 Реалізація Sui-гаманця у середовищі iOS (Swift)

Розробка мобільного застосунку, який виконує функції Sui-гаманця, вимагає комплексного підходу до архітектури, безпеки та роботи з мережею. На відміну від традиційних застосунків, блокчейн-клієнт має взаємодіяти з децентралізованим середовищем, забезпечувати локальне зберігання приватних ключів, формувати криптографічні підписи та відправляти транзакції до RPC-сервера. У цьому підрозділі описано процес створення iOS-застосунку на мові Swift, що реалізує повний цикл роботи Sui-гаманця: генерацію та імпорт ключів, перегляд балансу, взаємодію з Move-модулями та виконання транзакцій.

Архітектура була побудована за принципами MVVM, що дозволило розділити логіку даних, бізнес-процеси та UI-компоненти. У структурі застосунку виділено кілька основних шарів:

– файл WalletManager відповідає за генерацію, імпорт та зберігання мнемонічних фраз, створення приватного/публічного ключів і формування транзакцій;

– файл SuiClient здійснює комунікацію з блокчейном через RPC, формує та відправляє запити, декодує відповіді;

– структура Models структура для опису токенів, транзакцій, об’єктів, що повертає блокчейн;

– класи Views & ViewModels інтерфейс гаманця, екрани відображення балансу, історії операцій, форми переказу.

Особливу увагу було приділено безпеці – приватні ключі не передаються у мережу та не зберігаються у відкритому вигляді (див. додаток Г). Для зберігання мнемонічної фрази застосовано Keychain, що відповідає стандартам безпеки Apple, структуру проекту можна переглянути на рисунку 2.4.

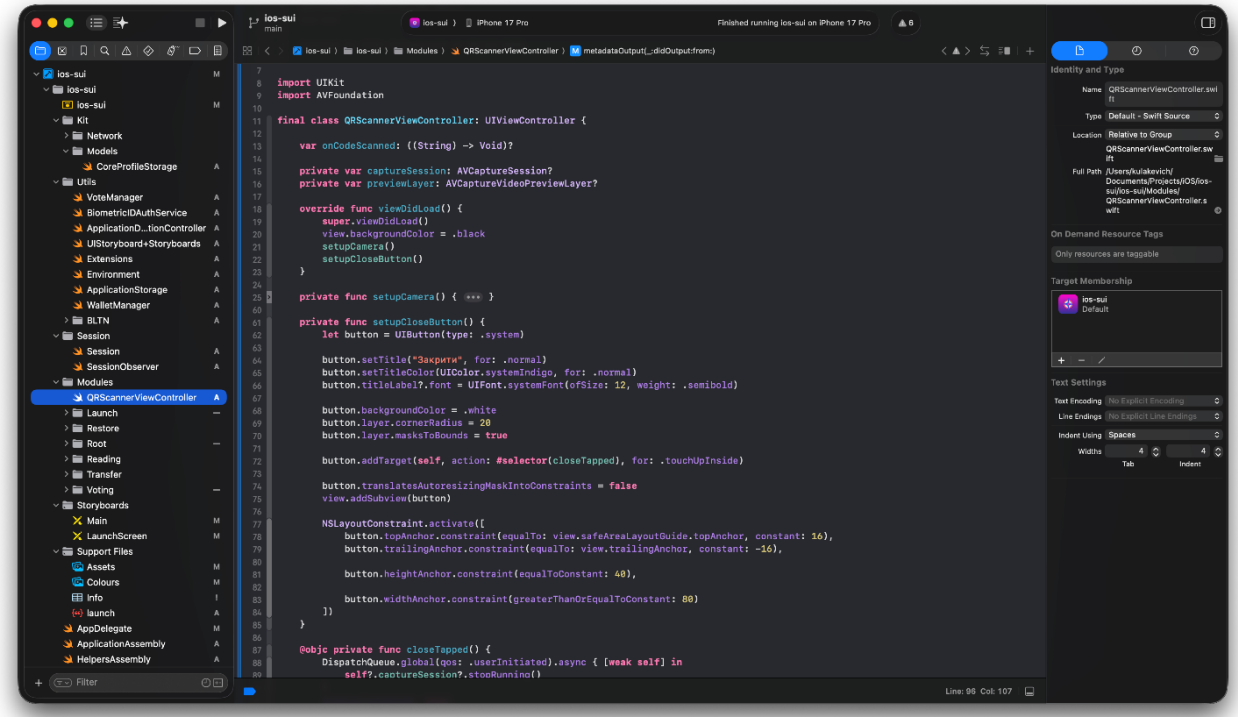

Рисунок 2.4 – Структура проекту та головної директорії Utils

Першим етапом створення гаманця стало впровадження механізму генерації мнемонічної фрази (seed phrase), яка є основою для приватного ключа Sui. Для цього використано сторонню бібліотеку, адаптовану для Swift, що підтримує алгоритм BIP-39. Приклад коду створення фрази подано у лістингу 2.6.

Лістинг 2.6 – Створення гаманця Wallet

```
let mnemonic = Mnemonic.generate(strength: .default)
let seed = Mnemonic.seed(mnemonic: mnemonic)
let privateKey = SuiPrivateKey(seed: seed)
KeychainService.save("wallet.mnemonic", mnemonic)
```

кінець лістингу 2.6

Після створення seed фраза одразу зберігається у Keychain, це гарантує, що дані не потрапляють у незахищену пам’ять та не можуть бути прочитані сторонніми процесами (див. рис. 2.5).

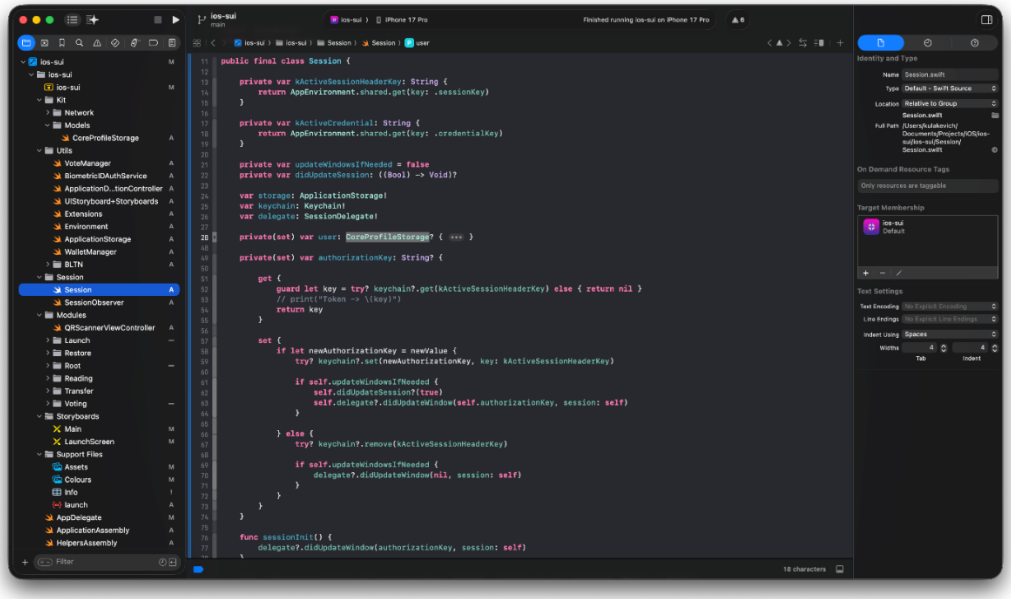

Рисунок 2.5 – Файл Session і збереження сид-фрази у Keychain

Для комунікації з мережею застосунок використовує JSON-RPC API. Був реалізований універсальний клієнт, який відправляє запити та обробляє відповіді (лістинг 2.7)

Лістинг 2.7 – Відправка запиту до RPC

```
func call(method: String, params: [Encodable]) async throws -> JSON {
      var request = URLRequest(url: rpcURL); request.httpMethod = "POST"
      let payload = RPCRequest(method: method, params: params)
      request.httpBody = try JSONEncoder().encode(payload)
      let (data, _) = try await URLSession.shared.data(for: request)
      return try JSONDecoder().decode(JSON.self, from: data)
}
```

кінець лістингу 2.7

Цей механізм дозволив реалізувати низку функцій:

– отримання балансу;

– перегляд списку об’єктів (coins);

– відправлення транзакцій;

– виклик функцій Move-модулів.

