ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методу децентралізованого обміну файлами із використанням технології Blockchain»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

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

_____ Дмитро ШЕВЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Дмитро ШЕВЧЕНКО
група КНДМ-63

Керівник: Олег ІЛЬЇН
науковий ступінь, вчене звання д.т.н., професор

Рецензент:
науковий ступінь, вчене звання
_____
(Ім'я, ПРІЗВИЩЕ)

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Шевченку Дмитру Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка методу децентралізованого обміну файлами із використанням технології Blockchain

керівник кваліфікаційної роботи Олег ІЛЬЇН д.т.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023р. №145

2. Строк подання кваліфікаційної роботи «20» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: метеріали тез для конференції, науково-дослідної, науково-професійної та переддипломної практик

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1.АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ ДЛЯ ОБМІНУ ФАЙЛАМИ
2.АНАЛІЗ ТЕХНОЛОГІЇ BLOCKCHAIN
3.РОЗРОБКА МЕТОДУ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ФАЙЛАМИ
5. Перелік графічного матеріалу: *презентація*

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10-01.11.23	
2	Вивчення матеріалів для аналізу розвитку технологій обміну файлами	01.11-11.11.23	
3	Дослідження технологій обміну файлами	11.11-15.11.23	
4	Аналіз технології Blockchain	15.11-20.11.23	
5	Дослідження архітектури децентралізованих додатків	27.11-03.12.23	
6	Розробка методу децентралізованого обміну файлами із використанням технології Blockchain	03.12-17.12.23	
7	Оформлення роботи: вступ, висновки, реферат	7.12-15.12.23	
8	Розробка демонстраційних матеріалів	15.12-20.12.23	

Здобувач вищої освіти _____ Дмитро ШЕВЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Олег ІЛЬЇН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 82 стор. 34 рис., 12 джерел.

Мета роботи – розробка методу децентралізованого обміну файлами із використанням технології Blockchain, спрямованого на підвищення безпеки, ефективності та надійності обміну інформацією між користувачами.

Об'єкт дослідження – технології обміну файлами та їх розвиток в сучасному інформаційному середовищі.

Предмет дослідження – метод децентралізованого обміну файлами з використанням технології Blockchain як інноваційного інструменту для підвищення безпеки та надійності обміну файловою інформацією.

Короткий зміст роботи: Кваліфікаційна робота присвячена розробці методу децентралізованого обміну файлами, використовуючи технологію Blockchain. Робота складається з трьох основних частин:

- Аналіз розвитку технологій для обміну файлами
- Аналіз технології Blockchain
- Розробка методу децентралізованого обміну файлами

Під час виконання роботи було вивчено еволюцію технологій обміну файлами, окреслено основні тенденції та характеристики. Освітлено принципи та можливості технології Blockchain, а також вивчено її вплив на сферу обміну файлами. Визначено вимоги до створеної системи обміну файлами. Розроблено та реалізовано метод децентралізованого обміну файлами на основі технології Blockchain. Здійснено тестування функціональності, ефективності та безпеки запропонованого методу.

КЛЮЧОВІ СЛОВА: BLOCKCHAIN, ОБМІН ФАЙЛАМИ, DAPP, SOLIDITY, ДЕЦЕНТРАЛІЗАЦІЯ

ABSTRACT

Text part of the master's qualification work: 82 pages, 34 pictures, 12 sources.

The purpose of the work is to develop a method of decentralized file sharing using Blockchain technology aimed at improving the security, efficiency and reliability of information exchange between users.

Object of research - file sharing technologies and their development in the modern information environment.

The subject of the study is a method of decentralized file sharing using Blockchain technology as an innovative tool for improving the security and reliability of file information exchange.

Summary of work: The qualification work is devoted to the development of a method of decentralized file sharing using Blockchain technology. The work consists of three main parts:

- Analysis of the development of file sharing technologies
- Analysis of Blockchain technology
- Development of a decentralized file sharing method

In the course of the work, the evolution of file sharing technologies was studied, the main trends and characteristics were outlined. The principles and capabilities of Blockchain technology are highlighted, and its impact on the file sharing industry is studied. The requirements for the created file sharing system are determined. A method based on Blockchain technology is developed and implemented. The efficiency and security of the proposed method are analysed.

