МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
15 грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія керування криптовалютного гаманця в блокчейні Ethereum»
здобувача групи ІН.м-23 Сергєєва Владислава Сергійовича

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

_____ Владислав СЕРГЄЄВ
(підпис)

Керівник,
асистент кафедри комп'ютерних наук,
к.ф.-м.н. Ольга ШУТИЛЄВА _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.м-23 Сергєєва Владислава Сергійовича

1. Тема роботи: «Інформаційна технологія керування криптовалютним гаманцем в блокчейні Ethereum»
затверджую наказом по СумДУ від «06» грудня 2023 р. № 1412-VI
2. Термін здачі здобувачем кваліфікаційної роботи до 15 грудня 2023 року
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми, постановка й формування завдань для області дослідження. 2) Огляд технологій, що використовуються в криптовалютних гаманцях. 3) Розробка криптовалютного гаманця в блокчейні Ethereum. 4) Аналіз виконаної роботи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» листопада 2023 р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми, постановка й формування завдань для області дослідження*		
2	*Огляд технологій, що використовуються в криптовалютних гаманцях*		
3	*Розробка криптовалютного гаманця в блокчейні Ethereum*		
4	*Аналіз виконаної роботи*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 55 стор., 23 рис., 1 додаток, 45 джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки інформаційна технологія керування криптовалютним гаманцем в блокчейні Ethereum відіграє ключову роль у забезпеченні безпеки, прозорості та ефективності управління цифровими активами. Цей напрям дослідження спрямований на розвиток інструментів та стратегій для оптимізації використання криптовалют, забезпечуючи користувачам надійний та зручний доступ до їх фінансових ресурсів.

Об’єкт дослідження – аналіз технічних рішень та стратегій для забезпечення безпеки та оптимального управління цифровими активами.

Мета роботи – розробка криптовалютного гаманця на блокчейні Ethereum.

Методи дослідження – дослідження та розбір даних, що вже існують, для вивчення їх структури, функцій та взаємозв'язків.

Результати – розроблено криптовалютний гаманець, який надає можливість створення та відновлення акаунтів. Відображає дані, щодо балансу користувачів та дозволяє створювати транзакції в мережі Ethereum.

БЛОКЧЕЙН, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, КРИПТОВАЛЮТНИЙ ГАМАНЕЦЬ, DAPP, ETHEREUM.

ЗМІСТ

ВСТУП

Актуальність. Сучасний світ стрімко розвивається вперед до цифрового майбутнього, роль інформаційних технологій набуває ключового значення. Одним із найбільш інноваційних та значущих напрямків в цій сфері є розробка криптовалют та інфраструктурних рішень на основі блокчейн-технології. Серед цієї всієї різноманітності даних, блокчейн Ethereum відрізняється своєю унікальністю та величезним потенціалом для розвитку.

Об’єкт дослідження. Аналіз технічних рішень та стратегій для забезпечення безпеки та оптимального управління цифровими активами.

Предмет дослідження. Методи керування та створення криптовалютного гаманця з урахуванням безпекової складової.

Гіпотеза. Основним елементом роботи з будь-якою криптовалютою є засіб зберігання та управління цифровими активами, тобто криптовалютний гаманець. Технології, що лежать в основі цього інструменту, є предметом ретельного вивчення та розробок для забезпечення максимальної безпеки, швидкості та функціональності, сучасності на найближчого майбуття. У контексті швидкого зростання інтересу до криптовалют, важливо розглядати інформаційні технології криптовалютних гаманців як складову, що забезпечує фінансову децентралізацію, перехід до цифрових форм власності та фінансових операцій.

Структура. Виконання дипломної роботи присвячено аналітичному огляду інформаційних технологій криптовалютного гаманця в блокчейні Ethereum. Зокрема, буде досліджено механізми зберігання приватних ключів, процеси генерації та верифікації транзакцій, а також забезпечення високого рівня конфіденційності та захисту від кібератак. Після чого реалізований власний приклад гаманця..

1. ІНФОРМАЦІНИЙ ОГЛЯД ТЕХНОЛОГІЙ КРИПТОВАЛЮТНИХ ГАМАНЦІВ

1.1 Актуальність тематики роботи

Життя сьогодення переходить в інтернет та цифровий світ, а з ним і наші гроші. Тут криптовалютні гаманці виявляються справді корисною річчю. Самі по собі вони дозволяють безпечно зберігати та керувати цифровими активами користувачів. Важливо, що контроль над ними власник здійснює самостійно, без посередників. І ще вони часто вписуються в ідею, коли можливо обходити бюрократичні моменти та самостійно у будь-який момент за необхідності виконувати перекази. Криптовалютні гаманці – це вже звичайний сучасний гаманець, який виходить за межі простого зберігання грошей, вони як ключ для входу в новий світ фінансів з власними правилами користування.

Навіть найбільш відома крипто-біржа Binance, нещодавно анонсувала запуск свого крипто-гаманця «Binance Web3 Wallet»[1]. Цей Web3-гаманець з технологією самостійного зберігання коштів запланований стати основним рішенням для мільйонів користувачів Binance. З його допомогою можна буде отримати доступ до безпечного і зручного методу обміну тисяч токенів у різних мережах за вигідними цінами. Крім того, він покращить користувацький досвід з децентралізованими застосунками (DApps), що дають змогу швидко переказувати кошти між біржою та гаманцем, на одній платформі.

Технологічний гігант Microsoft також заявив, що збирається інтегрувати крипто-гаманець в свою консоль Xbox[2]. На думку Майкла Зільберберга, керівника відділу по роботі з інвесторами в криптофонді AltTabCapital, такі рішення, щодо відкриття ігрового ринку, здатні збільшити об’єм в крипто-секторі на $2,5 млрд.

Не менш важливу роль криптовалюта та криптовалютні гаманці відіграють в житті багатьох українців, адже приблизно 12,7% населення України є власниками тієї чи іншої криптовалюти – це близько 5,6 млн осіб. А також з початку повномасштабного вторгнення частина донатів надходила

саме засобами криптовалюти. Навіть на сайті Міністерства цифрової трансформації України є стаття з переліком офіційних криптогаманців, на які можна перерахувати кошти для армії України[3].

1.2 Огляд існуючих рішень у сфері криптовалютних гаманців

На сьогоднішній день існує різноманіття криптовалютних гаманців, кожен з яких має свої переваги та недоліки. Провівши аналіз, можна відзначити, що більшість з них обмежена підтримкою лише однієї чи кількох криптовалют, пропонує зручний інтерфейс для користувача, але часто вимагає високих комісій за транзакції. Під час аналізу даних було вирішено зупинитися на трьох найбільш популярних: Coinbase, Metamask та Trezor. Розглянемо більш детально кожний з них.

1. Coinbase виділяється своїм зручним інтерфейсом, який спрощує процес купівлі та продажу криптовалют, особливо для початківців. Платформа також надає страхові поліси для покриття можливих збитків внаслідок злому чи інших порушень безпеки. Значний вибір криптовалют – ще одна перевага Coinbase. Проте, важливо відзначити, що ця платформа відома високими комісіями, а також незважаючи на розширений вибір, в порівнянні з іншими біржами, вибір на Coinbase може вважатися обмеженим. Важливо також відзначити відсутність анонімності на цій платформі, оскільки користувачам необхідно надавати особисті дані для проходження KYC та іншу інформацію перед здійсненням транзакцій, що робить їх відстеженням[4].

2. Trezor – це високооцінений гаманець, відомий своєю високою рівнем безпеки та анонімності[5]. Гаманець підтримує широкий спектр криптовалют, таких як Bitcoin, Ethereum і Litecoin, а також безліч альткоїнів. Одна з ключових переваг – зручний користувацький інтерфейс, який особливо корисний для початківців, спрощуючи їхню навігацію та управління криптовалютами. Треба також відзначити, що програмне забезпечення Trezor

є відкритим кодом, що дозволяє спільноті перевіряти та здійснювати його аудит, підвищуючи загальну прозорість та рівень безпеки.

3. Metamask є безпечним та легким у використанні децентралізованим криптовалютним гаманцем із підтримкою різних блокчейнів, таких як Ethereum, Polygon та Binance Smart Chain. Однак важливо відзначити деякі недоліки цієї платформи. По-перше, відсутність анонімності може викликати певні обмеження, оскільки транзакції можна відстежити, враховуючи, що MetaMask вимагає надання особистих ідентифікаційних даних та іншої інформації від користувачів. Крім того, залежність від розширень браузера становить проблему для користувачів Safari, оскільки MetaMask, як розширення, не підтримується цим браузером. Це може бути певним недоліком для користувачів, оскільки для ефективного використання гаманця потрібні певні технічні знання[6].

Враховуючи все вищесказане можна прийти до висновку, що розробка криптовалютного гаманця є важливою задачею для проведення криптовалютних транзакцій. А також сприятиме загальному розвитку подальшої взаємодії користувачів в децентралізованій мережі.

1.3 Постановка задачі

1. Літературний огляд:

- Провести докладний літературний огляд за темою, включаючи ключові публікації та наукові дослідження в галузі криптовалют, блокчейн-технологій та керування гаманцем в Ethereum.
- Визначити актуальні проблеми та виклики, що стосуються інформаційної технології керування криптовалютним гаманцем.

2. Аналіз методик управління гаманцем:

- Розглянути різні підходи до керування криптовалютним гаманцем в блокчейні Ethereum.
- Проаналізувати методи шифрування та засоби захисту від несанкціонованого доступу до гаманця.

3. Створення смарт-контракту для гаманця:

- Розглянути варіанти використання смарт-контрактів в Ethereum для управління криптовалютним гаманцем.
- Дослідити можливості автоматизації операцій та розширення функціональності гаманця через смарт-контракти.

4. Визначення процесу керування гаманцем:

- Розробити базовий порядок дій для створення та ефективного управління криптовалютним гаманцем в Ethereum.
- Включити етапи встановлення гаманця, забезпечення безпеки та використання смарт-контрактів для оптимізації операцій.

5. Тестування технології:

- Розробити та реалізувати тестовий сценарій для перевірки ефективності інформаційної технології управління гаманцем.
- Провести тестування транзакцій та безпеки системи.

6. Висновки та рекомендації:

- Сформулювати висновки на основі результатів дослідження.
- Надати рекомендації для подальшого розвитку та вдосконалення інформаційної технології керування криптовалютним гаманцем в блокчейні Ethereum.

2. ОСНОВНІ ПОЛОЖЕННЯ ДЛЯ ТЕХНОЛОГІЙ УПРАВЛІННЯ КРИПТОВАЛЮТНИМ ГАМАНЦЕМ

2.1 Аналіз методів створення та роботи з криптовалютними гаманцями

Для всіх власників криптовалюти або токенів NFT актуальним є питання зберігання своїх активів. Тому важливо розуміти, які засоби зберігання криптоактивів існують.

Як уже було зазначено раніше, гаманець – це інструмент, за допомогою якого власник взаємодіє зі своїми активами[7]. Гаманець може бути як програмою, так і фізичним пристроєм. Він необхідний для зберігання, накопичення, контролю та витрати цифрової валюти.

Важливо розуміти принципову відмінність звичайного гаманця, в якому ми зберігаємо свої паперові гроші від криптовалютного. У криптовалютному гаманці активи не зберігаються. Цифровий токен, криптовалюта, NFT – все це записи в блокчейні, а гаманець за допомогою певних механізмів дозволяє отримати доступ до цих записів та змінювати їх за бажанням власника.

Одна з функцій гаманця генерувати дані, необхідних проведення транзакцій. Цими даними є ключі, приватні та публічні. Таким чином, гаманці можна поділити на два види: кастодіальні і некастодіальні.

Кастодіальним гаманцем вважається той, у якому приватний ключ зберігається на стороні сервісу, який надає гаманець, тобто кастодіан (охоронець). Найпростіший приклад кастодиального гаманця – зберігання коштів на біржі[8]. Переваги зберігання на кастодіальному гаманці:

- Полегшений доступ. Легко управляти своїми криптовалютними активами, просто використовуючи кілька натискань кнопок на безпечному та надійному пристрої.
- Втрата приватного ключа або мнемонічної фрази не призведе до негайної втрати коштів. У будь-який момент можна ініціювати процедуру

відновлення доступу. Це може зайняти певний час і вимагати певного відступу від традиційної анонімності, але це варто робити, щоб уникнути повної втрати.

- Існує можливість скасування певних транзакцій та повернення коштів, хоча всі операції в межах блокчейну зазвичай вважаються незворотніми. Несприятливі помилки користувачів можуть бути врегульовані за допомогою деяких механізмів, але без участі посередника шанси на успішне відновлення обмежені. Тим не менш, деякі сервіси готові надати користувачам певну компенсацію в разі скоєних ними помилок[9].
- Низька комісія та висока швидкість транзакцій.