Для балансу використано RPC-метод: sui_getBalance, а для отримання списку активів sui_getCoins. Усі відповіді перетворюються на Swift-структури для зручності використання у ViewModel. Однією з ключових задач було створення переказу SUI з використанням приватного ключа користувача. Транзакція формується локально, підписується приватним ключем, і лише після цього передається через RPC (лістинг 2.8).

Особлива увага була приділена підтримці різних типів помилок (рис. 2.6):

– noCoinsFound – коли користувач не має об’єкта SUI достатнього розміру;

– invalid signature – некоректний підпис через помилки в seed;

– invalid params – неправильний формат параметрів, очікування строкових значень.

Лістинг 2.8 – Формування та підпис транзакції RPC

```
let txData = try SuiTransactionBuilder.paySui(
    to: recipient,
    amount: amount,
    gasBudget: gas
)
let signature = try privateKey.sign(txData)
Відправлення:
let result = try await client.executeTransaction(
    txBytes: txData.base64EncodedString(),
    signature: signature.base64
)
```

кінець лістингу 2.8

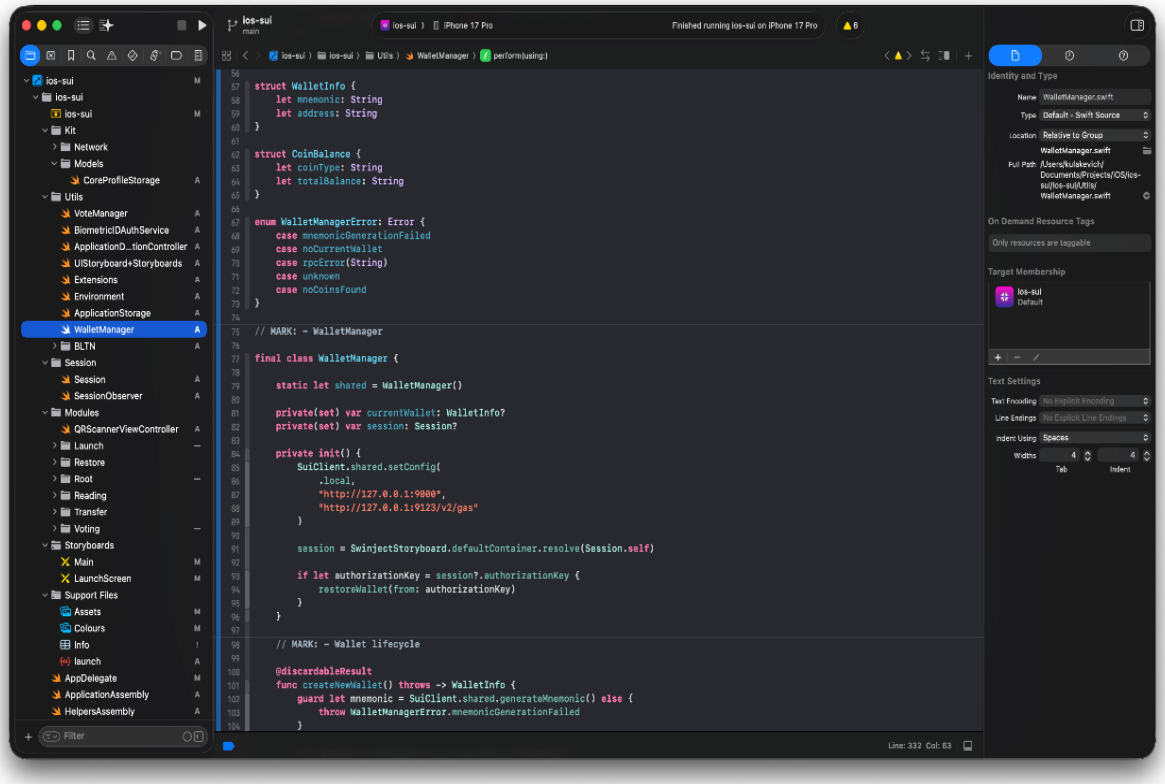

Рисунок 2.6 – Файл WalletManager та структура обробки помилок

Систематичне тестування дозволило адаптувати гаманець для стабільної роботи у локальній мережі та тестнеті. Для інтеграції системи голосування, реалізованої у Move, було створено функцію виклику entry-функцій модулів. Це дає змогу працювати з блокчейном на рівні користувача:

– створювати об’єкт голосу;

– оновлювати його;

– перевіряти стан;

– проводити транзакції з іншими об’єктами.

Застосунок виступає повноцінним клієнтом Move-контрактів.

Графічний інтерфейс створено на SwiftUI та UIKit (для окремих екранів), що забезпечує швидку побудову UI та адаптивність. Під час виконання транзакцій баланс оновлюється автоматично через повторний RPC-запит (лістинг 2.9)

Лістинг 2.9 – Оновлення балансу через запит до RPC

```
let result = try await client.callFunction(
    packageId: package, module: "ballot", function: "create", arguments:
[choice], gasBudget: 10000000)
Task {
    self.balance = try await client.getBalance(address)
}
```

кінець лістингу 2.9

Таким чином, користувач одразу бачить зміни після відправлення SUI чи взаємодії з Move-модулями.

2.4 Інтеграція Move-модулів та комплексна програмна реалізація Sui-гаманця

Розробка повноцінного мобільного Sui-гаманця потребує комплексного підходу, який охоплює не лише роботу з ключами та базовими транзакціями, а й можливість взаємодії з користувацькими Move-модулями, що реалізують прикладну логіку блокчейн-системи. У межах цього проєкту було розроблено та інтегровано модулі голосування, створено механізми для виклику їхніх функцій зі Swift-коду, а також впроваджено інструменти для підрахунку голосів, перевірки стану об’єктів і забезпечення коректної взаємодії з об’єктною моделлю Sui.

У процесі інтеграції мобільний застосунок із середовищем Move розглядався як клієнт, який працює виключно з об’єктами, що існують у блокчейні. На відміну від платформ, де більшість логіки побудовано навколо глобального стану, Sui вимагає визначення конкретних об’єктів і їхніх власників. Саме тому всі функції, що реалізують голосування, створення бюлетенів і перевірку їхнього стану, виконуються над окремими об’єктами, які повертає Move-можуль. У Swift-проєкті була створена система серіалізації та десеріалізації об’єктів, що дозволяє отримувати їх у структурованому вигляді, а також здійснювати подальшу взаємодію з ними через RPC-запити.

Проєктна логіка вимагала розробки двох основних компонентів: механізму виклику entry-функцій модулів та механізму читання стану об’єктів. Взаємодія із Move-модулем здійснювалася через функцію callFunction, що формує пакет параметрів, базованих на формальних типах, визначених у Move. Для забезпечення стабільності застосунку було впроваджено моделі, які автоматично перетворюють дані з JSON у Swift-структури. Таким чином, застосунок отримує доступ до інформації про створені об’єкти голосування, їхній статус та можливі зміни, визначені логікою контракту. Завдяки цьому користувач може перевірити свій голос, переглянути кількість поданих голосів або ініціювати створення нового об’єкта голосування.

Створення транзакцій є ключовим етапом у взаємодії застосунку з блокчейном. У процесі інтеграції було реалізовано систему формування транзакцій відповідно до вимог Sui, яка включає підпис приватним ключем, визначення об’єктів, що змінюються, та призначення бюджету газу. Формування транзакції для голосування відбувається у кілька кроків. Спершу застосунок отримує актуальний список об’єктів, визначає доступний gas coin, формує байтове представлення транзакції, а потім підписує його приватним ключем. Підписана транзакція надсилається до RPC-сервера. У разі успішного виконання мережа повертає digest, який зберігається локально і може використовуватися для перевірки статусу.

Під час інтеграції модулів виникла низка труднощів, пов’язаних із тим, що Sui вимагає точного визначення типів об’єктів, їхніх ID та ролей. Будь-яка невідповідність у структурі аргументів або у визначенні owner призводила до помилок типу “Invalid Params” або “Object does not exist”. Крім того, під час тестування виявлено, що у випадку, коли транзакція впливає на об’єкт, який не належить поточному користувачу, консенсус відхиляє операцію. Це потребувало створення додаткової внутрішньої перевірки — застосунок визначає, якими саме об’єктами володіє користувач, та відображає тільки ті, з якими він може взаємодіяти. Така логіка дозволила мінімізувати помилки та забезпечити коректну взаємодію застосунку з контрактами.