KEYWORDS: BLOCKCHAIN, FILE SHARING, DAPP, SOLIDITY, DECENTRALIZATION

ЗМІСТ

ВСТУП

В сучасному світі, охопленому стрімким розвитком інформаційних технологій та зростанням інтересу до децентралізації, питання забезпечення безпеки та ефективності обміну файлами стає дедалі більш актуальним. Одним із перспективних напрямків у цьому контексті є використання технології Blockchain, яка вже знайшла широке застосування у сферах фінансів, логістики та галузях, де важливість безпеки та надійності висока. Дипломна робота присвячена розробці та вивченню методу децентралізованого обміну файлами, в основі якого лежить використання блокчейн-технології. У цьому контексті вирішуються не лише технічні аспекти розробки, але й питання забезпечення конфіденційності, цілісності та доступності даних, що робить дане дослідження вкрай важливим у контексті сучасних викликів інформаційного суспільства. У відповідь на зазначені виклики та проблеми, обрана тема дипломної роботи націлена на розробку імплементації, яка поєднує потужність блокчейн-технології з вимогами сучасного обміну файлами. Децентралізована природа блокчейн дозволяє забезпечити високий рівень безпеки, автентифікації та невідворотної історії транзакцій, у той час як обмін файлами стає більш прозорим та ресурсозберігаючим.

Мета дослідження полягає в створенні ефективного механізму, який враховує технічні аспекти розробки, властивості блокчейн-технології та потреби користувачів у сучасному інформаційному середовищі. Засоби децентралізації та розподіленої технології блокчейн формують основу для створення надійної та безпечної інфраструктури обміну файлами, де кожна транзакція фіксується і затверджується розподіленою мережею, що забезпечує відсутність центральних точок вразливості та підвищує стійкість до різноманітних атак.

Дана робота має на меті не лише розробити функціональний прототип системи децентралізованого обміну файлами на базі блокчейн-технології, але й вивчити ефективність та перспективи впровадження такого рішення в реальних умовах.

1 АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ ДЛЯ ОБМІНУ ФАЙЛАМИ

1.1 Основи файлообмінних мереж

Файлообмінна мережа - це система, в якій комп'ютери або інші пристрої підключені один до одного з метою обміну файлами та ресурсами. Ці мережі створюються для того, щоб дозволити користувачам надсилати, отримувати та обмінюватися даними, такими як тексти, зображення, відео, аудіофайли, програми тощо. Основні характеристики файлообмінних мереж:

- Топологія: Файлообмінні мережі можуть мати різні топології, такі як зірка, кільце, шина чи дерево, залежно від конфігурації та вимог користувачів.
- Протоколи обміну файлами: Визначають, як саме відбувається обмін файлами між пристроями. Серед відомих протоколів можна виділити FTP (File Transfer Protocol), SMB (Server Message Block), NFS (Network File System) тощо.
- Безпека: Забезпечення безпеки даних в мережі, наприклад, шляхом використання аутентифікації, шифрування та авторизації для обмеження доступу до файлів.
- Централізовані і децентралізовані рішення: Файлообмінні мережі можуть мати центральний сервер, який контролює доступ та обмін, або працювати за принципом P2P, де кожен вузол може обмінюватися файлами без прямого централізованого керування.
- Системи Керування Версіями: Деякі файлообмінні мережі можуть включати системи керування версіями, які дозволяють відстежувати зміни в файлах та відновлювати попередні версії.
- Рівень Доступу: Контроль рівня доступу до файлів, дозволяючи визначати, які користувачі чи групи можуть переглядати, змінювати або видаляти конкретні файли чи папки.

  Далі описаний процес обміну файлами в файлообмінних мережах:

- Визначення файлів для обміну: Користувач визначає файли, які він хоче

надіслати чи отримати від інших користувачів.

- Вибір адресата чи місця призначення: Вказується адресат чи місце, куди буде відправлено чи звідки буде отримано файл.
- Використання Протоколів Обміну: Використання визначеного протоколу для передачі файлу через мережу.
- Аутентифікація та Авторизація: Забезпечення безпеки шляхом підтвердження ідентичності та визначення прав доступу.
- Підтвердження Та Завершення: Отримання підтвердження про успішний обмін та завершення процесу.
- Основним застосуванням файлообмінних мереж є:
- Бізнес та Корпоративні Системи: Обмін документами, звітами та іншою корпоративною інформацією.
- Освітній Сектор: Доступ до навчальних матеріалів та обмін ресурсами між студентами та викладачами.
- Розваги та Мультимедіа: Обмін музикою, відео, фотографіями та іншими мультимедійними ресурсами.
- Спільноти та Співпраця: Спільний доступ до файлів для роботи над проектами чи завданнями.