Недоліки кастодіального зберігання

- Кастодіан контролює засоби під час використання гаманців.
- Якщо провайдер гаманця вимкнеться, доступ до коштів буде втрачено.
- Влада може обмежити діяльність кастодіану, сервери можуть вилучатися. Кастодіан часто централізований і схильний до державного регулювання.
- Немає можливості використовувати закритий ключ для доступу до своїх засобів з інших гаманців.
- Втрати платформи, на якій зберігаються засоби, наприклад, внаслідок злому хакерами, можуть безпосередньо вплинути на користувачів.
- Повільне виведення коштів.
- Неможливість доступу до засобів під час технічних робіт.

Некастодіальний гаманець – це той, приватний ключ якого, зберігається у власника гаманця[10]. Тобто, сервіс, який надав некастодіальний гаманець, не має доступу до приватного ключа.

Головною перевагою некастодіального гаманця є повний контроль активів власника. Недоліки також є. Якщо втратити приватний ключ або seed-фразу, це спричинить втрату коштів. Некастодіальний гаманець також може бути зламаний хакерами.

Також гаманці можна розділити на 2 категорії: холодні та гарячі.

Таблиця 2.1 – Порівняння гаманців за категоріями

Холодний гаманець	**Гарячий гаманець**
Офлайн пристрій, який зберігається фізично у власника	Повністю зберігається в інтернеті
Отримати доступ можна лише фізично	Вразливий до фішингових та хакерських атак
Займає більше часу для проведення транзакції	Інтернет з’єднання робить онлайн транзакції швидшими
Зазвичай коштують від 50$ до 200$	У більшості випадків безкоштовні або мають плату за користування

Слід згадати ще таку окрему категорію, як паперові гаманці. Паперовими гаманцями користувалися на самому зародженню криптовалют їх суть заключалася в тому, що користувачі роздруковували якісь дані транзакції. До них увійшли ключі та QR-коди, щоб гаманці на мобільних пристроях могли їх сканувати. Однак паперові гаманці легко пошкодити або втратити, тому багато власників криптовалюти більше не використовують їх[11].

З Таблиці 2.1 можна зробити висновок, що гарячі гаманці підходять більше для користувачів, які активно проводять транзакції та всяко використовують криптовалюту. Холодні гаманці в свою чергу підходять для надійного збереження великих об’ємів криптовалют з майже неактивним користувачем.

Безпека активів у гаманці нерозривно пов'язана із існуванням публічного та приватного ключа. Ці ключі дозволяють створити важливі елементи гаманця: адресу та цифровий підпис, які дозволяють здійснювати транзакції.

Під час реєстрації гаманця спочатку генерується приватний ключ. На основі приватного генерується публічний ключ. Знаючи приватний ключ, можна легко визначити публічний. Підібрати приватний, знаючи публічний ключ не вдасться. Приватний ключ є ключем, який дає доступ до активів у

гаманці[12]. Тому варто акцентувати увагу на приватному ключі не можна передавати ніколи і нікому. Це рівнозначно втраті власних коштів.

Публічний ключ дозволяє створити адресу – ідентифікатор, який по своїй суті схожий на номер рахунку у банку. Називаючи свою адресу, користувач дає можливість іншій людині однозначно ідентифікувати цю адресу та відправити на неї криптовалюту. Публічний ключ можна передавати стороннім людям, це платіжний реквізит, за яким цільовий одержувач отримає кошти на свій гаманець[13].

На сьогодні замість використання приватного ключа як інструменту захисту гаманця було розроблено інший спосіб. Він ґрунтується на використанні слів замість складної комбінації цифр та букв. Ця комбінація називається seed-виразом. Можна зустріти також назви: мнемонічний вираз, відновна фраза чи просто seed.

Більшість криптовалютних гаманців сьогодні підтримують seed-вираз – це послідовність з 12, 18 чи 24 слів. Наприклад: yogurt afford beta fit failure vast down youth steel vacuum car skirt.

Вираз генерується на момент створення гаманця і залишається незмінним увесь час його існування. Seed-вираз доступний тільки власнику гаманця, розробник програми або будь-хто інший не мають до неї доступу. Не зайвим буде згадати, що його нікому і за жодних умов не можна розголошувати. При створенні гаманця потрібно зафіксувати у надійному місці. Носій, на якому зафіксовано копію seed-виразу, не повинен підключатися до Інтернету. Флеш-накопичувач, аркуш паперу, гравіювання на металі, ось кілька поширених способів зберігання. Якщо його втрачено, то відновити вже не вдасться. Виникає питання, чи можливо підібрати seed-вираз, наприклад, за допомогою спеціального програмного забезпечення? Якщо коротко ні. Фахівці стверджують, що при сучасних технічних можливостях на підбір seed-виразу з дванадцяти слів потрібен час, який можна порівняти до нескінченності[14-15].

2.2 Технології та методи безпеки в криптогаманцях

У сучасному цифровому світі, де криптовалюти швидко набувають популярності, питання безпеки використання криптовалютних гаманців стає вельми актуальним. Криптографічна безпека, яка лежить в основі цих гаманців, визначає весь спектр заходів, спрямованих на захист фінансових активів і особистої інформації користувачів.

Захист власних цифрових коштів починається з вибору правильного криптовалютного гаманця. Різноманітність гаманців, від онлайн-платформ до апаратних пристроїв, вносить свої особливості та виклики у сферу безпеки. Важливо розуміти, яким чином різні типи гаманців гарантують захист ваших активів від потенційних загроз.

Однак безпека в криптовалютних гаманцях не обмежується лише їх вибором. Такі аспекти як, безпечне зберігання приватних ключів, двофакторна аутентифікація та захист від шахрайства, стають ключовими елементами безпечного управління криптовалютами. Тому вивчення та розуміння найсучасніших підходів для забезпечення безпеки в криптовалютних гаманцях є необхідним для кожного власника цифрових активів.

Основною функцією гаманця є захист приватного ключа. Це пояснюється тим, що у світі блокчейнів будь-хто, хто володіє закритим ключем, є справжнім власником цифрового активу, тож якщо приватний ключ витік, це означає, що право власності на криптовалюту втрачено. Тому безпека приватних ключів важлива для користувачів і гаманець насправді це програма, що управляється закритим ключем[16].

На даний момент існує дві основні форми гаманців. Одним з них є повновузловий гаманець. До них можна віднести Bitcoin-Core, Geth, Parity тощо. Цей тип гаманця не надає інтерфейс безпосередньо назовні. Теоретично він найбезпечніший, але недоліки теж очевидні. Потрібно підтримувати величезні дані блокчейну, і дані повинні бути синхронізовані до останньої

версії. Якщо ж не дотримуватися цих правил це призведе до неможливості запитати статус транзакції або навіть неможливості торгувати[17-18].

Інший вид гаманців називається легким[19], такий як загальновідомий MetaMask, зручний для більшості користувачів. Більшість мобільних гаманців та веб-гаманців є гаманцями такого типу. Вони не потребують збереження величезних блоків даних. Для роботи з ним потрібно знати лише приватний ключ, за допомогою якого підписується транзакція, а потім надсилається до повного вузла. Звичайно, для цього необхідно мати перевірений повний вузол, відкритий для цієї функції і готовий надавати ці послуги[20].

Наразі більшість програмного забезпечення для гаманців, яке надають біржі, все ще використовують паролі для перевірки користувача, а також використовує паролі користувачів для шифрування приватних ключів, що зберігаються в гаманцях. Користувач входить до гаманця через перевірку пароля, після чого користувач може виконувати перекази. Таким чином існує ризик витоку закритого ключа. Як тільки пароль користувача витік, це призводить до витоку приватного ключа, що означає, що користувач втратив право власності на цифрові активи в блокчейні. На відміну від інших систем управління, немає способу повернути право власності на закриті ключ. Ситуацію робить гіршим те, що багато користувачів зазвичай налаштовують автоматичне зберігання пароля, що дає зловмисникам більше можливостей отримати закриті ключі[21].

Тому для покращення безпеки були придумані додаткові способи верифікації, зокрема безперервний підхід автентифікації[22] на основі поведінкової біометрії. Завдяки такому підходу особистий ключ користувача можна захищати в режимі реального часу. Раз гаманець виявляє, що ідентифікаційні дані користувача не збігаються, він може негайно вийти або запобігти транзакції продовження, тим самим ефективно запобігаючи витоку приватного ключа користувача. Цей підхід є на основі біометричних даних поведінки миші користувача та мережі CNN[23]. Коли людина реєструється в гаманці як користувач, постачальник спочатку збирає дані про поведінку миші

користувача, потім використовує мережу CNN для навчання біометричної моделі користувача. Модель навчального процесу може бути на локальній машині користувача або на сервері, щоб збільшити швидкість навчання. Після цього біометрична модель користувача і закритий ключ поєднані та зберігаються в гаманці. Гаманець постійно перевіряє ідентичність користувача через збережену модель, що може ефективно запобігти витоку закритого ключа[24].

Також більшість існуючих криптовалютних гаманців стикаються з проблемою недостатньо безпечного та зручного процесу резервного копіювання та відновлення. Існують рішення засновані на криптографічній схемі для безпечного резервного копіювання апаратних гаманців використовують візуальну перевірку гаманця людиною[25]. Також є пропозиції, щодо додаткового механізму захисту коштів такого як, розподіл грошей між двома гаманцями – з невеликою та великою сумами. Для цього створюється схема ієрархічних гаманців, яка називається детермінованим підгаманцем[26]. Це дозволяє користувачеві безпечно пересилати кошти з гаманця з великою сумою на менший.

Враховуючи потреби безпеки криптовалютних гаманців, є багаторівневі архітектури на основі стратегії Defense-in-Depth[27]. Ця стратегія забезпечує захист приватних ключів на трьох різних рівнях, використовуючи різноманітні механізми захисту. Особливу увага приділена рівновазі між зручністю використання та безпекою.

Окремо існують рішення для ще однієї важливої проблеми у сфері криптовалют – втрата доступу до закритих ключів, що призводить до повної втрати акаунту. Для рішення подібних проблем існує механізм, під назвою lean recovery транзакцій[28], він дозволяє змінювати ключі гаманця для створення транзакції та відновлення за необхідності.

2.3 Вузли блокчейну

Вузли блокчейну виступають як невід'ємна частина цифрової інфраструктури, що забезпечує децентралізовану і надійну систему реєстрації та обробки транзакцій. Кожен вузол представляє собою самостійний комп'ютер або сервер, який здатний взаємодіяти з іншими вузлами у мережі. Завдяки принципу децентралізації, вузли блокчейну спільно управляють базою даних, забезпечуючи необхідний рівень консенсусу та відсутність централізованого контролю.

Кожен вузол мережі блокчейну має свою роль у підтримці цілісності системи. Вони допомагають у верифікації та обробці нових транзакцій, а також у забезпеченні безпеки мережі за допомогою криптографії. Така архітектура вузлів блокчейну створює унікальну модель, де жоден окремий вузол не є владарем, але вони спільно працюють над створенням імовірності консенсусу та безпеки даних.

Для повного розуміння необхідно розібратися більше з тим, що таке вузли взагалі.

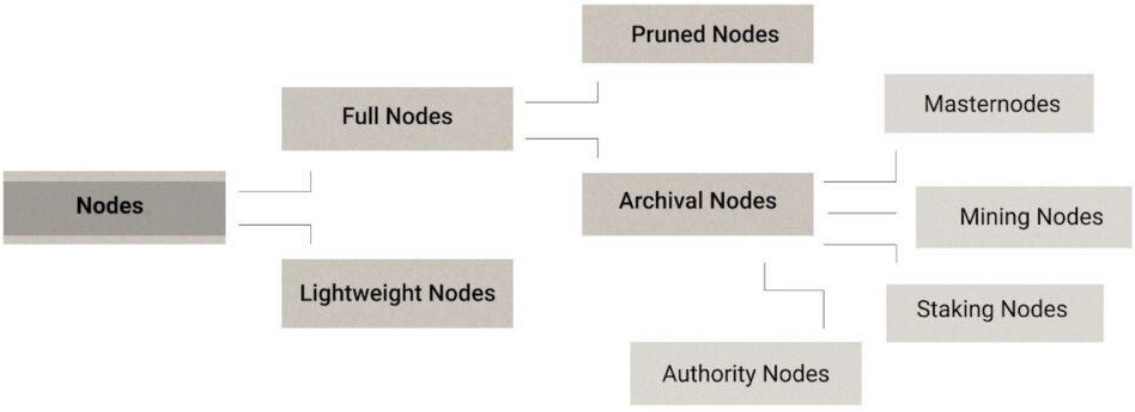


Рисунок 2.1 – Види вузлів

На рисунку 2.1 зображено ієрархію вузлів в блокчейні. Вони є зацікавленими сторонами мережі, які уповноважені відстежувати та слугувати комунікаційними центрами для різних мережевих завдань. Основне завдання вузла — підтвердити законність кожної наступної партії транзакцій, відомих

як блоки. А також, призначити унікальний ідентифікатор кожному вузлу в мережі, що допомагає легко відрізнити вузол від інших вузлів[29].