Суттєву частину роботи було присвячено тестуванню модулів голосування в локальному середовищі. Тестування включало перевірку функцій create_vote, submit_vote та get_vote_result, що виконуються у різних сценаріях. Було змодельовано кілька типових ситуацій: створення нового голосування, подання голосу користувачем, спроба повторного голосування, що заборонена логікою контракту, перевірка результатів після кількох транзакцій, а також моделювання помилкових ситуацій. Для кожного тесту застосунок надсилав транзакцію, отримував digest та перевіряв статус виконання через RPC-функцію sui_getTransactionBlock.

Процес налагодження включав детальне відстеження RPC-відповідей. Було розроблено спеціальний debug-режим, який у вигляді форматованого журналу виводить усі відправлені параметри, транзакційні пакети, RPC-методи та відповіді. Це дозволило швидко виявляти помилки у формуванні транзакцій, неправильні типи аргументів чи невідповідність ID об’єктів. Такий підхід забезпечив високу точність взаємодії застосунку з блокчейном та дав змогу проводити глибоке тестування без необхідності створення складного бекенд-середовища.

Важливою частиною цього розділу стала робота над підвищенням стійкості застосунку до мережевих проблем. У мобільних умовах RPC-запити можуть виконуватися з великими затримками або перериватися. Для цього було створено систему повторних запитів з експоненційною затримкою, обробку помилок рівня 5xx та fallback-механізм із використанням резервного RPC-сервера. Окрім цього, рішення передбачає локальне кешування результатів запитів, таких як баланс, список об’єктів, історія транзакцій, що прискорює роботу застосунку.

Окремий блок дослідження стосувався безпеки гаманця. Було впроваджено кілька механізмів: шифроване зберігання мнемонічної фрази у Keychain, недоступність приватного ключа за межами WalletManager, автоматичне стирання ключових даних під час виходу з облікового запису, а також захист критичних функцій FaceID/TouchID. Таке рішення гарантує, що навіть у разі фізичного доступу до пристрою стороння особа не отримає доступ до ключів. Було також протестовано сценарії розшифрування seed phrase та відновлення гаманця після видалення застосунку, що підтвердило коректність роботи з Keychain.

Взаємодія з Move-модулями у контексті голосування стала ключовим компонентом практичної частини проєкту. Було реалізовано логіку, за якої кожен користувач створює окремий голос-об’єкт, підписує його транзакцією, а смартконтракт контролює правила взаємодії та підрахунку голосів. Такий підхід демонструє ефективність об’єктної моделі Sui у ситуаціях, де від кожного

учасника очікується автономна дія, що не потребує змін глобального стану. Мобільний застосунок виступає інструментом для створення голосів, перевірки статусів та відображення результатів.

Завдяки реалізованій архітектурі застосунок працює стабільно, підтримує повний цикл взаємодії з Sui та демонструє можливості побудови блокчейн-рішень на базі мобільних технологій. Інтеграція Move-модулів у мобільний екосистему відкриває широкі перспективи для створення децентралізованих додатків – від голосувань і токенізованих систем до складних облікових моделей, де цифрові об’єкти виступають повноцінними елементами логіки.

2.5 Тестування програмної реалізації SUI гаманця

Інтерфейс мобільного застосунку є одним із ключових компонентів системи, оскільки саме він забезпечує доступ користувача до функціональності Sui-блокчейна. У процесі розробки було створено інтуїтивний, мінімалістичний і водночас технологічно насичений інтерфейс, який дає змогу виконувати різні операції – від перегляду балансу до створення транзакцій чи участі в голосуванні. Структура застосунку побудована таким чином, щоб користувач міг у кілька натискань отримати доступ до всіх основних можливостей без необхідності поглибленого вивчення блокчейн-технологій.

Початковий екран застосунку відображає основну інформацію профілю користувача: адресу гаманця, доступний баланс та блок інтерактивних кнопок, які відкривають відповідні модулі роботи з мережею Sui. Адреса виводиться у скороченому форматі для зручності сприйняття, а баланс оновлюється автоматично після кожної транзакції або за запитом користувача. Цей екран також містить доступ до меню перегляду seed-фрази, що дає можливість безпечно зберегти або відновити гаманець у разі потреби (рис. 2.7).

Важливим елементом інтерфейсу є модуль транзакцій, який реалізує перегляд усіх операцій користувача – надсилання токенів, отримання коштів або взаємодія зі смартконтрактами. Цей розділ представлений у вигляді таблиці, де

кожен запис містить час виконання, унікальний ідентифікатор транзакції, її статус та тип операції. Такий підхід дозволяє користувачу прозоро відстежувати всі дії, виконані його гаманцем, а також підтверджувати їх у блок-експлорері (рис. 2.8).

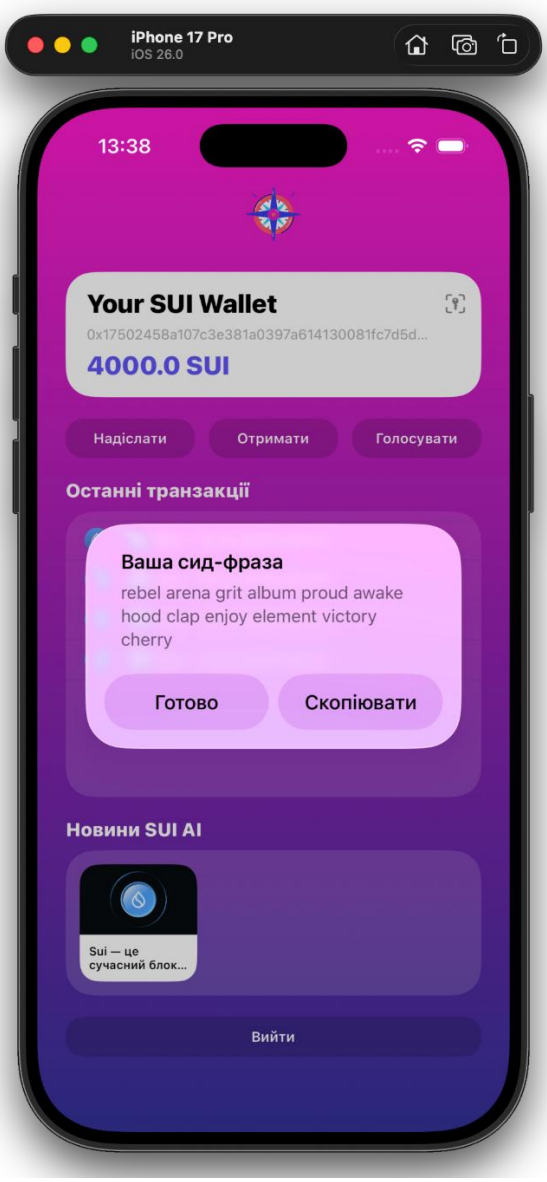

Рисунок 2.7 – Головний екран застосунку

Суттєву увагу приділено взаємодії зі смартконтрактами голосування. Інтерфейс реалізовано у вигляді окремого екрана, де користувач може обирати варіант голосу натисканням на відповідну кнопку. Після вибору застосунок формує транзакцію, підписує її приватним ключем, локально збереженим у

системі Keychain, і надсилає в мережу. Користувач отримує індикатор статусу виконання транзакції та повідомлення про успішне завершення або відхилення операції. Простота процедури вибору робить систему зручною для впровадження у мобільні голосувальні середовища, де швидкість взаємодії є важливою (рис. 2.9).

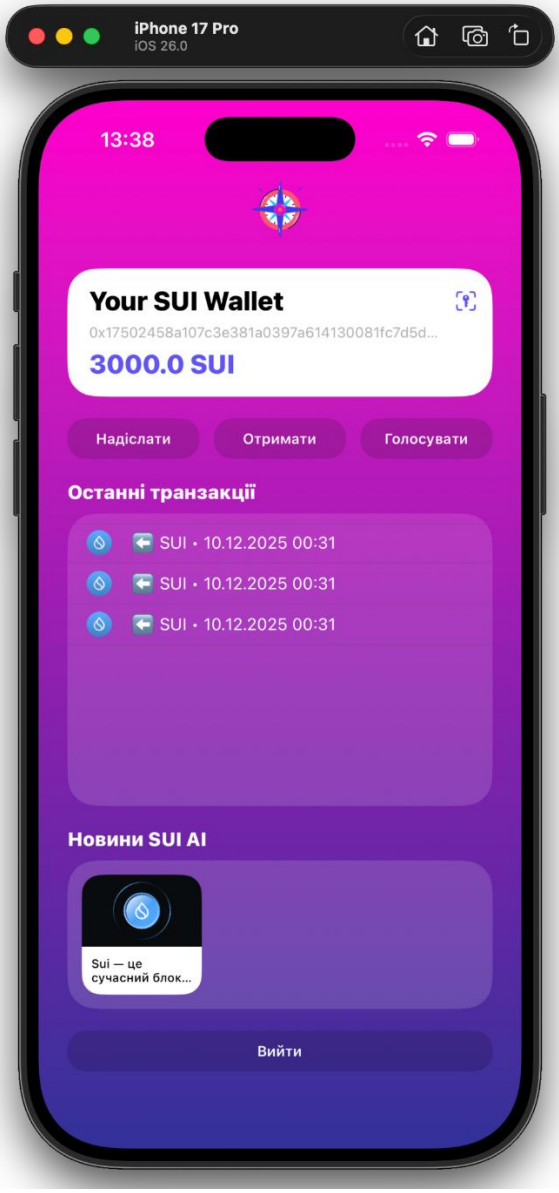

Рисунок 2.8 – Таблиця транзакцій

Особливістю застосунку є інтеграція модуля генерації статей та підказок зі штучним інтелектом. У межах роботи було реалізовано можливість формування контенту через API-взаємодію, а отримані результати відображаються у вигляді

окремих сторінок, які користувач може перегортати в каруселі. Це додає застосунку функціональності поза межами звичайного гаманця та робить його інструментом для дослідження та навчання.