Усе це робить файлообмінні мережі ключовим елементом для спільної роботи, обміну інформацією та забезпечення продуктивності в різних сферах діяльності (Рисунок 1.1).

Рисунок 1.1 - Схема процесу обміну файлами в файлообмінних мережах

1.2 Історія розвитку файлообмінних мереж

Розглянемо історію файлових мереж та як вони розвивалися протягом років.

Ранні етапи (1950–1960 рр.): Перші комп'ютери знаходились в етапі активного розвитку, і системи зберігання даних обмежувалися магнітними стрічками та перфорованими картками. Обмін інформацією відбувався через зовнішні пристрої, а ідея "мережі" в контексті файлів була практично невідомою.[1]

Локальні файлові системи (1970–1980 рр.): З появою особистих комп'ютерів, з'явилися й перші локальні файлові системи. Системи, такі як FAT та NTFS від Microsoft, стали стандартами для організації файлів на дисках комп'ютерів. Вони спрямовані на роботу з індивідуальними комп'ютерами та користувачами.[1]

Мережеві файлообмінні протоколи (1980-1990 рр.): Із зростанням популярності мереж та розширенням доступу до комп'ютерів, з'явилися протоколи, такі як FTP та SMB, які стали основними для обміну файлами через мережу. Це відкрило нові можливості для спільної роботи та обміну ресурсами між комп'ютерами.

Клієнт-серверні архітектури (1990-2000 рр.): З ростом об'єму даних та збільшенням обсягу обробки даних, стали потрібні більш ефективні рішення для обміну файлами. Клієнт-серверні архітектури забезпечили ефективний спосіб обміну та управління файлами за допомогою централізованих серверів.

Розвиток Інтернету та хмарних технологій (з 2000 р.): З ростом Інтернету та з'явленням хмарних технологій, файлові мережі отримали нові можливості. Хмарні сховища надають користувачам можливість зберігати та обмінюватися файлами через Інтернет, надаючи високу доступність та мобільність.

Децентралізовані технології (з 2010 р.): З ростом інтересу до децентралізованих технологій, таких як блокчейн, з'явилися децентралізовані файлові системи. Тут керування та обмін файлами здійснюється без централізованого контролю, надаючи більшу безпеку та прозорість.[1]

Ці етапи відображають, як з плином часу файлові мережі еволюціонували від простих систем зберігання до складних та розгалужених мереж, які нині грають

ключову роль у спільній роботі та обміні інформацією.

1.3 Порівняльний аналіз різних моделей обміну файлами

Централізована модель обміну файлами визначається наявністю єдиного центрального сервера, який виступає як ключовий елемент управління та зберіганням даних. Однією з основних характеристик цієї моделі є те, що всі файли та запити на доступ до них обробляються через цю централізовану точку.[2]

Із позитивних сторін централізованої моделі варто відзначити легкість управління та адміністрування. Централізований сервер дозволяє адміністраторам ефективно встановлювати правила безпеки, керувати доступом та впроваджувати політики, що спрощує процеси управління для організації.

З іншого боку, ця модель має свої недоліки, серед яких слід виділити одноочікуваність. Залежність від єдиного сервера означає, що якщо цей сервер вийде з ладу чи стане недоступним, вся система може стати непрацездатною. Це може призвести до значних проблем, особливо якщо організація має велику кількість користувачів та активний обсяг обміну файлами.

Додатково, обмежене масштабування є ще однією важливою недолікою централізованої моделі. При збільшенні обсягу даних та кількості користувачів може виникнути проблема з швидкістю обміну файлами, оскільки центральний сервер обмежений своєю пропускною здатністю.