Таблиця 2.2 – Функції вузлів

Функція	Призначення вузлів
Обробка даних	обробляють і маніпулюють даними різними способами.
Комунікація	полегшують взаємодію між пристроями.
Управління зберіганням	ефективно зберігають дані та керують ними.
Контроль та рішення	приймають рішення та контролюють роботу системи.
Маршрутизація	визначають шляхи передачі даних.
Безпека	забезпечують безпеку даних і мережі.
Моніторинг	контролюють продуктивність і стан системи
Інтерфейс користувача	забезпечують інтерфейси для взаємодії з користувачем

Розглянемо більш детально по кожному виду вузлів.

Повні вузли є найважливішим типом вузла в мережі Blockchain, оскільки вони зберігають повну копію книги Blockchain. Ці вузли завантажують і зберігають копію кожної транзакції та блоку в мережі, що дозволяє їм незалежно перевіряти всю історію Blockchain. Повні вузли є основою мережі Blockchain і мають важливе значення для підтримки її цілісності[30].

Легкі вузли, є більш полегшеною версією повних вузлів. Вони призначені для роботи на пристроях з обмеженою пам’яттю та обчислювальною потужністю, таких як смартфони та планшети. Легкі вузли завантажують не весь блокчейн, а невелику його частину, яка містить інформацію, пов’язану з їхніми транзакціями[31].

Майнінгові вузли відповідають за перевірку транзакцій і додавання нових блоків до Blockchain[32]. Ці вузли виконують складні обчислення для

вирішення математичних завдань, які дозволяють їм створювати нові блоки та отримувати винагороду у вигляді криптовалюти.

2.4 Блокчейн Ethereum

Засновником Ethereum є Віталій Бутерін. Він проявив зацікавленість біткоінами під впливом свого батька, після чого віддав свою увагу вивченню та аналізу цієї криптовалюти. Внаслідок своїх досліджень він почав активно заходити в сферу криптовалют та біткоін-галузі, публікуючи статті про обмін криптовалюти. Спільно з іншим колегою, він заснував Bitcoin Magazine[33], відзначивши важливі події та тенденції у цьому сегменті.

На етапі розвитку своїх ідей Бутерін визначив потенціал біткоїна, який можна було б значно розширити, перетворивши його функціонал за межі простого обміну валюти. Відходячи від концепції біткоіна як цифрового еквівалента банку, Бутерін пропонує уявити банк як комп'ютер із трьома базовими інструкціями: внесення коштів на рахунок, зняття коштів та передача коштів на інший рахунок.

У технічному контексті це можна розглядати як банк у вигляді комп'ютерної мережі з програмно-визначеними інструкціями, що дозволяють автоматизовано виконувати різноманітні операції. На прикладі Ethereum, який був винайдений та запущений Бутеріном для розширення можливостей біткоїна, відзначається можливість запускати будь-які інструкції обробки, а не лише обміну валютою. Ethereum пропонує спрощену та ефективну інфраструктуру для створення та виконання унікальних інструкцій, які можуть бути застосовані до фінансових операцій, а також до складних бізнес-процесів, таких як розподіл доходів та управління фінансами підприємств.

Коли мережа Ethereum була запущена в 2015 році, це стало першим проєктом, який розширив варіанти використання блокчейнів шляхом впровадження унікальних технологій, які дозволили людям створювати власні цифрові токени та самопідтримувані, автономні програми[34]. Ця інновація

проклала шлях для широкого спектру ринків, включаючи децентралізоване фінансування (DeFi), первинні пропозиції монет (ICO), GameFi та незамінні токени (NFT).

На високому рівні Ethereum складається з кількох ключових компонентів:

- Розумні контракти: правила, що визначають, за яких умов гроші можуть переходити з рук в руки[35].
- Блокчейн Ethereum: запис усієї історії Ethereum – кожна транзакція та виклик смарт-контракту зберігаються в його блокчейні.
- Механізм консенсусу: метод перевірки та запису даних у блокчейні; він також допомагає захистити мережу та відповідає за випуск нових токенів в обіг[36].
- Віртуальна машина Ethereum (EVM): частина Ethereum, яка виконує правила Ethereum і забезпечує відповідність надісланої транзакції або смарт-контракту правилам[37].
- Ether: токен Ethereum, необхідний для здійснення транзакцій і виконання смарт-контрактів на Ethereum.

Ethereum має певну схожість з біткойнами в тому, що він покладається на блокчейн для зберігання та захисту транзакцій.

Блокчейн – це ланцюжок упорядкованих у хронологічному порядку блоків, що містять дані підтверджених транзакцій. Уявіть це як бухгалтерську книгу, де записуються всі дії, що виконуються в мережі або на платформі. Важливо, що ця книга є загальнодоступною, тобто учасники мережі та навіть сторонні особи можуть легко відстежувати її вміст. Крім того, копії цієї книги поширюються в глобальній мережі комп’ютерів, відомих як «вузли»[38].

Ця архітектура дозволяє учасникам володіти копією блокчейну та колективно перевіряти дійсність доданого до нього вмісту. Деякі з переваг цього включають:

- Жодної точки відмови
- Дані абсолютно прозорі, надійні та незмінні

- Стійкий до цензури

Однак відмінність Ethereum від Bitcoin полягає в тому, що вузли мають не лише перевіряти та записувати дані транзакцій, але й відстежувати «стан» мережі.

Стан Ethereum – це поточна інформація про всі запущені на ньому додатки, включаючи баланс кожного користувача, увесь код смарт-контракту, де він зберігається та будь-які внесені зміни[39].

Раніше Ethereum і Bitcoin використовували той самий консенсусний протокол для перевірки даних і додавання їх до блокчейну – відомий як підтвердження роботи (PoW). Це включає майнінгові вузли, які змагаються один з одним за допомогою енергоємних машин, щоб отримати право додати наступний блок до блокчейну. Це відбувається приблизно раз на 10 хвилин.

Однак у 2022 році Ethereum зазнав значного переходу, відомого як «The Merge»[40], який переніс мережу на блокчейн із підтвердженням частки (PoS).

Замість того, щоб вимагати від майнінгових вузлів використання дорогого обладнання для виявлення нових блоків, нова система PoS вимагає від користувачів внести та заблокувати 32 ефіру – рідної криптовалюти Ethereum, щоб стати валідаторами мережі.

Є три основні переваги переходу:

- Новий блокчейн PoS Ethereum підтримує реалізацію нових «ланцюжків фрагментів». Це будуть 64 менші блокчейни, кожен з яких оброблятиме свої власні пакети даних, що дозволить Ethereum обробляти значно більше транзакцій за секунду.
- Новий блокчейн Ethereum споживає на 99,95% менше енергії, ніж версія з підтвердженням роботи.
- Оскільки валідаторам не потрібно буде купувати та використовувати дороге обладнання для майнінгу, це зменшить бар’єр для входу людей до мережі. Це має допомогти покращити загальну децентралізацію та захист мережі.

EVM — це нативна система обробки Ethereum, яка дозволяє розробникам створювати смарт-контракти та дозволяє вузлам легко з ними взаємодіяти. Розробники Ethereum пишуть розумні контракти за допомогою Solidity, мови програмування, схожої на Javascript і C++. Ці розумні контракти, написані в Solidity, можуть читати люди, але не комп'ютери. Тому його потрібно перетворити на низькорівневі машинні інструкції – так звані коди операцій – які EVM може легко зрозуміти та виконати[41]. Важливо знати, що кожен вузол Ethereum має власну EVM.

Коли користувач надсилає транзакцію в смарт-контракт, розгорнутий на Ethereum, кожен вузол запускає смарт-контракт і транзакцію через власну EVM. У цьому змодельованому середовищі кожен вузол може побачити, яким буде кінцевий результат і чи буде в результаті дійсна транзакція чи ні. Якщо всі вузли досягають однакового дійсного результату, зміни вносяться, а оновлений стан Ethereum записується в блокчейн.

Ether потрібен для будь-яких дій в Ethereum, і коли він використовується для виконання розумних контактів у мережі, його часто називають «газом». Обсяг газу, необхідний для оплати, визначається типом транзакції, яку ви плануєте виконати, і кількістю транзакцій Ethereum, які очікують перевірки. Чим складніша транзакція, тим вище плата за газ.

Як згадувалося раніше, деякі користувачі взаємодіють з Ethereum через вузли. Іншими словами, вони повинні підключити свої комп'ютери до Ethereum, завантаживши програмне забезпечення блокчейну у свою систему. Залежно від обраного програмного клієнта Ethereum вам може знадобитися завантажити повну копію блокчейну Ethereum.

Крім того, користувач може просто згенерувати закритий ключ і створити адресу гаманця, щоб почати взаємодіяти з блокчейном. Гаманець у цьому контексті відноситься до цифрових або фізичних пристроїв зберігання, призначених для криптовалют.

І хоча більшість припускає, що власники криптовалюти зберігають свої цифрові активи в гаманцях, ці програми та пристрої функціонують як системи

зберігання особистих ключів. Ви не залишаєте блокчейн Ethereum. Натомість блокчейн призначає всім власникам ефіру приватний ключ, який дозволяє їм отримати доступ до свого балансу ефіру та використовувати його як заманеться. Коли користувач передає ефір, блокчейн оновлює баланс, щоб відобразити зміну права власності на передані монети.

Таким чином, Ethereum визначається не лише валютою Ether (ETH), але й можливістю виконання різноманітних інструкцій, розширюючи функціонал біткоїна та надаючи відмінні можливості для програмно-визначених операцій в крипто-середовищі.

2.5 DApp

Тепер можна відокремити додаток від окремої компанії чи власника та створити «децентралізований додаток», також відомий як скорочення DApp. Його можна вимовляти з двох складів як «dee-app» або з одного складу як «dapp».

Децентралізовані програми часто описують як однорангові з відмітною характеристикою, що немає жодного сервера чи організації, яка б ними керувала, як у моделі клієнт-сервер. Це відмінні властивості смарт-контракту та гнучкість платформи. Але щоб зрозуміти, чим DApp насправді відрізняється від централізованої програми, варто розглянути, що входить до складу сучасної централізованої програми. Прототип сучасного програмного забезпечення включає принаймні один інтерфейс користувача; це може бути мобільний додаток, завантажений із магазину додатків, веб-сайт до якого можна отримати доступ із комп’ютера чи мобільного пристрою або настільний додаток, встановлений на комп’ютері.

Зазвичай це стосується даних. Ці дані можуть бути надані однією групою чи компанією, як-от програма погоди за допомогою національної метеорологічної організації, або, як у програмі соціальної мережі, їх можуть

надати самі кінцеві користувачі. Нарешті, це передбачає певне маніпулювання даними або обчисленнями.

DApp використовує блокчейн в основі зберігання та обробки даних. Це реалізується за допомогою смарт-контракту. Зараз інтерфейс користувача для DApp зазвичай створюється за допомогою традиційної моделі веб-сайту.

Таким чином, повний DApp можна розглядати як веб-сайт плюс один або кілька смарт-контрактів. DApp має ті самі загальні властивості, що й традиційна програма[42]. Отже, головна відмінність полягає в тому, що дані та обчислення надаються блокчейном.

Переваги використання блокчейнів для DApps такі:

1. Користувач може побачити, що станеться, перш ніж виконувати функцію або надсилати будь-які дані.

2. Після того, як користувач здійснив взаємодію, її не можна скасувати, підробити або видалити. Самі по собі ці властивості корисні. Це втілює децентралізацію на рівні протоколу. Однак це сприяє іншому типу децентралізації, яка є рушійною філософською мотивацією DApps.

3. Управління може бути децентралізоване, щоб користувачі програми брали безпосередню участь в управлінні нею.

На цьому етапі ми розглянемо два приклади: один використовує перші дві властивості, а інший допомагає продемонструвати ідею управління або структурної децентралізації.

CryptoKitties – одна з найвідоміших децентралізованих програм. Це гра, створена Axiom Zen[43], яка дозволяє гравцям торгувати, розводити, колекціонувати та продавати віртуальних котів. Унікальні або дефіцитні предмети, які можна продавати, є шаблоном, який добре використовується в іграх, як традиційних, так і цифрових; однак у CryptoKitties віртуальні елементи записуються в блокчейн. Таким чином, дії є прозорими та гарантованими.

Іншим прикладом DApp є DAO децентралізована автономна організація. Додаток регулювався торговими токенами, які мали право голосу. У цьому

сенсі механізм програми гарантувався децентралізованим рівнем Ethereum, але сама концепція була розроблена для децентралізації повноважень.