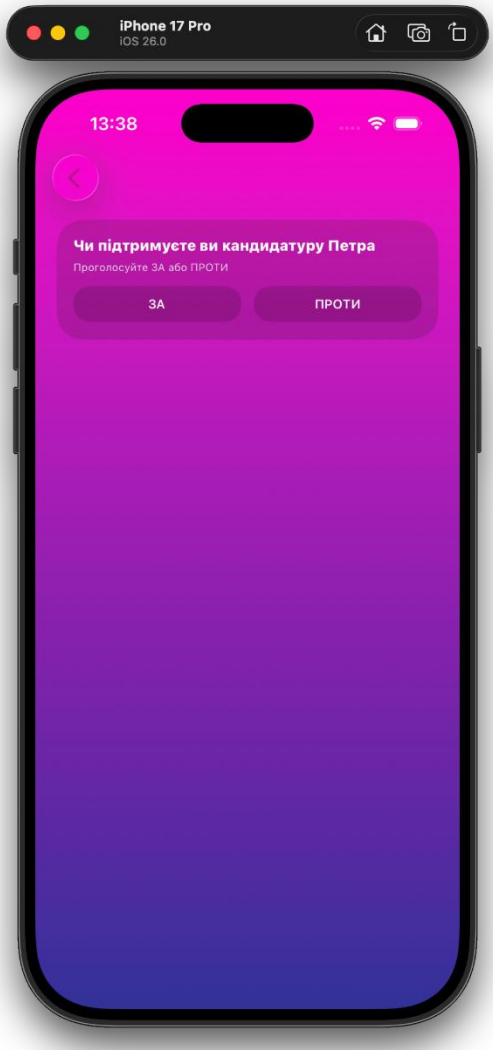

Рисунок 2.9 – Екран голосування

Інтерфейс надсилання та отримання токенів реалізований з особливою увагою до юзабіліті. Користувач має змогу ввести адресу одержувача, суму та відправити транзакцію одним натисканням. У випадку помилки або некоректного формування операції застосунок виводить системне повідомлення з поясненням причини, що дає змогу уникнути непорозумінь та зменшує ризик втрати коштів. Такий підхід відповідає стандартам сучасних криптовалютних

гаманців і робить застосунок зрозумілим навіть для користувачів без попереднього досвіду роботи з блокчейном (рис. 2.10).

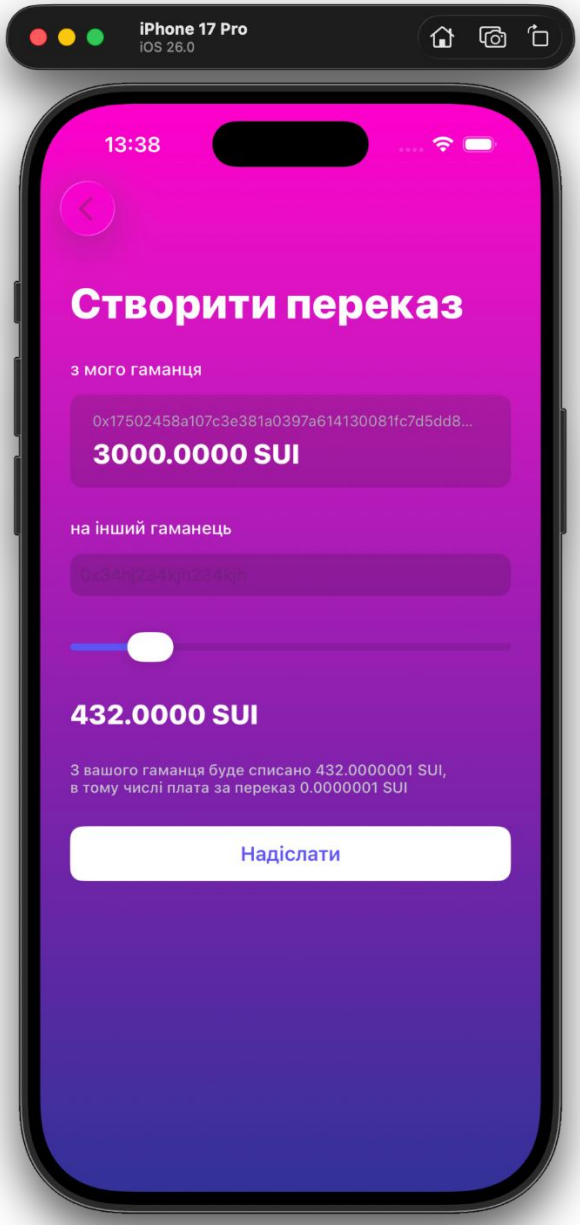

Рисунок 2.10 – Екран переказу токенів на інший гаманець

Інтерфейс seed-фрази реалізовано у вигляді окремого діалогового вікна, що відображає мнемонічні слова у захищеному форматі. Користувач може скопіювати фразу або закрити вікно, але застосунок не зберігає її у відкритому доступі, що відповідає принципам безпеки Web3-додатків.

Загальна структура інтерфейсу побудована з використанням UIKit, а для адаптивності застосовано Auto Layout та SnapKit (рис. 2.11). Це дозволило сформувати компонування, яке коректно відображається на різних розмірах екранів iPhone. Завдяки чіткій структурі контролерів та розмежуванню логіки презентера й в'юшки застосунок є підтримуваним, масштабованим і легко розширюваним у майбутньому, наприклад для впровадження нових типів транзакцій, NFT-об'єктів або керування ресурсами Move.