Також, важко не враховувати витрати на обладнання, які пов'язані з централізованою моделлю. Обробка великої кількості запитів та зберігання значної кількості даних вимагає потужного обладнання, що може призвести до значних витрат для організації.[2]

У висновку, хоча централізована модель обміну файлами має свої переваги у легкому управлінні та адмініструванні, її недоліки, такі як одноочікуваність та обмежене масштабування, потребують уважного врахування при виборі підходу до організації обміну файлами в сучасному бізнес-середовищі (Рисунок 1.2).

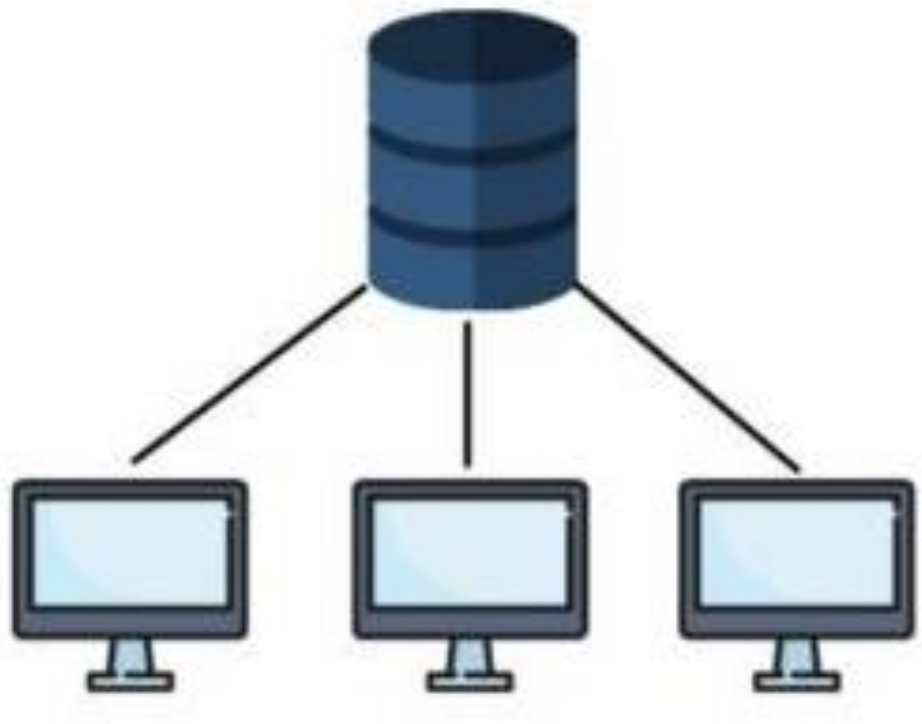


Рисунок 1.2 - Візуалізація централізованої моделі обміну даними

Децентралізована модель обміну файлами — це спосіб організації обміну інформацією, що відрізняється від традиційного централізованого підходу. В основі цього підходу лежить ідея відсутності єдиного центрального сервера, який контролює обмін даними. Замість цього, файли обмінюються через розподілені вузли в мережі, кожен з яких є рівноправним учасником.

Однією з ключових особливостей децентралізованої моделі є розділений реєстр, де кожен вузол утримує свої власні дані та метадані про файли. Це дозволяє кожному вузлу функціонувати автономно, без необхідності постійної взаємодії з центральним сервером.

Зв'язок в децентралізованій моделі здійснюється за допомогою протоколів Peer-to-Peer (P2P), що дозволяє безпосередньо обмінюватися файлами між вузлами. Цей підхід зменшує залежність від центрального сервера та підвищує швидкість обміну даними.[2]

Ще однією важливою складовою децентралізованої моделі є шифрування та

безпека. Інформація шифрується та захищається криптографічними методами, що забезпечує конфіденційність та безпеку файлів під час обміну.

У сферах застосування децентралізованої моделі можна виділити розділені хмарні сервіси, що дозволяють файлам розподілятися між вузлами та забезпечують високу доступність. Блокчейн технології також використовують децентралізовану модель для обміну даними за допомогою "розумних контрактів", які регулюють умови обміну та доступу.

Переваги децентралізованої моделі включають високу надійність, оскільки система залишається функціональною навіть при відмові деяких вузлів, а також конфіденційність та анонімність обміну файлами.

Заключно слід відзначити, що децентралізована модель обміну файлами дозволяє організаціям та користувачам отримати більше гнучкості та контролю в обміні інформацією, але вимагає вдосконалення методів безпеки та управління, оскільки вона впливає на традиційні концепції обміну даними (Рисунок 1.3).