У просторі DApp існує низка програм. Поки що вони в основному підпадають під такі загальні області:

- Збір коштів (ICO)
- Торгові майданчики, включаючи біржі
- Identify providers — дізнайтеся свого клієнта (KYC) і боротьба з відмиванням грошей
- Фінансові послуги
- Сек'юритизація активів
- Управління ланцюгом поставок
- Ігри.

Зараз розглянемо деякі з цих програм.

Щоб надавати фінансові послуги чи послуги з цінними паперами, як-от банк чи фондова біржа, у світі DApp існує значний бар’єр входу. Тягар дотримання нормативних вимог, персоналу, інфраструктури та інституційних відносин, необхідних для роботи, приголомшливий.

У Сполучених Штатах, наприклад, капітал для відкриття банку становить від 12 до 20 мільйонів доларів США.

Використовуючи систему на основі смарт-контрактів, де депозити регулюються загальнодоступним комп’ютерним кодом, кожен, хто вміє писати програмне забезпечення, може створити систему, яка безпечно обробляє великі обсяги активів.

Амбітний розробник програмного забезпечення у 2016 році створив діючу біржу криптовалют під назвою EtherDelta. Смарт-контракт біржі містив понад мільярд доларів ETH і токенів на піку.

Однією з часто згадуваних цілей децентралізованих проектів є «самоволодіння» даними. Після такої кількості порушень безпеки та викриття великих компаній, які продають особисту інформацію користувачів, DApps надають користувачам можливість мати більше контролю над своїми даними.

Добре розроблена система розумних контрактів може дозволити регуляторам або законодавцям дозволяти або контролювати певну діяльність на платформі[44].

Уявіть програму продажів, де продавець та споживачі можуть зареєструватися на ринку. Кожен учасник матиме унікальний ідентифікатор у програмі, і транзакції можна буде публікувати та розраховувати безпосередньо через програму. Хоча це може здатися небажаним для деяких менш скрупульозних членів аудиторії, держава може автоматично оподатковувати кожну транзакцію до відповідного рівня. Коли податкові ставки зміняться, уряд може просто відкоригувати ставку в смарт-контракті безпосередньо, і торговцям не доведеться виконувати жодних обтяжливих дій для впровадження нових ставок. Перенесення контролю до спільної системи замість розрізнених бюрократичних структур має великий потенціал для узгодження суспільних і комерційних інтересів.

DApps все ще є технологією, що розвивається. Для широкого використання необхідно подолати різні проблеми, включаючи узгодженість з нормативними актами, надійність даних і здатність швидко масштабуватись.

Інтерфейси користувача в сучасних DApps все ще доступні централізовано[45]. З одного боку, у філософії децентралізації це нормально. Критичні частини децентралізовані в блокчейні. В ідеалі вся програма, включаючи зображення та візуальні компоненти інтерфейсу користувача, також є децентралізованою. Блокчейн можна використовувати для захисту та розповсюдження цих файлів, а не лише програм і токенів. Міжпланетна файлова система є одним із прикладів розподіленого протоколу зберігання даних, який використовує для цього блокчейн. У майбутньому ми можемо побачити, що DApps стануть повністю децентралізованими за допомогою протоколів, таких як IPFS, для зберігання та обслуговування своїх файлів.

Тим часом відомі нам платформи продовжуватимуть розвиватися. Пропозиції щодо змін до Ethereum може подати кожен, і кожна нова

пропозиція нумерується відповідно до порядку подання. EIP-20, 20-та пропозиція, став стандартом ERC-20, який сприяв буму ICO.

Завдяки ERC-20 розробники токенів, біржі токенів і програмне забезпечення для гаманців Ethereum можуть реалізувати загальний інтерфейс, і всі ці різні сторони зможуть працювати разом, створюючи сумісні продукти. Це подібно до того, як Sony і Philips випустили стандарт CD Audio, щоб виробники компакт-дисків і програвачі компакт-дисків, а також продюсери музики, музичні магазини та споживачі могли використовувати один і той же тип дисків.

Рання стаття Віталія в журналі Bitcoin Magazine досліджувала «Якщо біткойн хоче досягти успіху…». Століттями ми використовували гроші майже так само, як їх було винайдено — обмінюючи шматки металу на товари та послуги. Загальнодоступному блокчейну всього кілька років, а обізнаність про блокчейн продовжує зростати. Уже відомо низку злетів і падінь, але справжній основний успіх буде досягнутий у довгостроковій перспективі.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Програмна реалізація

Створення криптовалютного гаманця, починається зі створення нового гаманця. Для реалізації даної можливості я використав такий підхід:

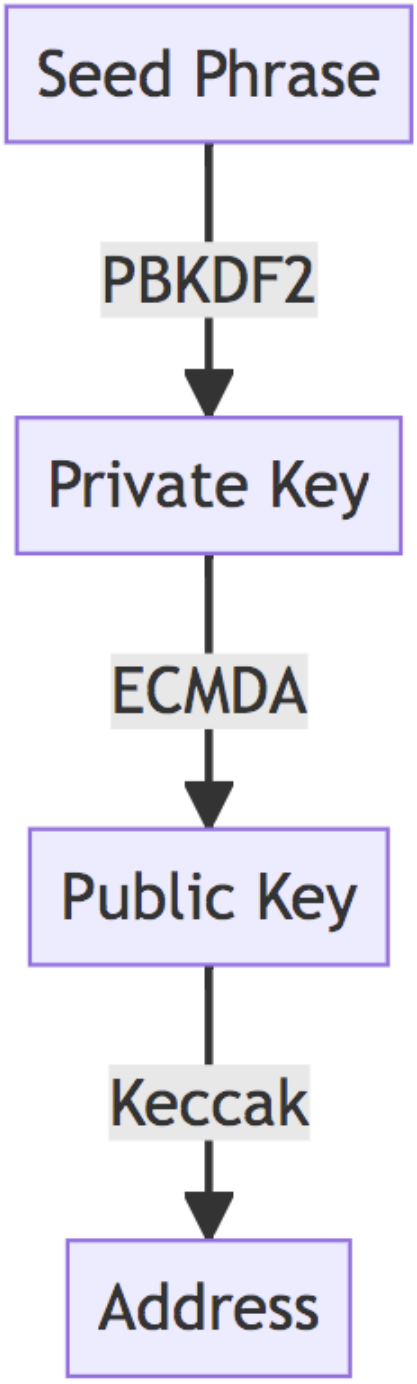


Рисунок 3.1 – Логіка роботи крипто-гаманця

Загалом, це звичайна, стабільна та найбільш поширена модель роботи крипто-гаманця, яка підтримується більшістю існуючих блокчейнів.

Як я це реалізовував? Для початку створив декілька необхідних для роботи моделей. Перша з яких це модель акаунту:

```
export interface Account {
    privateKey: string,
    address: string,
    balance: string,
}
```

Рисунок 3.2 – Код інтерфейсу Account

Цей код представляє інтерфейс з назвою Account. Інтерфейс в TypeScript використовується для визначення структури об'єктів або класів. У даному випадку, інтерфейс Account описує об'єкт, який повинен мати три властивості:

- privateKey: Це string, який представляє закритий ключ облікового запису. Зазвичай, це секретна інформація, яка використовується для підпису та розшифрування даних.
- address: Це також string, який відображає адресу облікового запису. У криптовалютних системах, наприклад, це може бути адреса гаманця, яка використовується для отримання коштів.
- balance: Це string, який представляє баланс облікового запису.

Цей інтерфейс в подальшому використовується для забезпечення того, що об'єкти, які використовуються як облікові записи в моїй програмі, відповідають конкретній загальній структурі.

Наступна модель відповідає за тип `Chain`, а також два об'єкти `goerli` та `mainnet`, які йому відповідають.

```
export type Chain = {
    chainId: string;
    name: string;
    blockExplorerUrl: string;
    rpcUrl: string;
  };

export const goerli: Chain = {
    chainId: '5',
    name: 'Goerli',
    blockExplorerUrl: 'https://goerli.etherscan.io',
    rpcUrl: 'https://goerli.infura.io/v3/59b59e23bb7c44d799b5db4a1b83e4ee',
};

export const mainnet: Chain = {
    chainId: '1',
    name: 'Ethereum',
    blockExplorerUrl: 'https://etherscan.io',
    rpcUrl: 'https://mainnet.infura.io/v3/59b59e23bb7c44d799b5db4a1b83e4ee',
};

export const CHAINS_CONFIG = {
    [goerli.chainId]: goerli,
    [mainnet.chainId]: mainnet,
};
```

Рисунок 3.3 – Код типу Chain

Основні деталі:

1. Тип Chain

- chainId: Рядок, що представляє ідентифікатор ланцюга.
- name: Рядок, що представляє назву ланцюга.
- blockExplorerUrl: Рядок, що містить URL блокексплорера для ланцюга.
- rpcUrl: Рядок, що містить URL вузла RPC для ланцюга.

2. Об’єкти goerli та mainnet

- Ці об'єкти є налаштуванням для двох різних ланцюгів: Goerli та Mainnet.
- Для кожного ланцюга вказані chainId, name, blockExplorerUrl та rpcUrl.

3. Об’єкт CHAINS_CONFIG

- Цей об'єкт містить конфігурацію для різних ланцюгів, де ключем є chainId, а значенням є об'єкт типу Chain.

Остання модель є інтерфейсом транзакції, що включає в себе:

- hash (string): унікальний ідентифікатор транзакції.
- from_address (string): адреса, з якої відправлено транзакцію.
- to_address (string): адреса, на яку відправлено транзакцію.
- value (string): значення або сума, яка була відправлена в транзакції.
- block_timestamp (string): мітка часу, яка вказує на час включення транзакції до блоку.

```
export interface Transaction {
    hash: string,
    from_address: string,
    to_address: string,
    value: string,
    block_timestamp: string,
};
```

Рисунок 3.4 – Код інтерфейсу Transaction

Наступне про що треба розповісти це про саму логіку роботи програми, основні ідеї якої прописані у трьох файлах, які об’єднані однією назвою scenes.

Перший файл описує логіку створення та відновлення акаунтів за допомогою гаманця. Ось частина роботи його логіки:

```
function AccountCreate() {
  const [seedphrase, setSeedphrase] = useState('');
  const [account, setAccount] = useState<Account | null>(null);
  const [showRecoverInput, setShowRecoverInput] = useState(false);

  function handleChange(event: React.ChangeEvent<HTMLInputElement>) {
    setSeedphrase(event.target.value);
  }

  const handleKeyDown = async (event: React.KeyboardEvent<HTMLInputElement>) => {
    if (event.keyCode === 13) {
      event.preventDefault();
      recoverAccount(seedphrase);
    }
  }

  const recoverAccount = useCallback(
    async (recoveryPhrase: string) => {

      const result = await generateAccount(recoveryPhrase);

      setAccount(result.account);

      if (localStorage.getItem(recoveryPhraseKeyName) !== recoveryPhrase) {
        localStorage.setItem(recoveryPhraseKeyName, recoveryPhrase);
      }

    }, []
  );

  useEffect(() => {

    const localStorageRecoveryPhrase = localStorage.getItem(recoveryPhraseKeyName)
    if (localStorageRecoveryPhrase) {
      setSeedphrase(localStorageRecoveryPhrase);
      recoverAccount(localStorageRecoveryPhrase);
    }
```

Рисунок 3.5 – Частина коду для створення та відновлення акаунта

Трохи більш детально розглянемо, як це все взагалі працює:

- seedphrase – стан, що зберігає введений seedphrase або приватний ключ.
- account – стан, що зберігає об'єкт облікового запису (або null, якщо обліковий запис не створено або не відновлено).
- showRecoverInput – стан, що вказує, чи відображати поле для введення ключа відновлення, чи ні.

Окремо необхідно розглянути функції та для чого вони використовуються:

- handleChange: викликається при зміні значення в інпуті. Оновлює стан seedphrase.
- handleKeyDown: викликається при натисканні клавіші в інпуті. Якщо натискана клавіша - "Enter", то викликає функцію recoverAccount для відновлення облікового запису за введеним seedphrase.
- recoverAccount: відновлює обліковий запис за введеним seedphrase, встановлює відновлений обліковий запис в стан account і зберігає seedphrase у localStorage.
- createAccount: створює новий обліковий запис і встановлює його в стан account.