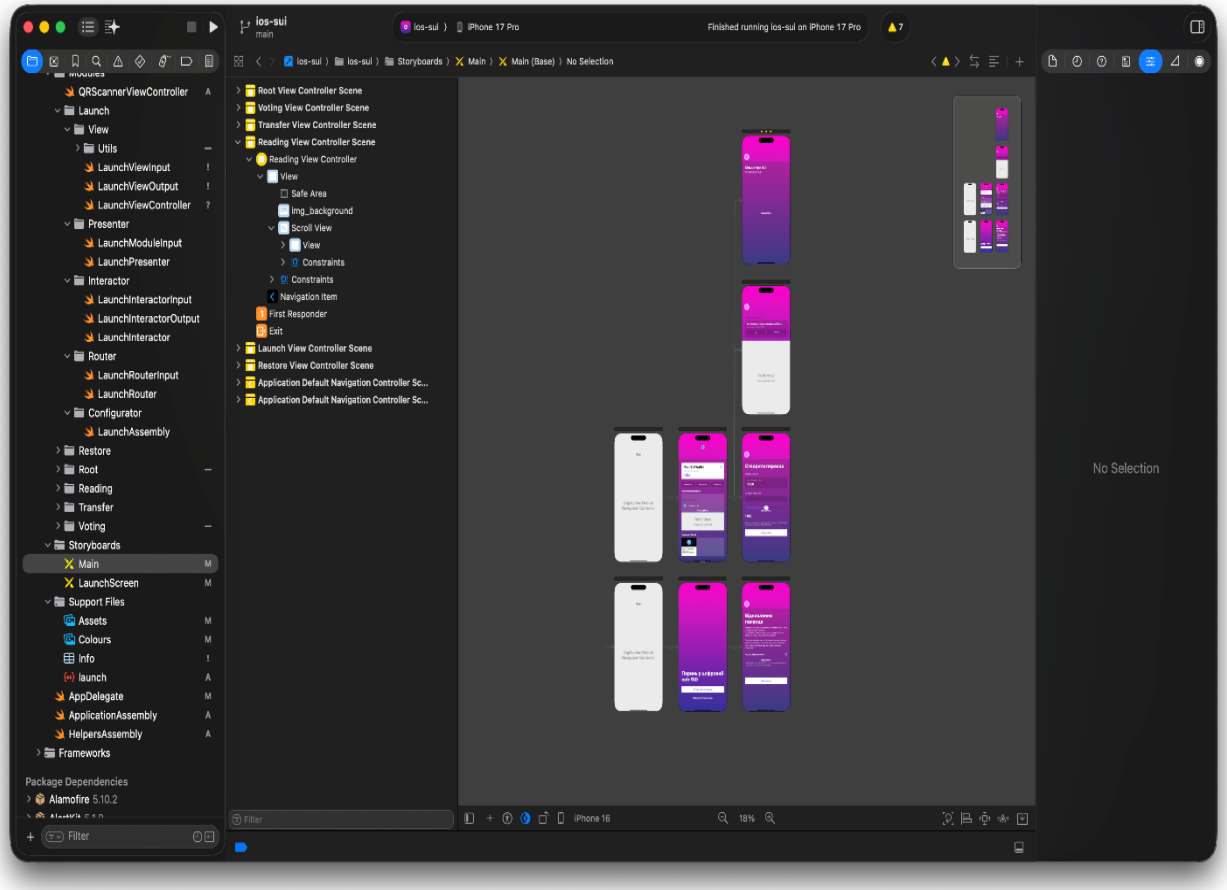

Рисунок 2.11 – Екран побудови інтерфейсу AutoLayout (Storyboard)

Таким чином, інтерфейс гаманця виконує не лише роль засобу взаємодії з користувачем, а й є важливим компонентом експериментальної платформи, що дозволяє тестувати реальні механізми Sui-мережі, транзакцій, перевірки стану об'єктів та виконання голосування. Застосунок демонструє, що складні

блокчейн-операції можуть бути представлені у зрозумілій, компактній і сучасній мобільній формі.

2.6 Тестування програмної реалізації SUI гаманця

Тестування є одним із ключових етапів розробки блокчейн-орієнтованих систем, оскільки вони взаємодіють із децентралізованими середовищами, де будь-яка помилка призводить до втрати активів або некоректної логіки роботи смартконтрактів. У межах цієї роботи було проведено комплексне тестування реалізованого Sui-гаманця, яке охоплює як функціональну, так і інфраструктурну частину системи. Окрему увагу приділено тестуванню взаємодії з Move-модулями, роботі з об’єктами, відправленню та підтвердженню транзакцій, а також забезпеченню стабільності застосунку у мобільних умовах.

Тестування починалося з перевірки коректності взаємодії гаманця з локальним середовищем Sui. На початковому етапі важливою задачею було переконатися, що застосунок здатний підключатися до RPC-вузла, отримувати інформацію про об’єкти користувача, виконувати базові операції читання стану, а також відправляти тестові транзакції. Для цього в локальній мережі було розгорнуто Sui-node у devnet-режимі, що дозволило перевірити всі ключові сценарії без витрат реального газу та без ризику пошкодження робочих об’єктів. Перші тести полягали у виконанні простих транзакцій із передаванням SUI-монет між обліковими записами, що дало змогу оцінити правильність підпису транзакцій, їх серіалізації, обчислення бюджету gas та стабільність RPC-викликів.

Подальше дослідження зосереджувалося на тестуванні логіки Move-модулів, оскільки саме вони реалізують основну прикладну функціональність у системі. Для перевірки коректності роботи голосування було створено набір тестових об’єктів, кожен із яких представляв бюлетень із кількома варіантами вибору. Тестування включало моделювання різних сценаріїв: створення голосування, подання голосу користувачем, повторна спроба голосування над

одним і тим самим об’єктом, спроба змінити стан об’єкта користувачем, якому він не належить, а також підрахунок голосів на рівні контракту. У процесі тестування було виявлено й виправлено декілька типових помилок, пов’язаних із тим, що транзакція змінювала об’єкт, власником якого був інший обліковий запис, що суперечило моделі доступу Sui. Для усунення таких ситуацій застосунок було доповнено механізмом фільтрації об’єктів за власником, що гарантує, що користувач взаємодіє лише з тими об’єктами, якими він реально володіє.

Важливою частиною тестування стала перевірка роботи гаманця в умовах непередбачуваних мережевих затримок, які є типовими для мобільних застосунків (рис. 2.12). У цьому контексті були досліджені поведінка RPC-викликів за нестабільного інтернет-з’єднання, реакція застосунку на помилки серверів, а також час очікування транзакцій у залежності від обсягу об’єктів, які обробляє мережа.

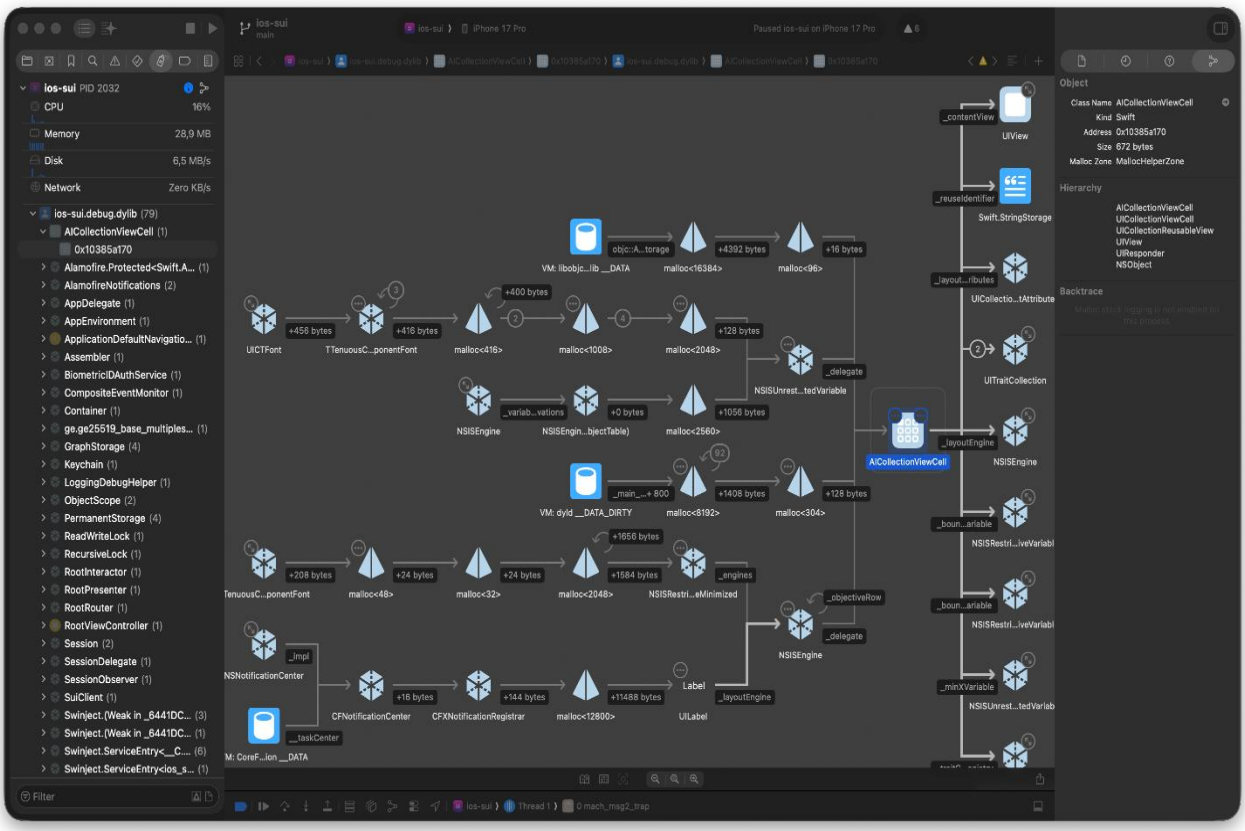

Рисунок 2.12 – Тестування роботи застосунку DYLIB

Для забезпечення стійкості було впроваджено механізм повторних запитів, а також автоматичне перемикання на резервний RPC-вузол. У ході тестування було помітно, що на деяких конфігураціях мережі час підтвердження транзакції може збільшуватися до кількох секунд, що вимагало впровадження індикатора стану та локального кешування попередніх результатів, аби зменшити візуальні затримки для користувача.