Рисунок 1.3 - Візуалізація децентралізованої моделі обміну даними

Ієрархічна модель в області файлообмінних мереж — це підхід, що визначає структуру системи через упорядкування різних рівнів доступу та управління. Цей підхід базується на ідеї чіткого розподілу обов'язків та повноважень між різними рівнями, надаючи кожному з них конкретну роль в організації файлів та ресурсів.

Великою перевагою ієрархічної моделі є забезпечення чіткості в управлінні. Кожен рівень в ієрархії має свою власну функцію та відповідальність. На вищих рівнях знаходяться стратегічні рішення та глобальне управління, тоді як на нижчих рівнях — конкретні файли та ресурси.

Такий підхід сприяє систематизації ресурсів та створенню організованої структури. Наприклад, кореневий рівень може визначати загальні політики та стандарти для всієї системи, в той час як проміжні рівні можуть фокусуватися на відділах або функціональних областях.

Важливим аспектом є листові рівні, які представляють конкретні ресурси чи файли. На цьому рівні відбувається конкретний обмін інформацією та файлами. Звідси виникає можливість швидкого доступу до необхідних даних на рівні, який відповідає конкретному завданню чи відділу.

Проте існують і обмеження. Ієрархічна модель може виявитися менш гнучкою, оскільки зміни на одному рівні можуть вплинути на всю структуру. Системи, що базуються на цій моделі, можуть бути складні у вдосконаленні та адмініструванні через багато рівнів.

Також важливо враховувати залежність від конкретної системи. Організації повинні взяти до уваги, що міграція до інших платформ чи зміни у структурі можуть виявитися неоднозначними в ієрархічній моделі.

Все це враховуючи, ієрархічна модель виявляється ефективною для багатьох організацій, оскільки дозволяє впорядковано та чітко керувати файлами та ресурсами, але вимагає уважності при реалізації та підтримці.

Далі наведено порівняльний аналіз Централізованої, Децентралізованої та Ієрархічної моделей в обміні файлами.

Централізована Модель:

- Принципи Організації: Централізована модель базується на ідеї, що існує

єдиний центр управління, що контролює та регулює обмін файлами між користувачами.

- Переваги: Централізований Контроль та Керування - одна центральна точка контролю спрощує управління та забезпечує єдину точку доступу та контролю над файлами. Зменшення Ризиків Конфліктів - централізована структура зменшує ймовірність конфліктів та невідповідностей у даних, оскільки всі файли зосереджені в одному місці.
- Недоліки: Одинична Точка Відмови (SPOF) - можливість втрати доступу до всієї системи при відмові центрального сервера. Обмежена Масштабованість та Швидкість - зміна масштабів та швидкість доступу може стати проблемою при великому обсязі даних або багатьох користувачах.
- Застосування: Централізована модель ефективна для ситуацій, де потрібен строгий контроль над доступом та безпекою, таких як банківські системи чи корпоративні мережі.

Децентралізована Модель:

- Структура та Принципи: В децентралізованій моделі відсутня центральна точка контролю, і файли обмінюються між різними вузлами, кожен з яких має рівноправний статус.
- Сфери Застосування та Переваги: Гнучкість та Автономність: Відсутність центрального контролю дозволяє системі бути більш гнучкою та адаптивною до змін. Висока Надійність: Децентралізована структура забезпечує високий рівень надійності, оскільки відмова одного вузла не впливає на всю систему.
- Недоліки: Складне Управління та Координація: Управління та координація можуть стати складнішими, особливо при великій кількості вузлів. Можливість Конфліктів та Безпекових Проблем: Більша автономність може призвести до конфліктів та проблем у забезпеченні безпеки.
- Застосування: Децентралізована модель ефективна в умовах, де важлива автономність та гнучкість, таких як блокчейн-мережі або системи розподіленого обчислення.