Наступний файл описує логіку відображення даних конкретного акаунту. Розглянемо декілька його частин та як вони працюють.

```
interface AccountDetailProps {
  account: Account
}

const AccountDetail: React.FC<AccountDetailProps> = ({account}) => {
  const [destinationAddress, setDestinationAddress] = useState('');
  const [amount, setAmount] = useState(0);
  const [balance, setBalance] = useState(account.balance)

  const [networkResponse, setNetworkResponse] = useState<{ status: null | 'pending' | 'complete' | 'error', message: string | React.
  ReactElement }>({
    status: null,
    message: '',
  });

  useEffect(() => {
    const fetchData = async () => {
        const provider = new ethers.providers.JsonRpcProvider(goerli.rpcUrl);
        let accountBalance = await provider.getBalance(account.address);
        setBalance((String(toFixedIfNecessary(ethers.utils.formatEther(accountBalance)))));
    }
    fetchData();
}, [account.address])
```

Рисунок 3.6 – Код для відображення деталей конкретного облікового запису

Як саме це працює:

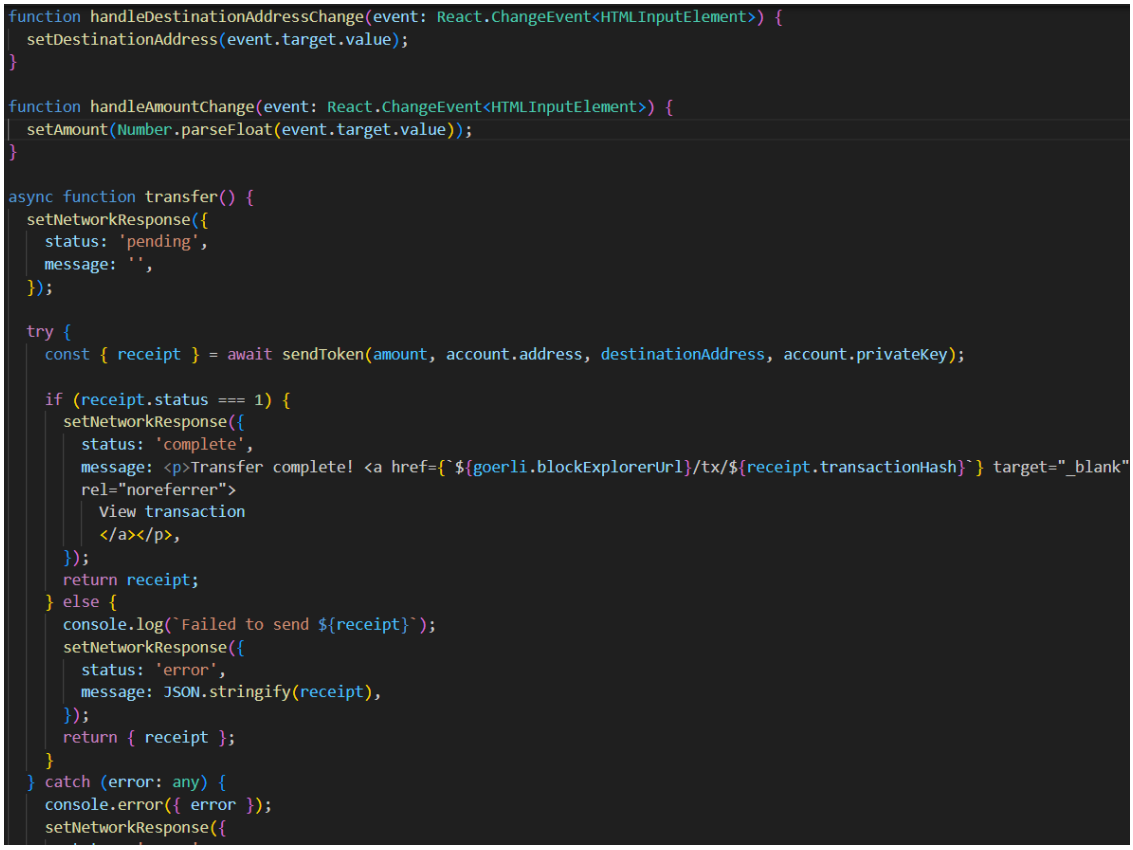
1. interface AccountDetailProps: інтерфейс для властивостей компонента AccountDetail. В даному випадку, компонент очікує отримати властивість account типу Account.

2. const AccountDetail: React.FC<AccountDetailProps> = ({account}) => { ... }: функціональний компонент AccountDetail. Його властивості передаються через деструктуризацію з об'єкта { account }. Компонент має властивості стану, які будуть оновлюватися в процесі взаємодії користувача або при отриманні даних з мережі.

Які є стани компонентів представлені в цьому коді та для чого вони використовуються:

- destinationAddress: зберігає введену адресу отримувача.
- amount: зберігає введену суму для транзакції.
- balance: зберігає баланс облікового запису з властивості account.balance.
- networkResponse: зберігає інформацію про статус мережевого запиту.

Окремо слід зазначити useEffect, який використовується для виклику функції fetchData при зміні account.address. Це призначено для оновлення балансу облікового запису при зміні адреси облікового запису. У функції fetchData створюється новий екземпляр JsonRpcProvider, а потім викликається метод getBalance для отримання актуального балансу. Отриманий баланс конвертується та оновлюється в стані balance.



```
function handleDestinationAddressChange(event: React.ChangeEvent<HTMLInputElement>) {
  setDestinationAddress(event.target.value);
}

function handleAmountChange(event: React.ChangeEvent<HTMLInputElement>) {
  setAmount(Number.parseFloat(event.target.value));
}

async function transfer() {
  setNetworkResponse({
    status: 'pending',
    message: '',
  });

  try {
    const { receipt } = await sendToken(amount, account.address, destinationAddress, account.privateKey);

    if (receipt.status === 1) {
      setNetworkResponse({
        status: 'complete',
        message: <p>Transfer complete! <a href={`${goerli.blockExplorerUrl}/tx/${receipt.transactionHash}`} target="_blank"
        rel="noreferrer">
          View transaction
          </a></p>,
      });
      return receipt;
    } else {
      console.log(`Failed to send ${receipt}`);
      setNetworkResponse({
        status: 'error',
        message: JSON.stringify(receipt),
      });
      return { receipt };
    }
  } catch (error: any) {
    console.error({ error });
    setNetworkResponse({
```

Рисунок 3.7 – Код функцій

Окремо пояснемо логіку трьох наявних функцій, які знаходяться в цьому ж файлі. А також, більш детально розглянемо, для чого вони використовуються:

1. handleDestinationAddressChange function:

- викликається при зміні вмісту в інпуті, призначеному для введення адреси отримувача.
- setDestinationAddress призначений для оновлення стану destinationAddress зі значенням, введеним користувачем в інпут.

2. handleAmountChange function:

- викликається при зміні вмісту в інпуті, призначеному для введення суми транзакції.
- setAmount призначений для оновлення стану amount з числовим значенням, введеним користувачем в інпут.

3. transfer function:

- викликається при натисканні на кнопку для виконання транзакції.
- оновлює стан networkResponse, встановлюючи статус "pending" та порожнє повідомлення.
- виконує транзакцію викликом функції sendToken, передаючи параметри amount, account.address, destinationAddress та account.privateKey.
- залежно від статусу транзакції, оновлює стан networkResponse з відповідним статусом:
  - Якщо транзакція успішна, повідомляє про успіх та надає посилання на блокексплорер для перегляду транзакції.
  - Якщо транзакція не вдалася, виводить повідомлення про помилку та виводить інформацію про транзакцію у консоль.
  - Якщо виникає будь-яка помилка, виводить повідомлення про помилку, включаючи інформацію про причину помилки.

Останній третій файл відповідає за логіку роботи та відображення транзакцій.

```
type AccountTransactionsProps = {
  account: Account,
};

const AccountTransactions: React.FC<AccountTransactionsProps> = ({ account }) => {
  const [transactions, setTransactions] = useState<Transaction[]>([]);

  const [networkResponse, setNetworkResponse] = useState<{ status: null | 'pending' | 'complete' | 'error', message: string | React.
  ReactElement }>({
    status: null,
    message: '',
  });
```

Рисунок 3.8 – Код компонентів транзакції

1. type AccountTransactionsProps: тип, який описує властивість для компонента AccountTransactions. Компонент очікує отримати властивість account типу Account.

2. const AccountTransactions: React.FC<AccountTransactionsProps> = ({ account }) => { ... }: функціональний компонент, властивості якого передаються через деструктуризацію з об'єкта { account }. Компонент має два стани: transactions для зберігання списку транзакцій та networkResponse для зберігання інформації про статус мережевого запиту та повідомлення.

Стани компонента AccountTransactions:

- transactions: зберігає список транзакцій для відображення.
- networkResponse: зберігає інформацію про статус мережевого запиту.

```
const getTransactions = useCallback(
  () => {
      setNetworkResponse({
          status: 'pending',
          message: '',
        });
      TransactionService.getTransactions(account.address).then(response => {
        setTransactions(response.data.result);
      }).catch(error => {
          console.log({error})
          setNetworkResponse({
              status: 'error',
              message: JSON.stringify(error),
            });
      }).finally(()=>{
          setNetworkResponse({
              status: 'complete',
              message: '',
          });
      });
    },[account.address]
) ;

useEffect(() => {
  getTransactions();
}, [getTransactions]);
```

Рисунок 3.9 – Код функції getTransactions, useEffect

Цей код використовується для виклику сервісу транзакцій та отримання транзакцій для певного облікового запису. Як це реалізовується?

1. getTransactions – функція, яка викликається для отримання транзакцій облікового запису.
2. Спочатку встановлює стан networkResponse зі статусом "pending" і порожнім повідомленням.

3. Далі TransactionService.getTransactions(account.address) для звернення до сервісу транзакцій та отримання списку транзакцій.

4. Якщо запит вдається, встановлюється список транзакцій у стан transactions.

5. Якщо виникає помилка, виводить інформацію про помилку у консоль та встановлює стан networkResponse зі статусом "error" і відповідним повідомленням.

6. Незалежно від того, чи вдалася або невдала транзакція, завжди встановлює стан networkResponse зі статусом "complete" та порожнім повідомленням у розділ finally.

useEffect, в свою чергу, використовується для автоматичного виклику функції getTransactions при монтажі компонента або при зміні функції getTransactions.

Вона викликає getTransactions при монтажі, оскільки getTransactions не змінюється під час виконання компонента.

Основна мета – отримати транзакції та оновити стан компонента при завантаженні. Розглянемо код файлу TransactionService.

```
export class TransactionService {

  static API_URL =  'https://deep-index.moralis.io/api/v2';
  static API_KEY =  'EuwYtjWWWHGbnwsCnGauMtMMaEQZugtjaws2ybm2ZpSR15a8vzl6QUPkEUWHTOCU';

  static async getTransactions(address: string) {
    const options = {
        method: 'GET',
        url: `${TransactionService.API_URL}/${address}`,
        params: {chain: goerli.name.toLowerCase()},
        headers: {accept: 'application/json', 'X-API-Key': TransactionService.API_KEY}
      };

    const response = await axios.request(options);
    return response;
  }

}
```

Рисунок 3.10 – Код класу TransactionService

Цей код визначає клас TransactionService, який є сервісом для взаємодії з API для отримання інформації про транзакції. Давайте розглянемо код быльш детально:

Статичні змінні:

- API_URL: Змінна, що містить URL API сервісу транзакцій.
- API_KEY: Змінна, що містить ключ API для доступу до сервісу.

Далы йде getTransactions метод, який приймає параметр address для отримання транзакцій.

Створює об'єкт options для конфігурації HTTP-запиту, вказуючи метод, URL API з адресою та параметром ланцюга, і заголовки з ключем API.

Викликає axios.request(options) для виконання HTTP-запиту і отримання відповіді.

3.2 Тестування роботи гаманця

Для початку роботи з гаманцем необхідно створити акаунт. Для цього просто натискаємо кнопку “Create Account”.