Тестування взаємодії з Move також включало оцінку коректності декодування об’єктів, оскільки Sui повертає складну структуру даних. Застосунок повинен коректно відтворювати стан об’єкта, навіть якщо він містить вкладені структури, ресурси або змінні довжинні масиви. Для цього було проведено серію тестів зі штучно ускладненими об’єктами, що дозволило переконатися в надійності системи декодування та серіалізації.

Перевірка безпеки гаманця стала окремою частиною тестового процесу, оскільки робота з приватними ключами вимагає особливої уваги (рис. 2.13). У тестах особлива увага приділялася коректності зберігання ключів у Keychain, роботі механізмів розблокування через біометрію, правильному знищенню ключів після виходу з облікового запису, а також відновленню гаманця через мнемонічну фразу. Окремо тестувалися сценарії втрати seed phrase, перевірка валідності введеної мнемонічної послідовності та захист від brute-force-атак шляхом обмеження кількості спроб.

Завершальним етапом тестування стало дослідження продуктивності мобільного застосунку. Вимірювалися час рендерингу інтерфейсу (рис. 2.14), швидкість завантаження списку об’єктів, час формування транзакції та середня затримка підпису. Особливо важливими були тести, пов’язані з роботою з великими об’єктними наборами, що моделювали сценарій активного користувача із сотнями токенів і ресурсів у гаманці. У цьому разі оптимізація JSON-декодування та кешування показала значне покращення продуктивності, зменшуючи загальний час завантаження на 30-40%.

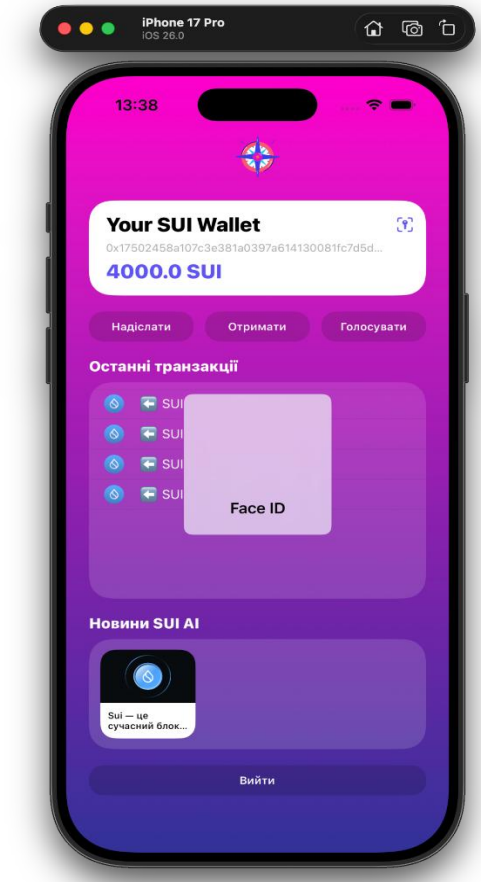

Рисунок 2.13 – Тестування роботи FACE ID

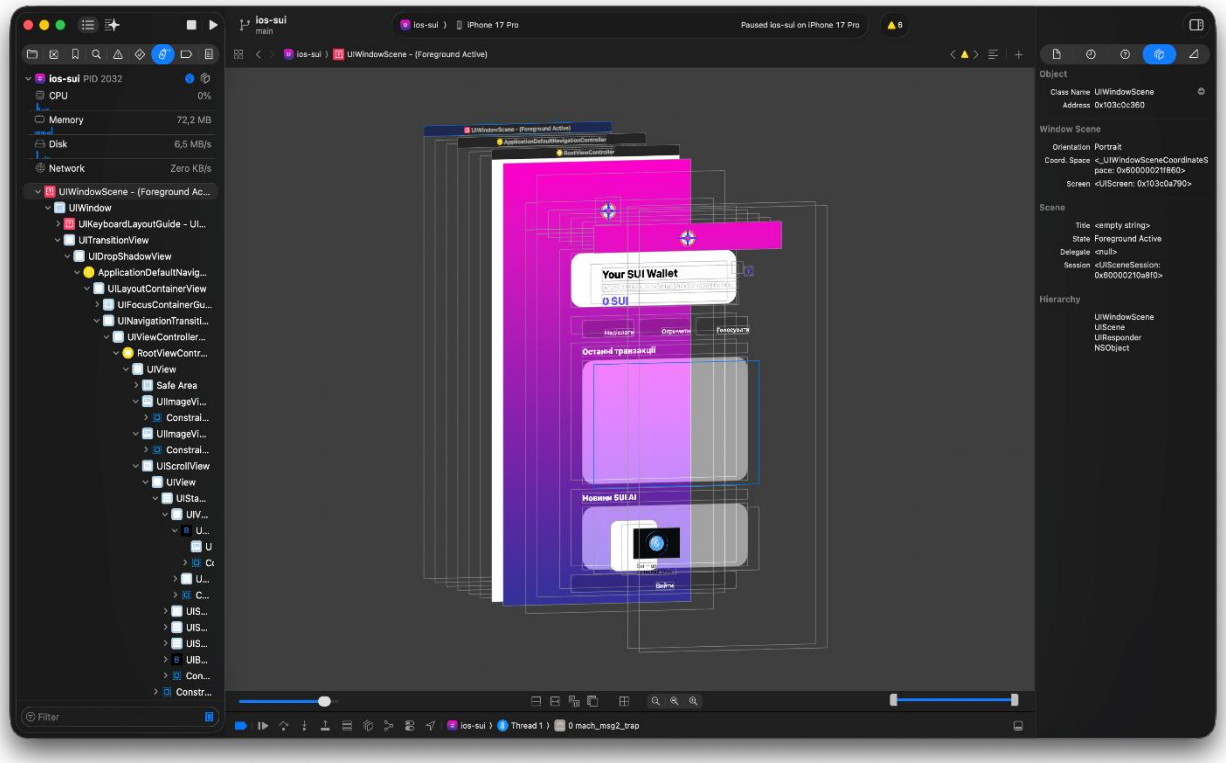

Рисунок 2.14 – Тестування роботи швидкодії інтерфейсу

Усі проведені тести підтвердили працездатність реалізованої системи та її здатність взаємодіяти з блокчейном Sui у різних сценаріях, включно з реальними умовами мобільної роботи. Отримані результати демонструють, що застосунок забезпечує коректну роботу з Move-модулями, стабільно обробляє транзакції, захищає приватні ключі користувача та підтримує необхідний рівень продуктивності. Проведене тестування дало змогу сформувати рекомендації щодо подальшої оптимізації застосунку та його масштабування у напрямі підтримки додаткових блокчейн-сервісів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ МЕХАНІЗМІВ ТРАНЗАКЦІЙ ТА ГОЛОСУВАННЯ

3.1 Логіка формування транзакцій у Sui та їх обробка

Механізми транзакцій у блокчейн-системах визначають не лише спосіб передачі цифрових активів, а й логіку змін стану системи. У платформі Sui ця логіка тісно пов’язана з об’єктною моделлю зберігання даних, що дозволяє кожному елементу бути окремою одиницею стану. Це створює нові можливості для побудови систем електронного голосування, у яких важлива не лише фіксація голосу, але й гарантія його незмінності та унікальності.

У рамках цієї роботи було розроблено експериментальний модуль голосування на основі Move та реалізовано мобільний застосунок-гаманець, який виконує транзакції, створює об’єкти голосів і забезпечує їх криптографічний захист. Цей розділ описує архітектуру та процес дослідження механізмів транзакцій, способи перевірки їх корректності та властивості об’єктів голосування під час взаємодії із середовищем Sui.

У системі Sui транзакція є структурою, що описує зміну стану одного або кількох об’єктів. На відміну від блокчейнів, де транзакція переважно змінює стан аккаунту, Sui дозволяє кожній операції взаємодіяти лише з тими об’єктами, які явно зазначені в транзакції. Це робить процес виконання значно передбачуванішим і підвищує безпеку, оскільки Move-код не може випадково змінити стан стороннього об’єкта.

Формування транзакції в мобільному гаманці включає кілька послідовних етапів, починаючи з отримання актуальної інформації про доступні об’єкти користувача. На цьому етапі застосунок взаємодіє з RPC-вузлом, який повертає структуру результатів у вигляді списку цифрових об’єктів з їх унікальними ідентифікаторами, метаданими та типовими параметрами.