Ієрархічна Модель:

- Основи Ієрархії: Ієрархічна модель організована у вигляді ієрархії, що включає кореневий рівень, проміжні рівні та листові рівні.
- Можливості та Обмеження: Чітке Розподілення Обов'язків: Ієрархічна структура забезпечує чітке розподілення обов'язків та повноважень для ефективного управління. Складніше Управління та Масштабованість: Збереження балансу між рівнями може стати складнішим, особливо зі зростанням обсягів даних та числа вузлів.
- Недоліки: Обмежена Гнучкість та Масштабованість: Системи, що базуються на ієрархічній моделі, можуть бути менш гнучкими та менш здатними до швидкого реагування на зміни. Можливість Загальної Відмови: Відмова на вищих рівнях може призвести до відмови всієї системи.
- Застосування: Ієрархічна модель широко використовується в організаціях, де потрібне чітке розподілення влади та обов'язків, таких як корпоративні структури та управління великими підприємствами.
- Загальний Висновок: Централізована, децентралізована та ієрархічна моделі мають свої унікальні переваги та недоліки, які визначають їх відповідність конкретним сценаріям використання. Вибір моделі залежить від конкретних потреб організації, розмірів мережі, потреб у безпеці та гнучкості управління. Ефективне використання систем обміну файлами вимагає ретельного аналізу властивостей кожної моделі та їх адаптації до конкретного контексту (Рисунок 1.4).

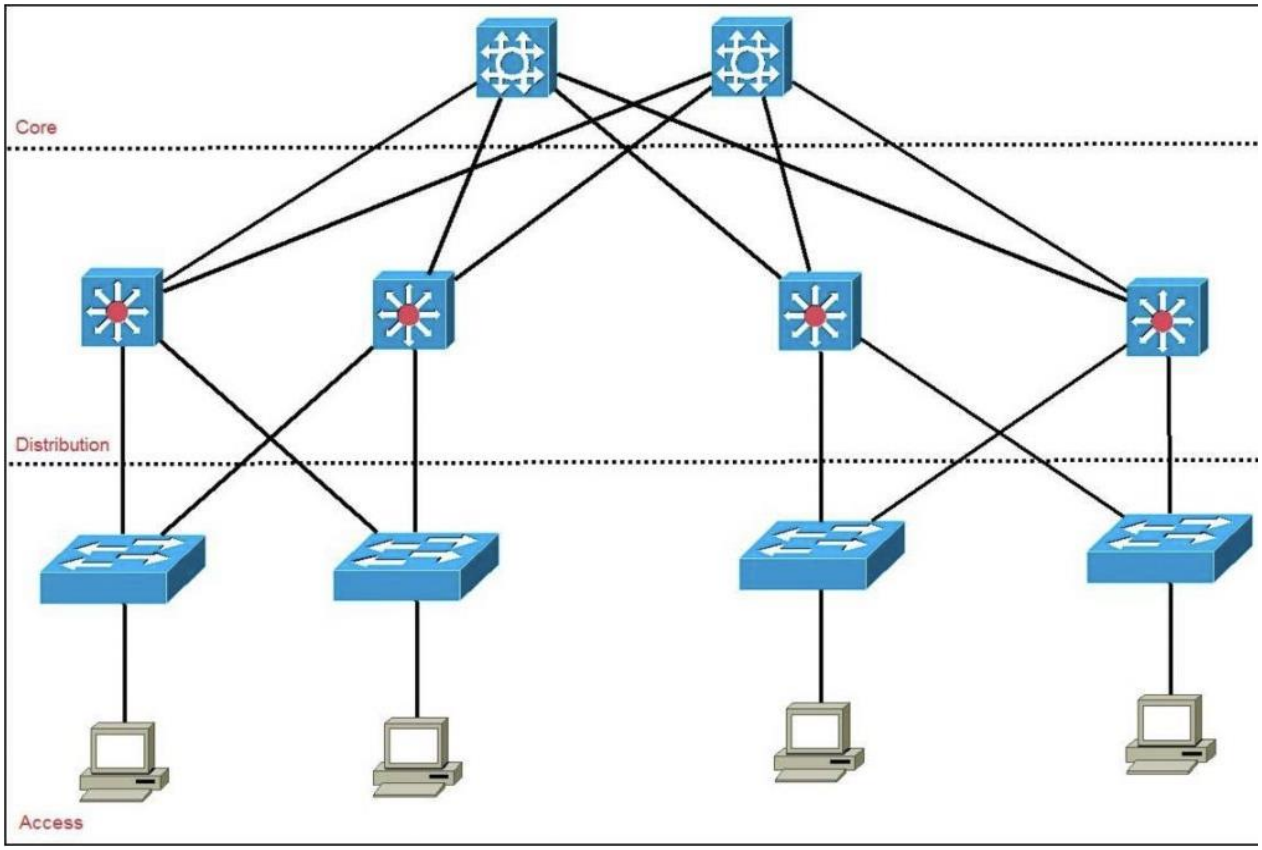


Рисунок 1.4 - Візуалізація ієрархічної моделі обміну даними

1.4 Основні принципи роботи файлообмінників

Далі розглянемо основні принципи роботи файлообмінників.