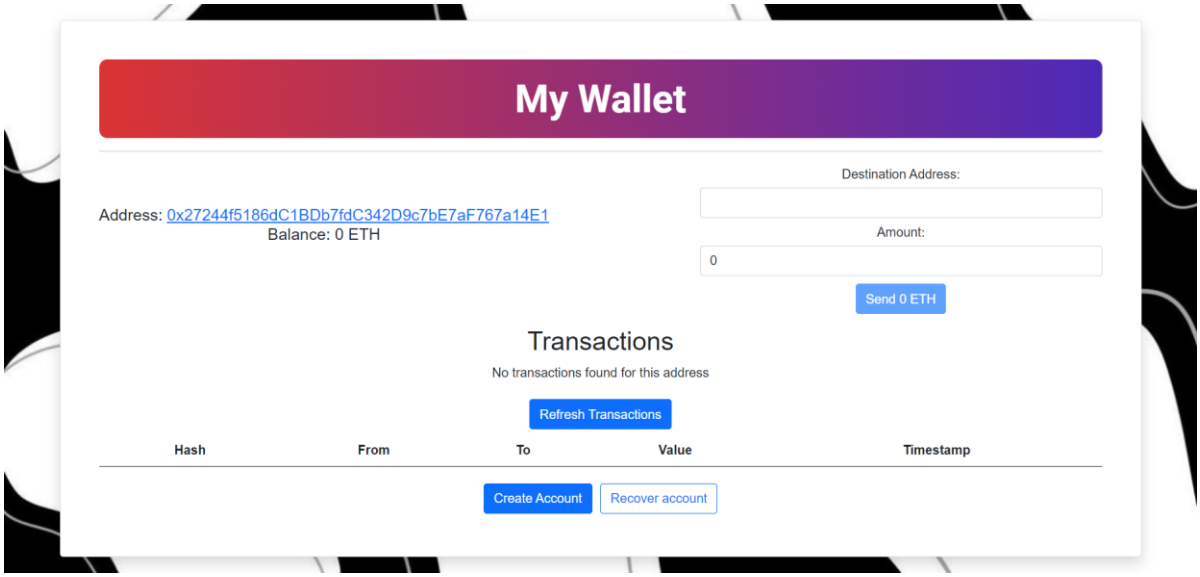


Рисунок 3.11 – Новий акаунт

Як ми бачимо створився абсолютно новий пустий акаунт. У ньому наявні такі дані як:

- Адреса гаманця, в даному випадку: 0x27244f5186dC1BDb7fdC342D9c7bE7aF767a14E1
- Баланс рівний нулю
- Пустий перелік транзакцій

Перевіримо чи дійсно створений акаунт записаний та існує в блокчейні. Для цього необхідно натиснути на адресу акаунту в лівій частині гаманця і повинно відкритись вікно, в якому будуть дані про запис акаунту.

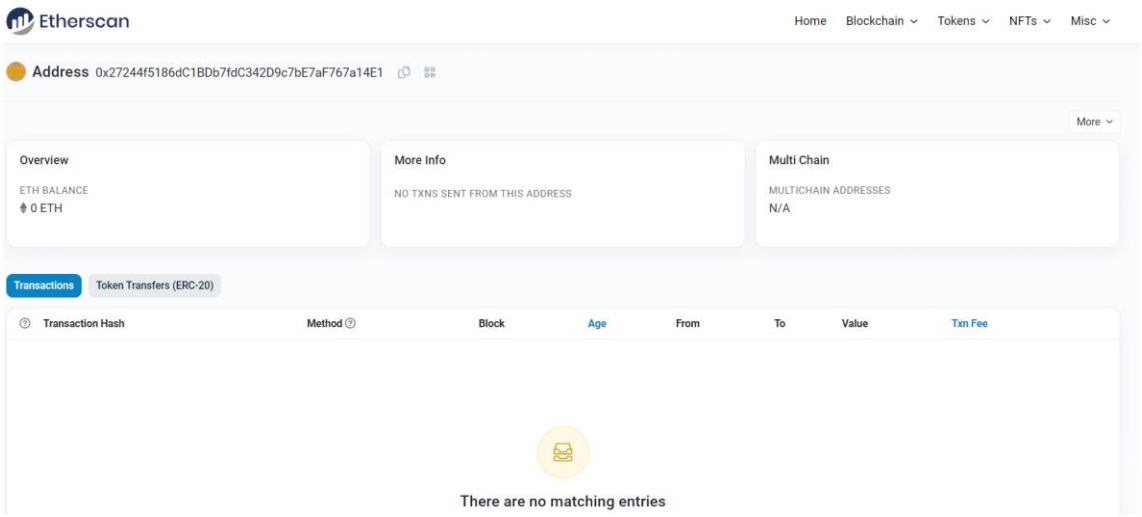


Рисунок 3.12 – Дані про запис акаунту до блокчейну

Як бачимо, створення та запис акаунту відбулись, тому функція створення гаманця працює коректно, а також сайті Etherscan дані про акаунт відповідають даним в самому гаманці.

Також ми можемо побачити поля інформації транзакцій на сайті Etherscan. Вони знадобляться нам пізніше. Вони несуть таку інформацію:

1. Transaction Hash: унікальний ідентифікатор транзакції в мережі. Це хеш-значення, яке використовується для однозначної ідентифікації конкретної транзакції. Кожна транзакція в мережі має свій власний унікальний хеш.

2. Method: поле, що вказує на тип операції чи функції, яку викликано в межах транзакції.

3. Block: номер блока, в якому включена транзакція. Блок - це група транзакцій, які об'єднуються разом і додаються до ланцюжка блокчейну.

4. Age: час, який минув від моменту включення транзакції до поточного моменту.

5. From: адреса відправника транзакції.

6. To: адреса отримувача транзакції.

7. Value: кількість криптовалюти, яка відправляється або отримується в ході транзакції.

8. Txn Fee: комісія за виконання транзакції, яку відправник платить майнерам за включення транзакції в блок. Це може включати в себе газову плату та можливо інші витрати, пов'язані з мережею.

Наступним етапом перевіримо можливість входу у вже існуючий акаунт. Для цього слід натиснути на кнопку “Recover account”, після чого відкриється поле для вводу seed фрази або приватного ключа, за допомогою яких і відкривається доступ до акаунту.

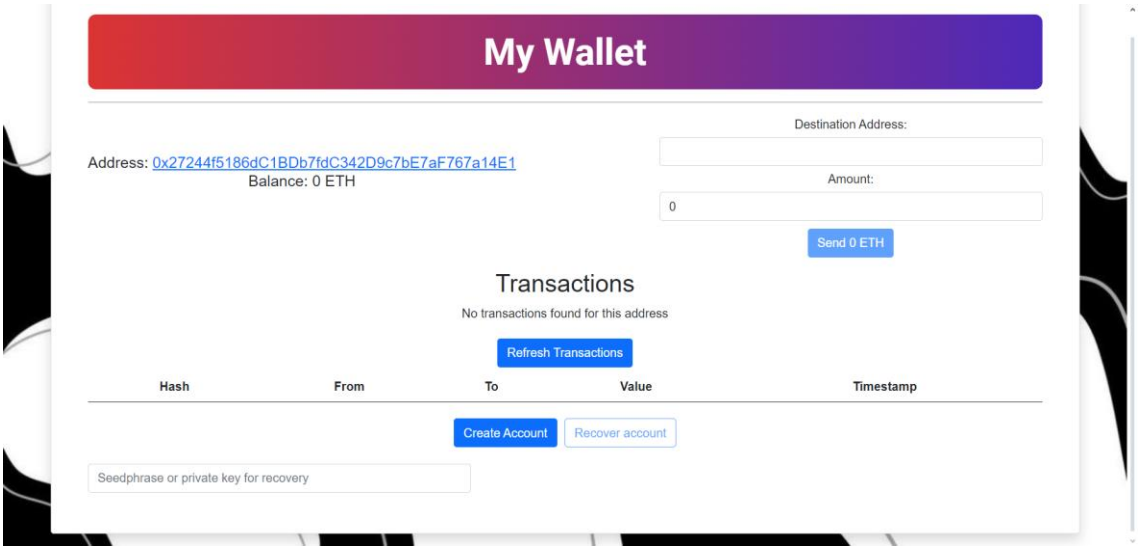


Рисунок 3.13 – Вхід до існуючого акаунту

Для демонстрації є попередньо підготовлений акаунт. Після вводу seed фрази, настикаємо “Enter” та отримаємо наступний результат:

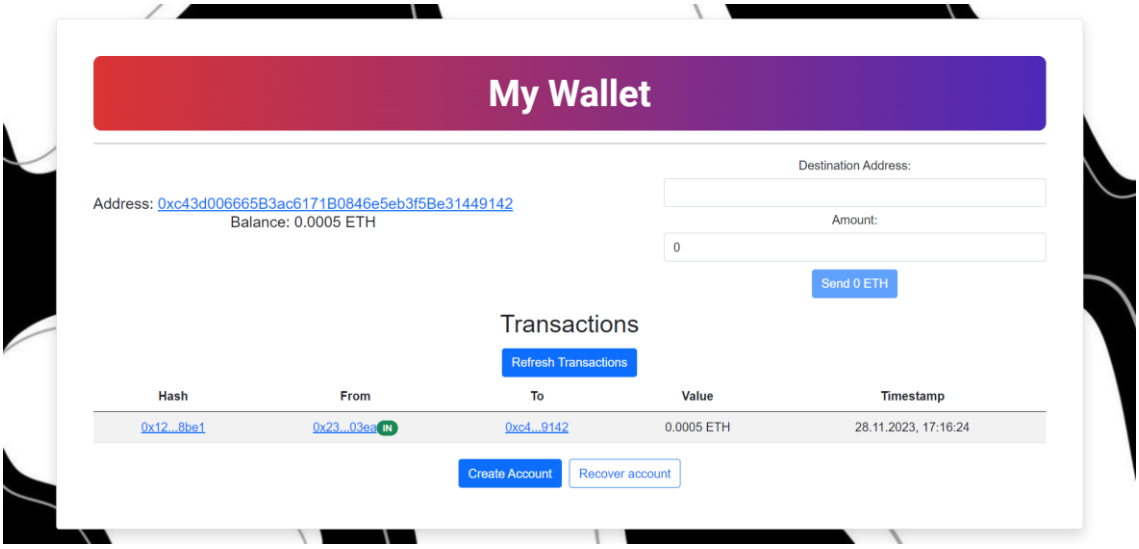


Рисунок 3.14 – Вхід до існуючого акаунту

У цілях безпеки, не показано sedd фразу. Як можна побачити вхід успішний та відображаються всі дані, щодо акаунту коректні. Для того, щоб впевнитись у вірності даних, необхідно перейти до перегляду акаунту за допомогою Etherscan.

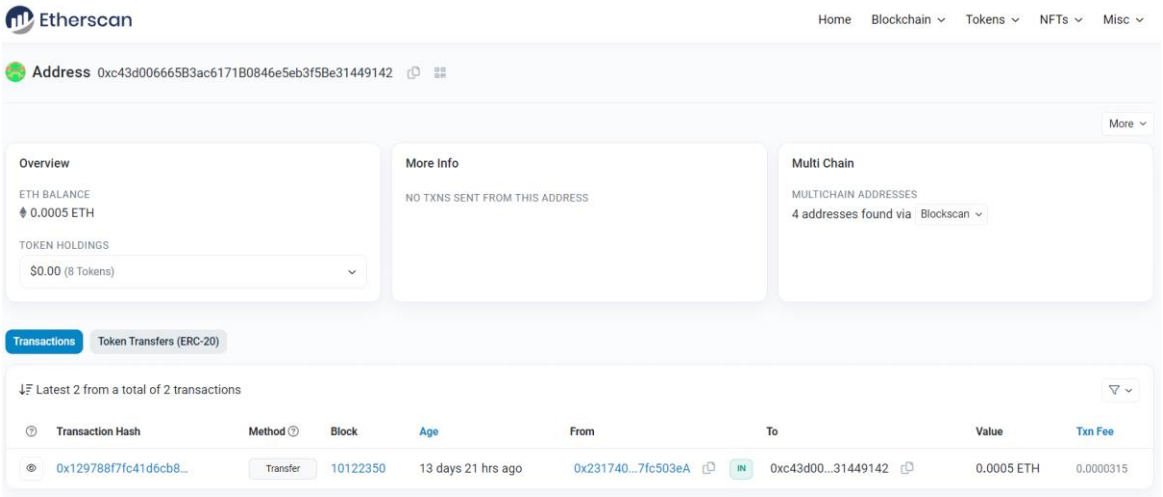


Рисунок 3.15 – Інформація, щодо існуючого акаунту

Як можна побачити інформація, щодо адреси акаунту, балансу та транзакцій співпадає.

Наступне, що необхідно перевірити це коректність відображення транзакцій. Для цього, я скористуюсь так званим краном та переведу на цей акаунт 0.001 ETH.