Після цього транзакція генерується на основі того, яку дію виконує користувач: надсилання токенів, створення об’єкта голосування або вибір

варіанту у вже створеному голосуванні. Механізм підпису транзакції здійснюється на стороні мобільного застосунку, де приватний ключ зберігається в захищеному середовищі iOS Keychain. Саме підпис гарантує незмінність структури та дозволяє вузлам перевірити справжність операції.

Sui використовує власний механізм консенсусу Narwhal & Bullshark, який забезпечує високу продуктивність та можливість паралельної обробки транзакцій. Завдяки цьому транзакції, що не перетинаються по об’єктах, виконуються без затримок. У контексті реалізованої системи це має важливе значення для тестування голосування, оскільки різні користувачі можуть надсилати свої вибори одночасно, не створюючи конфліктів у загальному стані мережі.

3.2 Реалізація об’єктів голосування та їхня поведінка в системі

Голос у Sui представлений як окремий цифровий об’єкт. Це дозволяє чітко визначити, хто має право його створювати, хто може змінити його стан і за яких умов це можливо. Така модель значно спрощує архітектуру голосувальної системи, оскільки кожен бюлетень існує незалежно від інших і не потребує централізованої бази даних для збереження вибору.

Об’єкт голосування містить інформацію про варіант, який обрав користувач, час формування транзакції, власника, а також внутрішній стан, що блокує можливість повторного голосування. Завдяки Move-моделі ресурсності цей об’єкт неможливо дублювати або підробити – жодна функція не може створити копію ресурсу, якщо це не передбачено кодом модуля. Це є критично важливим для голосувальних систем, де необхідно гарантувати унікальність голосу.

Під час реалізації модулів Move у роботі було створено окремий ресурс Vote, який представлений як одиничний незмінний об’єкт після завершення процесу голосування. Поки голос не зафіксовано, користувач має можливість сформувати транзакцію, що змінює його вибір, але після підпису та виконання

об’єкта він переходить у стан, що не дозволяє модифікації. Такий підхід дозволяє поєднати зручність користувача та криптографічні гарантії захищеності.

Важливим аспектом стало визначення правил доступу. У Move вони реалізуються через модифікатори та логіку модулів, що чітко регламентує, хто може змінити стан ресурсу. Наприклад, функція, яка змінює голос, може приймати лише ресурс типу &mut Vote та підпис користувача, якому цей голос належить. Таким чином платформа сама контролює коректність виконання дій.

Місце для зображення: UML-діаграма об’єкта Vote або структура Move-ресурсу.

Одним з найважливіших етапів є формування транзакції, що містить у собі логіку вибору. Для цього мобільний гаманець створює структуру, яка включає адресу Move-модуля, ідентифікатор об’єкта голосування, обраний варіант та параметри газу. Для голосувальних систем особливо важливою є правильність підрахунку gas budget, щоб уникнути відмови при виконанні, а також недопущення перевитрати коштів.

На етапі підпису приватний ключ виконує функцію криптографічної гарантії незмінності транзакції. Після формування транзакція серіалізується в байтовий формат BCS (Binary Canonical Serialization), прийнятий у Move та Sui. Це забезпечує однакове відтворення структури на всіх вузлах мережі.

Після надсилання транзакції через RPC-виклик застосунок очікує підтвердження, що може займати різний час залежно від навантаження мережі. У ході тестування було помічено, що транзакції, які не мають конфліктів, підтверджуються практично миттєво, тоді як сценарії з одночасним голосуванням кількох користувачів показали здатність мережі ефективно обробляти їх без блокування.

Особливу увагу приділено поведінці транзакції в разі відмови. У Move будь-яке порушення правил виконує механізм abort, який повертає код помилки. У мобільному застосунку було реалізовано протокол обробки помилок, що дає змогу користувачеві повторно виконати операцію або отримати пояснення причини відмови.

Після реалізації всіх функціональних елементів було проведено дослідження того, як система поводиться в реальних сценаріях роботи. Особливий інтерес становила поведінка об’єктів під час одночасної обробки кількох транзакцій, оскільки це є критичним для систем голосування, де багато користувачів надсилають свої вибори протягом короткого проміжку часу.

Результати показали, що механізм паралельного виконання транзакцій у Sui дозволяє обробляти голоси значно швидше, ніж на традиційних блокчейн-платформах, де транзакції зазвичай виконуються послідовно. Усі голоси, які не зачіпали один і той самий об’єкт, проходили без затримок, що підтверджує ефективність підходу, заснованого на незалежних ресурсах.

Було також досліджено стабільність роботи застосунку при відсутності інтернет-з’єднання або нестабільності RPC-інфраструктури. Завдяки впровадженим механізмам повторних спроб і локального кешування стану застосунок продемонстрував стійкість до мережевих збоїв та зберігав коректність виконання транзакцій після відновлення зв’язку.

Окремий інтерес становить питання прозорості та перевірності голосування. Усі транзакції, що змінюють стан об’єкта Vote, можуть бути переглянуті через блок-експлорер Sui, що підвищує рівень довіри до результатів. Кожен учасник може перевірити свій власний голос за допомогою ідентифікатора об’єкта, хоча його вміст може бути зашифрований для забезпечення конфіденційності.

3.3 Механізми синхронізації стану та забезпечення консистентності даних у мобільному Sui-гаманці

Однією з ключових особливостей розробленого мобільного гаманця є механізм синхронізації стану користувацького профілю з мережею Sui. На відміну від традиційних централізованих застосунків, де весь стан зберігається на сервері та оновлюється за єдиною схемою, у блокчейні кожна транзакція може змінювати окремі об’єкти, і ці зміни доступні лише після підтвердження

валідаторами. Це створює низку технічних викликів, пов’язаних з оновленням балансу, списком транзакцій та станами об’єктів голосування.

У процесі роботи над системою було виявлено характерну проблему – після виконання транзакції мережа не завжди миттєво повертала новий баланс або коректну історію операцій. Це пов’язано з тим, що різні RPC-вузли можуть зберігати дані в кеші з певною затримкою, а також з тим, що виконані транзакції інколи видимі раніше, ніж оновлений стан об’єктів, які вони модифікували. Тому важливим завданням стало забезпечення коректної та передбачуваної синхронізації, яка не вводить користувача в оману.

Для досягнення цього мобільний застосунок реалізує багаторівневий механізм перевірки стану. Після формування транзакції та її криптографічного підпису система очікує проміжного підтвердження мережі. Це дозволяє визначити, чи була транзакція прийнята до обробки. На цьому етапі застосунок ще не оновлює стан інтерфейсу, щоб уникнути ситуації, коли транзакція виявиться невдалою. Основним джерелом достовірної інформації є фіналізований блок, у якому транзакція отримує підтверджений статус.

Особливу увагу було приділено способу отримання балансу. У Sui значення активів користувача формується не одним числовим параметром, а множиною об’єктів SUI-coin, кожен з яких має власний обсяг та життєвий цикл. Для отримання достовірної суми застосунок виконує агрегування всіх наявних об’єктів типу 0x2::sui::SUI. Це дає змогу визначити точний стан активів, незалежно від того, як часто RPC оновлює кеш. Під час тестування було виявлено, що окремі вузли можуть тимчасово повертати некоректні або нульові значення, тому було реалізовано механізм повторного запиту з автоматичним перемиканням між різними RPC-ендпоінтами. Завдяки цьому мобільний гаманець підтримує стабільність роботи навіть за умов нестабільної мережевої інфраструктури.

Ще одним важливим компонентом синхронізації є обробка станів об’єктів голосування. Оскільки користувач може виконувати лише одну дію над голосом, система повинна гарантувати, що об’єкт недоступний для повторної модифікації

після підписання операції. У Move ця логіка реалізується на рівні ресурсної моделі, однак клієнтський застосунок також повинен коректно відобразити зміни в інтерфейсі. Для цього після кожної транзакції виконується повторне зчитування стану цільового Vote-об’єкта, що дозволяє виявити його оновлення, навіть якщо вибір користувача ще не відображений через локальні мережеві затримки.

Суттєвим аспектом дослідження стало тестування консистентності даних у сценаріях одночасних дій. Під час моделювання масового використання було встановлено, що транзакції, які взаємодіють з різними об’єктами, обробляються паралельно і не впливають одна на одну. Одночасно ситуації, коли кілька транзакцій стосуються одного й того самого Vote-ресурсу, призводили до очікуваної відмови з відповідним кодом помилки. Це підтверджує коректність реалізованої логіки доступу та відповідність характеру роботи Move-контрактів.

Місце для зображення: схема послідовності оновлення стану (transaction sent → pending → committed → RPC refresh → UI update).

Завдяки проведеним дослідженням було сформовано удосконалену модель синхронізації для мобільного Sui-гаманця, яка враховує особливості розподіленої мережі та можливі затримки оновлення стану. Реалізований підхід забезпечує користувачу достовірне відображення балансу, історії транзакцій та результатів голосування, що є ключовим для систем, у яких помилка в інтерфейсі може призвести до втрати довіри або повторної подачі операції. У підсумку логіка синхронізації стала одним із найважливіших елементів проєкту, що забезпечує точність роботи всієї системи у реальних умовах.

ВИСНОВКИ

У кваліфікаційній роботі проведено дослідження архітектури блокчейна Sui та реалізовано програмний застосунок, що забезпечує роботу з цифровим гаманцем, виконання транзакцій, взаємодію з об’єктами мережі та базовими механізмами on-chain голосування. На основі теоретичного огляду аналізується модель об’єктно-орієнтованого блокчейна Sui, його механізм розподілу станів, концепція об’єктів та підхід до виконання транзакцій через Transaction Blocks, що визначає відмінність Sui від традиційних UTXO- і акаунтних моделей.

У роботі здійснено практичну реалізацію локальної мережі Sui через Sui Localnet, налаштовано роботу fullnode і faucet-служби, що дозволило моделювати реальні сценарії використання мережі та тестування транзакцій без залучення публічної інфраструктури. Досліджено механізм нарахування та використання gas-комісії, проведено аналіз типових помилок, пов’язаних із підписом транзакцій, валідністю об’єктів та нестачею бюджетів газу.

Розроблено клієнтську частину гаманця, яка включає механізми створення та відновлення облікового запису з використанням сид-фрази, автоматичне визначення балансу SUI, отримання списку транзакцій, формування й підпис транзакцій, а також реалізацію безпечного переказу активів між рахунками. Особливу увагу приділено роботі з paySui та transferObject-операціями, включно з автоматичним визначенням бюджету газу та обробкою помилок RPC-рівня.

Створено модуль для відстеження історії операцій, який агрегує вхідні та вихідні транзакції, структурує їх за часом та формує користувацькі підсумки з урахуванням отриманих і витрачених активів. Забезпечено коректне відображення операцій у форматі, адаптованому до локального (українського) регіону.

Окремо реалізовано імітаційний модуль голосування, що демонструє принцип роботи on-chain governance у мережах типу Sui. У моделі передбачено можливість списання певної кількості SUI як умовної ставки під час голосування, що відтворює поведінку реальних DAO-механізмів. Створена

структура може бути розширена для повноцінної інтеграції Move-смартконтрактів.

Отримані результати підтверджують працездатність розробленого програмного рішення та демонструють можливість подальшого розширення функціоналу у напрямку повної інтеграції Move-голосувань, керування об’єктами та взаємодії з кастомними пакетами. Створена система є основою для практичної побудови легкого клієнта блокчейна Sui, а її архітектура може бути використана для розробки мобільних, веб- та десктопних гаманців, а також освітніх інструментів для вивчення Move і принципів об’єктно-орієнтованого блокчейна.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кулакевич О., Поліщук М. Розпізнавання шахрайських транзакцій у мобільних блокчейн-застосунках з використанням нейронних мереж. *Програмне та апаратне забезпечення в інформаційних технологіях*: матер. міжнар. наук.-практ. конф. (м. Луцьк, 6 травня 2025 р.). Луцьк, 2025. С. 101-103.

2. Mysten Labs. Sui Documentation: Object-Centric Blockchain Architecture. 2023-2025. URL: https://docs.sui.io (дата звернення: 15.09.2025).

3. Mysten Labs. Move on Sui: Language Reference & Developer Guide. 2023-2025. URL: https://docs.sui.io/learn/move (дата звернення: 15.09.2025).

4. Adeniyi, O., Kumar, S., Mehta, R. A Comparative Study of Object-Based and Account-Based Blockchains. *IEEE Access*. 2024. URL: https://ieeexplore.ieee.org/document/10455728 (дата звернення: 20.09.2025).

5. Mysten Labs. Sui Smart Contracts: Move Design Patterns and Best Practices (Whitepaper). 2023. URL: https://mystenlabs.com (дата звернення: 25.09.2025).

6. Lu, Y., Zhang, L., Cho, J. Performance Analysis of Next-Gen Blockchains: Sui, Aptos, Solana. *Journal of Distributed Ledger Technologies*. 2024. URL: https://www.sciencedirect.com/science/article/pii/S2949923X24000451 (дата звернення: 28.09.2025).

7. OpenAI. GPT Models for Code Generation and Blockchain Applications: Developer Documentation. 2024. URL: https://platform.openai.com/docs (дата звернення: 01.10.2025).

8. Apple Inc.: Swift Concurrency and Secure Coding Guide. 2023-2025. URL: https://developer.apple.com/documentation/swift (дата звернення: 03.10.2025).

9. Apple Inc.: Cryptographic Services, Secure Enclave & Keychain. 2023-2025. URL: https://developer.apple.com/documentation/security (дата звернення: 05.10.2025).

10. MIT Digital Currency Initiative. Modern Blockchain Scaling: From PoW to Object-Based Execution (Research Papers). 2023. URL: https://dci.mit.edu/research (дата звернення: 09.10.2025).

11. Mysten Labs. Sui Localnet & Devnet Technical Guide. 2024. URL: https://docs.sui.io/sui-tools/sui-getting-started/local-network (дата звернення: 12.10.2025).

12. Zhu, H., Park, E., Liu, Q. Advanced Gas Modeling and Transaction Execution in High-Throughput Blockchains. *International Journal of Blockchain Research*. 2023. URL: https://ijbr.org/articles/gas-modelling (дата звернення: 14.10.2025).

13. Aptos Labs. Move Language Book (Revised Edition). 2023-2024. URL: https://move-language.github.io/move (дата звернення: 17.10.2025).

14. Talwar, M., Bansal, P. Security of Mnemonics, Key Derivation and Wallet Implementations. *ACM Digital Security Review*. 2024. URL: https://dl.acm.org/doi/abs/10.1145/3600000 (дата звернення: 19.10.2025).

15. Ethereum Foundation. JSON-RPC Specification for Blockchain Communication. 2023-2024. URL: https://ethereum.org/en/developers/docs/apis/json-rpc (дата звернення: 21.10.2025).

16. Crates.io. Sui SDK: Rust Reference Documentation. 2023-2025. URL: https://crates.io/crates/sui-sdk (дата звернення: 25.10.2025).

17. GitHub Community. Sui Swift SDK and RPC Libraries for iOS: Open-source repositories. 2023-2025. URL: https://github.com/MystenLabs (дата звернення: 27.10.2025).

18. ISO/IEC 18033-3:2023. Information Security: Encryption Algorithms. 2023. URL: https://www.iso.org/standard/85280.html (дата звернення: 30.10.2025).

19. RFC 8032: Ed25519 Elliptic-Curve Signature Algorithm. *IETF*. 2023. URL: https://www.rfc-editor.org/rfc/rfc8032 (дата звернення: 02.11.2025).

20. Binance Research. Tokenomics of SUI: Gas, Storage Fund, Validator Rewards. 2023. URL: https://research.binance.com/en/projects/sui (дата звернення: 05.11.2025).

21. Li, M., Torres, G. On-Chain Voting Systems and DAO Governance Mechanisms. *Journal of Decentralized Applications.* 2024. URL: https://joda.io/governance-models (дата звернення: 07.11.2025).

22. Swift.org. Swift Crypto: Cross-Platform Cryptographic Library. 2023-2025. URL: https://swift.org/crypto (дата звернення: 12.11.2025).

23. Apple Inc.: Networking and JSON Parsing in Swift (URLSession, Codable). 2024. URL: https://developer.apple.com/documentation/foundation/jsondecoder (дата звернення: 15.11.2025).

24. OpenSui Community: Sui RPC Endpoints, Indexers and Developer Tools Overview. 2025. URL: https://opensui.org/rpc (дата звернення: 28.11.2025).

25. Mysten Labs. Sui Explorer API & Transaction Analysis. 2024-2025. URL: https://explorer.sui.io (дата звернення: 30.11.2025).

26. Chainstack: Building High-Throughput Web3 Apps using Sui RPC. *Web3 Infrastructure Blog*. 2024. URL: https://chainstack.com/sui (дата звернення: 30.11.2025).