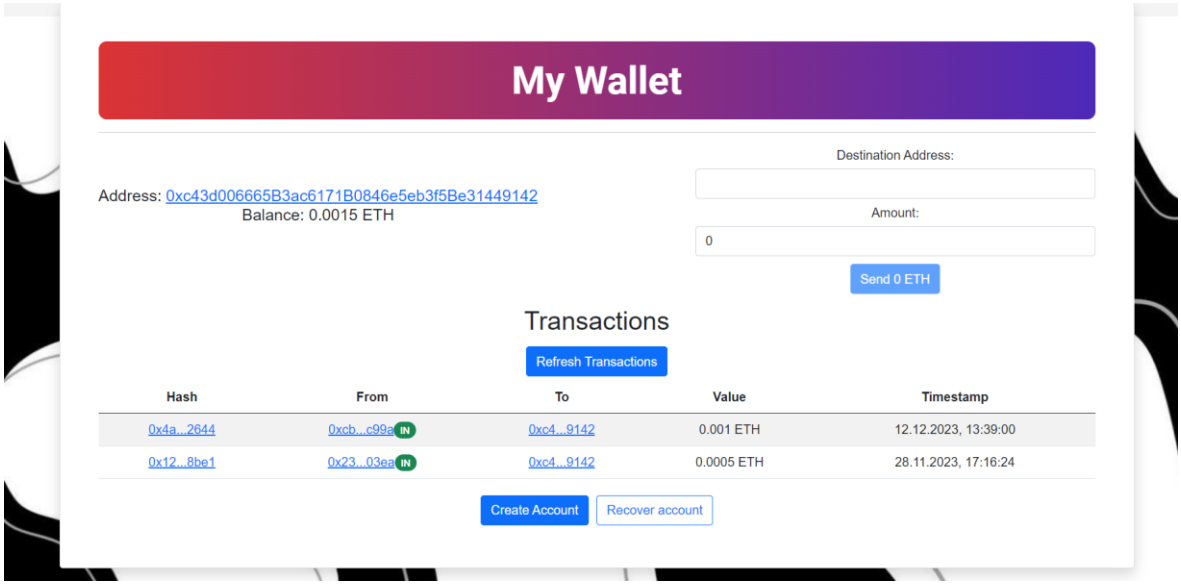


Рисунок 3.16 – Зміна балансу та транзакцій в гаманці

У гаманці змінилися дані, слід впевнитись чи відповідають вони дійсності, для цього слід знову ж таки звірити їх з даними про акаунт з сайту Etherscan.

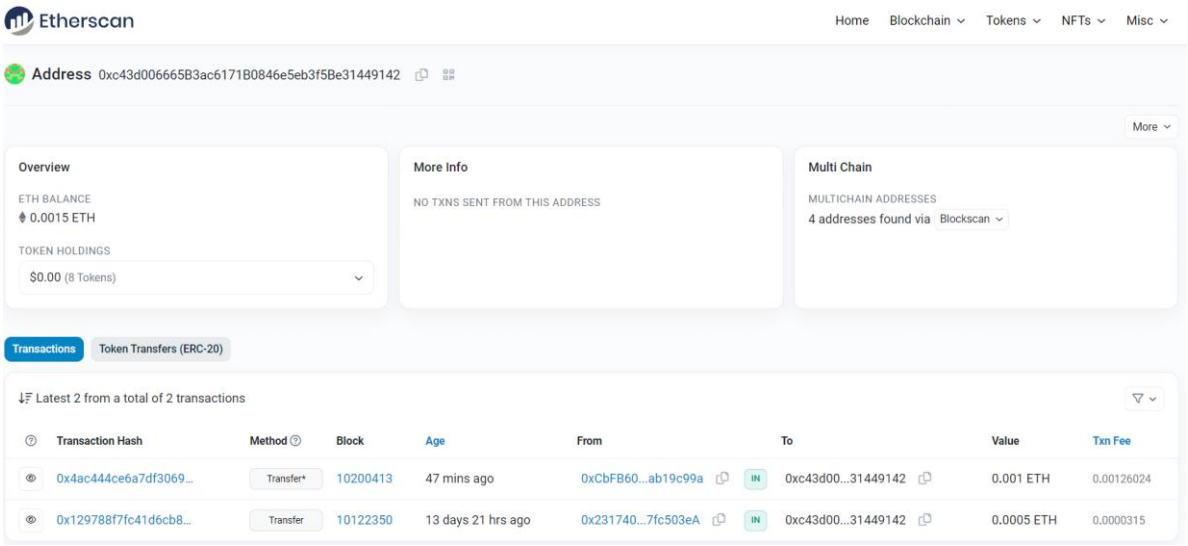


Рисунок 3.17 – Зміна даних другого акаунту

Завершальною перевіркою, буде перевірка роботи транзакцій. Для цього необхідно виконати наступні дії:

1. Увійти до акаунту з ненульовим балансом.
2. В поле “Destination Address:” ввести адресу отримувача.
3. В поле “Amount:” ввести кількість ETH, що ви хочете надіслати
4. Натиснути “Send”

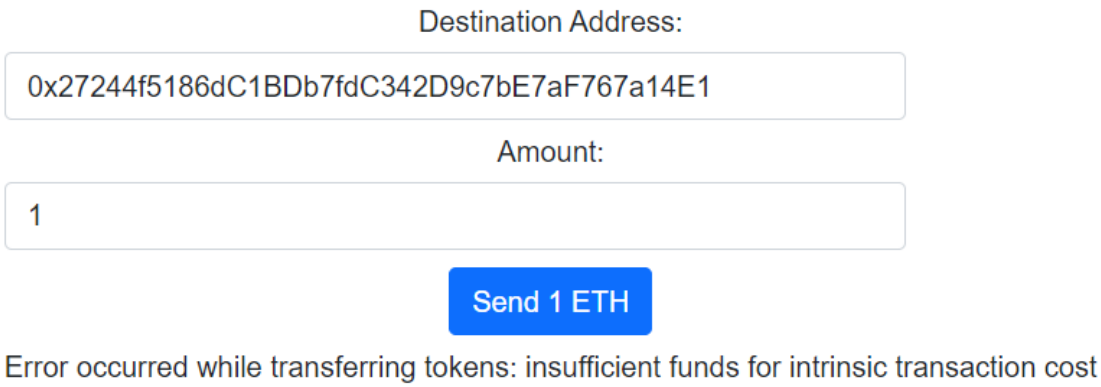


Рисунок 3.18 – Перевірка на недостатній баланс

У даному прикладі, введено існуючу адресу, але завелику кількість коштів для відправлення. Тому отримаємо в результаті помилку з наступним текстом «Error occurred while transferring tokens: insufficient funds for intrinsic transaction cost».

Завдяки цій перевірці у користувача при недостатньому балансі, транзакція не відправляється, а відміняється, за рахунок чого він не втрачає кошти на комісію.

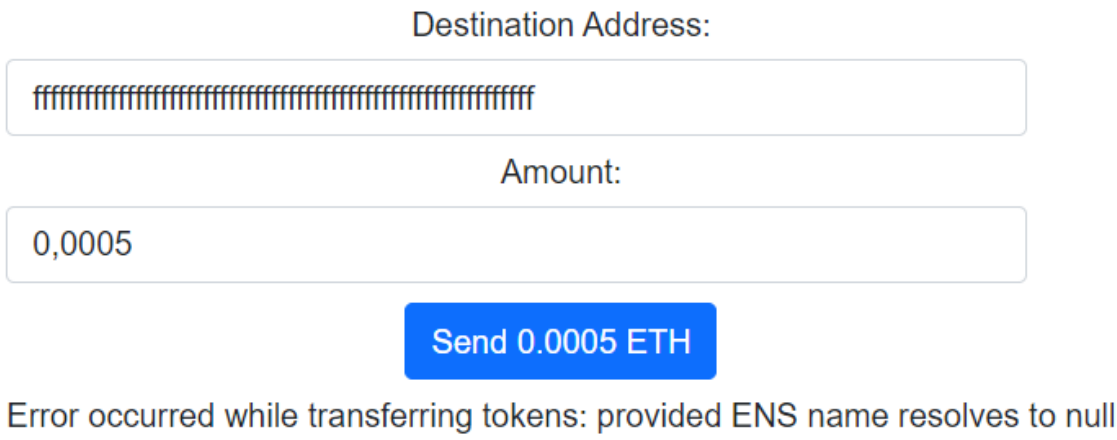


Рисунок 3.19 – Перевірка на неправильно введену адресу

У цьому ж прикладі навпаки, введено неіснуючу адресу, але достатню кількість коштів для відправлення. Тому отримаємо в результаті помилку з наступним текстом «Error occurred while transferring tokens: provided ENS name resolves to null».

Завдяки цій перевірці на правильність введення існуючої адреси гаманця, транзакція не відправляється, а відміняється, за рахунок чого користувач не втрачає кошти на комісію.

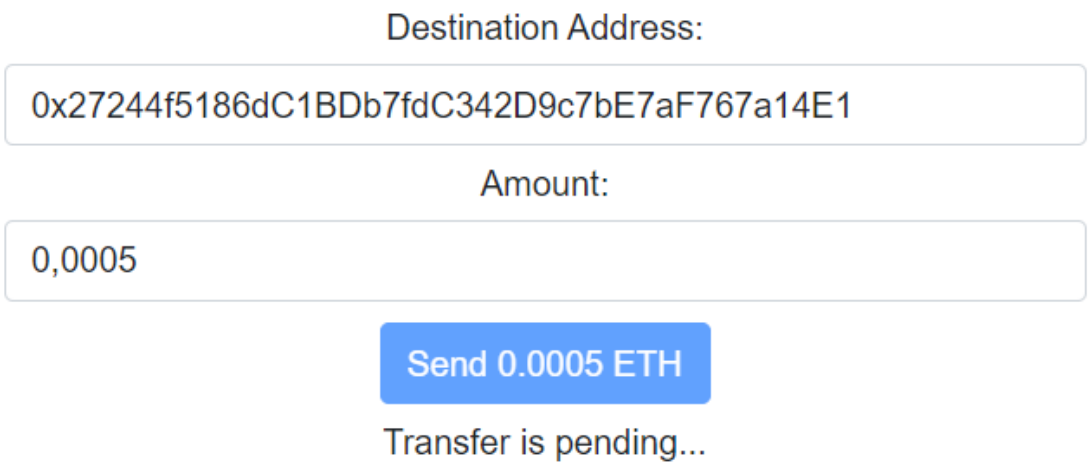


Рисунок 3.20 – Очікування транзакції

Після введення коректних даних гаманця та кількості ETH для переказу, транзакція відправляється в обробку, де очікує на підтвердження та запис до блокчейну. Під полем вводу з’являється відповідна інформація.

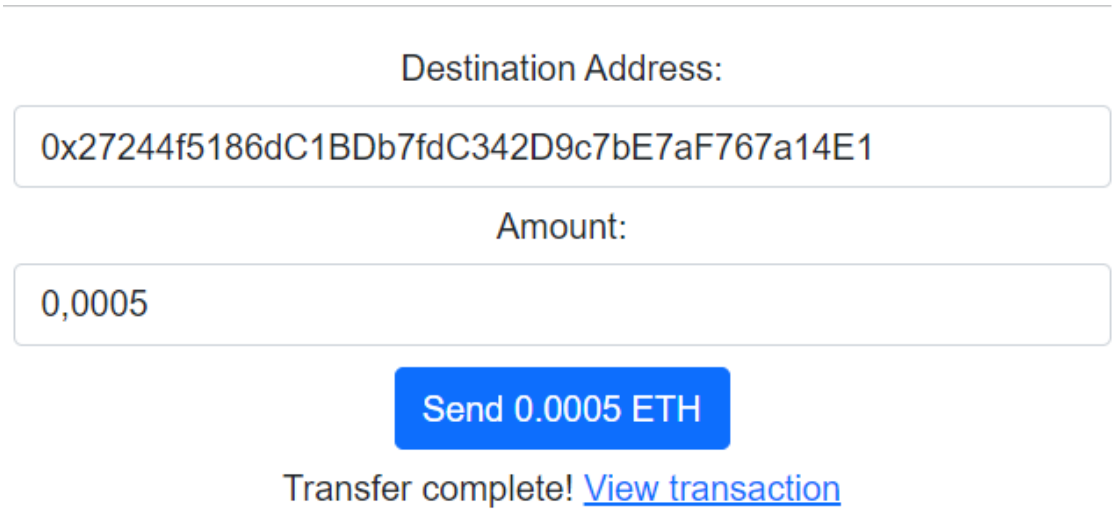


Рисунок 3.21 – Транзакція успішна

Після успішного відпрацювання транзакції з’являється відповідний напис та посилання на перегляд запису про транзакцію.

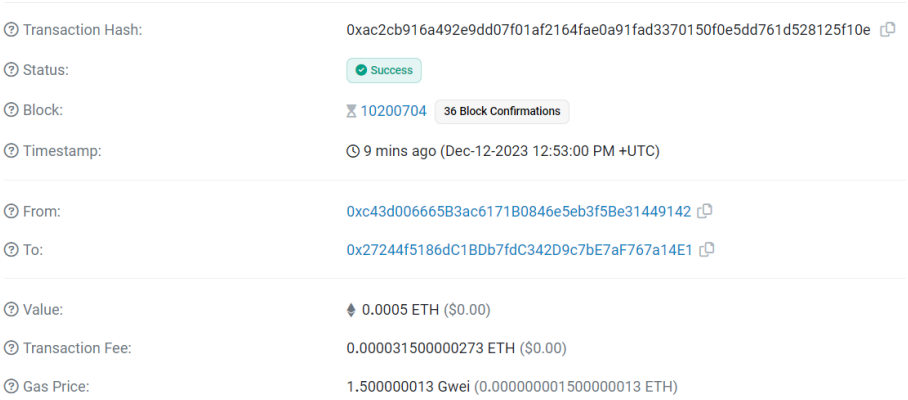


Рисунок 3.22 – Інформація про транзакцію з Etherscan

Відкривши більш детальний огляд ви можете переглянути дані транзакції, та впевнитись в їх коректності.

Слід зазначити, що для відправлення 0,0005 ETH, необхідно сплатити комісію валідатору транзакції, тому на рахунку залишиться не рівне число.

Останнє, що залишається перевірити це кнопку оновлення транзакцій. Для цього перейдемо на акаунт отримувача. Там не відображаються жодні транзакції, але після натискання на кнопку “Refresh Transactions” повинні додатись дані.

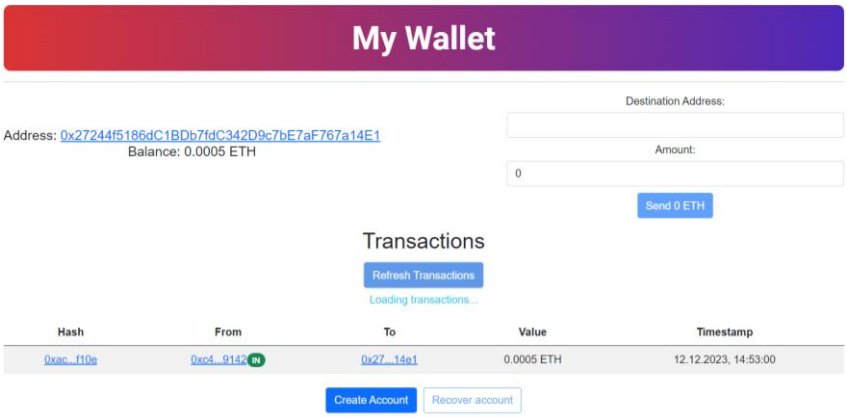


Рисунок 3.23 – Оновлення транзакцій

Як можна зрозуміти весь заявлений функціонал працює правильно. Транзакції оновлюються та відображаються. Є можливість переглянути інформацію про акаунт окремо по кожній транзакції не виходячи з гаманця.

ВИСНОВКИ

Отже, у ході виконання дипломної роботи було виявлено, що розвиток криптовалют та блокчейн-технологій має суттєвий вплив на сучасну фінансову систему. Розглянувши основні аспекти керування криптовалютним гаманцем в блокчейні Ethereum, було визначено, що використання блокчейн-технологій сприяє покращенню безпеки та ефективності операцій.

Також, досліджено основні функції гаманця, зокрема зберігання криптовалютних активів, проведення транзакцій та моніторинг стану рахунків. Розглянуто механізми шифрування та безпеки, що гарантують захист від несанкціонованого доступу до гаманця.

Підсумовуючи отримані результати, можна зазначити, що інформаційна технологія керування криптовалютним гаманцем в блокчейні Ethereum є необхідною складовою для подальшого розвитку цифрових фінансів. Розроблені рекомендації та висновки можуть служити основою для подальших досліджень та впровадження інновацій у сфері блокчейн-технологій та криптовалют.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introducing Binance Web3 Wallet. [Електронний ресурс] – Режим доступу до ресурсу: https://www.binance.com/en/blog/markets/introducing-binance-web3-wallet-5931309459106555347.

2. Microsoft планує інтегрувати криптогаманець у консоль Xbox. [Електронний ресурс] – Режим доступу до ресурсу: https://psm7.com/uk/cryptocurrency/microsoft-planiruet-integrirovat-kriptokoshelek-v-konsol-xbox.html.

3. Міністерство цифрової трансформації України. Офіційні криптогаманці, куди можна переказати гроші для армії України. [Електронний ресурс] – Режим доступу до ресурсу: https://thedigital.gov.ua/news/ofitsiyni-kriptogamantsi-kudi-mozhna-perekazati-groshi-dlya-armii-ukraini.

4. Andreas M. Antonopoulos, David A. Harding., Mastering Bitcoin. 2023.

5. Ellen Gunnarsd´ottir., Trezor One side-channel analysis setup. June 2020.

6. Lee, WM. Using the MetaMask Chrome Extension. In: Beginning Ethereum Smart Contracts Programming. Apress, Berkeley, CA, 2023.

7. Saurabh Suratkar, Mahesh Shirole, Sunil Bhirud, Cryptocurrency Wallet: A Review., IEEE, 2021.

8. Crypto.com. Custodial vs Non-Custodial Wallets. [Електронний ресурс] – Режим доступу до ресурсу: https://crypto.com/university/custodial-vs-non-custodial-wallets.

9. InfoStride. Comparative analysis of Custodial and Non-Custodial wallets. [Електронний ресурс] – Режим доступу до ресурсу: https://infostride.com/custodial-and-non-custodial-wallets/.

10. Kraken. What Are Custodial and Non-Custodial Crypto Wallets? [Електронний ресурс] – Режим доступу до ресурсу: https://www.kraken.com/uk-ua/learn/custodial-non-custodial-crypto-wallet.

11. OKX. Paper wallets: What are they and how do they work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.okx.com/ua/learn/paper-wallets.

12. Elsevier B.V., Future Generation Computer Systems, vol. 91, 2019, с. 136-143.

13. A.A. Monrat, O. Schelén, K. Andersson., A survey of blockchain from the perspectives of applications , challenges , and opportunities, IEEE Access, 2019.

14. S. Zollner, K.-K.R. Choo, N.-A. Le-Khac., An automated live forensic and postmortem analysis tool for bitcoin on Windows systems, IEEE Access, 2019.

15. T. Haigh, F. Breitinger, I. Baggili., If I had a million cryptos: cryptowallet application analysis and a trojan proof-of-concept, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 259, 2019, с. 45-65.

16. Taruna, Rishabh., Analysis of Security Issues in Blockchain Wallet, 2021.

17. Daojing He; Shihao Li; Cong Li; Sencun Zhu; Sammy Chan; Weidong Min; Nadra Guizani., Security Analysis of Cryptocurrency Wallets in Android-Based Applications, IEEE, 2020, с. 114-119.

18. T. Bui та інші., "Pitfalls of Open Architecture: How Friends Can Exploit Your Cryptocurrency Wallet", Proc. 12th European Workshop on Systems Security. с. 25–28, 2019.

19. Kostis Karantias., SoK: A Taxonomy of Cryptocurrency Wallets: Cryptology ePrint Archive, 2020.

20. Thomas Webbers., Bringing the security of Bitcoin full nodes to low resource devices, 2020.

21. Rezaeighaleh, Hossein, Improving Security of Crypto Wallets in Blockchain Technologies, Electronic Theses and Dissertations, 2020- с 403.

22. Chen W, Zheng Z, Cui J, Ngai E, Zheng P and Zhou Y., Detecting Ponzi schemes on Ethereum: Towards healthier blockchain technology Proceedings of the World Wide Web Conference, 2018, с. 1409-1418.

23. Volety T, Saini S, McGhin T, Liu C Z and Choo K K R., Cracking Bitcoin wallets: I want what you have in the wallets Future Generation Computer Systems, 2019, c. 136-143.

24. Zheng Z, Xie S, Dai H N, Chen X and Wang H., Blockchain challenges and opportunities: A survey International Journal of Web and Grid Services, 2018, c. 352-375.

25. Elrom E., Wallets and Smart Contracts in the Blockchain Developer, Berkeley: Apress, 2019, c 213-256.

26. Lee W.M., Using the metamask chrome extension Beginning Ethereum Smart Contracts, Programming, Berkeley, Apress, 2019, c. 93-126.

27. Rezaeighaleh H., Multilayered defense-in-depth architecture for cryptocurrency wallet, - IEEE, 2020.

28. Dolny T., Dugas S. .,How to Find Stolen Cryptocurrency: Litigation Tools Used by Insolvency Professionals in Dooga, - Banking & Finance Law Review, 2022.

29. Blockchain Council. What Are Blockchain Nodes? Detailed Guide [UPDATED], [Електронний ресурс] – Режим доступу до ресурсу: https://www.blockchain-council.org/blockchain/blockchain-nodes/.

30. What is a Blockchain Full Node?, [Електронний ресурс] – Режим доступу до ресурсу: https://originstamp.com/blog/what-is-a-blockchain-full-node/.

31. Wallstreetmojo Team, Light Node, [Електронний ресурс] – Режим доступу до ресурсу: https://www.wallstreetmojo.com/light-node/.

32. Definition of mining node – PCMag, [Електронний ресурс] – Режим доступу до ресурсу: https://www.pcmag.com/encyclopedia/term/mining-node.

33. Bitcoin Magazine. About Bitcoin Magazine, [Електронний ресурс] – Режим доступу до ресурсу: https://bitcoinmagazine.com/about.

34. Murphy, Hannah, The rise and fall of Ethereum., Financial Times. London: The Financial Times Ltd., 2018.

35. Di Angelo, Monika; Salzer, Gernot, Identification of token contracts on Ethereum: standard compliance and beyon, International Journal of Data Science and Analytics., 2023, c: 333–352.

36. Bashir, Imran, Mastering Blockchain – Third Edition : a deep dive into distributed ledgers, consensus protocols,... smart contracts, dapps, cryptocurrencies, ethereum.: Packt Publishing Limited., 2020, c. 312.

37. Wood, Gavin, Ethereum: A Secure Decentralized Generalised Transaction Ledger (EIP-150), 2018.

38. IBM. What is blockchain technology? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/blockchain.

39. Fortune. There are two very real reasons Ethereum is taking off. [Електронний ресурс] – Режим доступу до ресурсу: https://fortune.com/2021/05/06/ethereum-price-ether-predictions-why-it-is-rising-usd-bitcoin/.

40. Clark, Aaron, Ethereum's Energy Revamp Is No Guarantee of Global Climate Gains: Bloomberg.com, 2022.

41. Ethereum.org, ETHEREUM VIRTUAL MACHINE (EVM), [Електронний ресурс] – Режим доступу до ресурсу: https://ethereum.org/en/developers/docs/evm/.

42. Kaidong Wu, Yun Ma, Gang Huang, Xuanzhe Liu, A first look at blockchain-based decentralized applications: Special Issue: Blockchain and Decentralized Applications Volume51, Issue10, 2021.

43. Jiang X-J, Liu XF, CryptoKitties Transaction Network Analysis: The Rise and Fall of the First Blockchain Game Mania: Front, 2021.

44. Introduction to Smart Contracts, [Електронний ресурс] – Режим доступу до ресурсу: https://docs.soliditylang.org/en/v0.4.24/introduction-to-smart-contracts.html.

45. Froehlich M., Moser B., Alt F., Schmidt A., Supporting Interface Experimentation for Blockchain Applications: Adjunct Proceedings of the 2022, 2022.

ДОДАТОК А

App.tsx

```
import React from 'react';
import './App.css';
import AccountCreate from './scenes/Account/AccountCreate';

function App() {

  return (
    <div className="App container">
      <div className="background"></div>
      <AccountCreate />



    </div>

  );
}

export default App;
```

index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
```

```
);
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
reportWebVitals();
```

AccountUtils.ts

```
import { Wallet } from 'ethers';
import { Account } from '../models/Account';

export function generateAccount(seedPhrase: string = "", index: number = 0):
{ account: Account, seedPhrase: string } {
  let wallet: Wallet;

  if (seedPhrase === "") {
    seedPhrase = Wallet.createRandom().mnemonic.phrase;
  }

  wallet = (seedPhrase.includes(" ")) ? Wallet.fromMnemonic(seedPhrase,
m/44'/60'/0'/0/${index}) :
  new Wallet(seedPhrase);

  const { address } = wallet;
  const account = { address, privateKey: wallet.privateKey, balance: "0" };

  return { account, seedPhrase: seedPhrase.includes(" ")? seedPhrase : "" };
}
```

```
export function shortenAddress(str: string, numChars: number=4) {
  return ${str.substring(0, numChars)}...${str.substring(str.length - numChars)};
}

export function toFixedIfNecessary( value: string, decimalPlaces: number = 2 ){
  return +parseFloat(value).toFixed( decimalPlaces );
}
```

TransactionUtils.ts

```
import { ethers, Wallet } from 'ethers';
import { CHAINS_CONFIG, goerli } from '../models/Chain';

export async function sendToken(
  amount: number,
  from: string,
  to: string,
  privateKey: string,
) {

  const chain = CHAINS_CONFIG[goerli.chainId];

  const provider = new ethers.providers.JsonRpcProvider(chain.rpcUrl);

  const wallet: Wallet = new ethers.Wallet(privateKey, provider);

  const tx = {
    to,
    value: ethers.utils.parseEther(amount.toString()),
```

```
  };

  const transaction = await wallet.sendTransaction(tx);

  const receipt = await transaction.wait();

  return {transaction, receipt};
}
```