Реєстрація та Авторизація: Перший крок в роботі будь-якого файлообмінника - це процес реєстрації користувача. Після створення облікового запису користувач отримує можливість авторизації для доступу до сервісу.[3]

Завантаження та Збереження Файлів: Ключовий аспект - це можливість завантажувати файли на сервер файлообмінника. Файли зберігаються в безпечному середовищі, готові для обміну або спільної роботи.

Обмін та Спільна Робота: Файлообмінники надають користувачам можливість обмінюватися файлами та спільно працювати над ними. Це може включати коментування, виправлення та редагування файлів у реальному часі.

Безпека та Конфіденційність: Однією з головних пріоритетних задач є забезпечення високого рівня безпеки та конфіденційності. Застосування

шифрування та управління доступом гарантує захист файлів.

Інтерфейс та Навігація: Зручний інтерфейс та проста навігація дозволяють користувачам швидко знаходити, переглядати та взаємодіяти з файлами. Це важливо для комфортного використання сервісу.

Мобільність та Синхронізація: Популярні файлообмінники мають мобільні додатки, що дозволяють користуватися сервісом на різних пристроях. Синхронізація файлів забезпечує їх актуальність на всіх платформах.

Моніторинг та Аналітика: Деякі файлообмінники ведуть журнал дій користувачів та можуть надавати аналітичні звіти щодо використання файлів. Це полегшує відстеження активності та використання ресурсів.

Інтеграція з Іншими Сервісами: Здатність інтегруватися з іншими сервісами або платформами дозволяє зручно обмінюватися файлами та інформацією.

Інші Функції та Покращення: Функції, такі як резервне копіювання, вбудовані редактори та система повідомлень, додають додатковий функціонал та підвищують зручність використання.

Ці принципи становлять основу ефективного та безпечного обміну файлами через файлообмінники, а їх реалізація дозволяє користувачам ефективно використовувати ці сервіси.[3]

Серед популярних платформ для обміну файлами, включаючи децентралізовані, можна виділити наступні:

Google Drive. Тип: Централізована. Особливості: Інтеграція з Gmail, спільна робота в режимі реального часу, автоматична синхронізація.

Dropbox. Тип: Централізована. Особливості: Простий інтерфейс, можливість створювати посилання на файли, автоматична синхронізація.

Microsoft OneDrive. Тип: Централізована. Особливості: Інтеграція з Microsoft 365, спільна робота над документами, автоматичне створення резервних копій.

Apple iCloud. Тип: Централізована. Особливості: Синхронізація з Apple-пристроями, автоматична резервна копія фотографій та додатків.

BitTorrent. Тип: Децентралізована. Особливості: Peer-to-peer обмін файлами, велика швидкість завантаження, безпека через розподільну мережу.

IPFS (InterPlanetary File System). Тип: Децентралізована. Особливості: Зберігання файлів на різних вузлах мережі, відсутність централізованого сервера.

Sia. Тип: Децентралізована. Особливості: Зберігання файлів у розподіленій мережі блокчейну, шифрування та безпека.

Filecoin. Тип: Децентралізована. Особливості: Використання технології блокчейн для обміну і зберігання файлів, система винагород за майнінг.

Syncthing. Тип: Децентралізована. Особливості: Синхронізація файлів peer-to-peer, шифрування та приватність.

MEGA. Тип: Централізована. Особливості: Зашифрована синхронізація та зберігання файлів, можливість надавати доступ за посиланням.

Це лише декілька прикладів платформ, які представляють різні підходи до обміну файлами, і кожна з них має свої унікальні особливості та переваги.

2 АНАЛІЗ ТЕХНОЛОГІЇ BLOCKCHAIN

2.1 Типи технології розподіленого реєстру DLT

Концепція технології розподіленого реєстру (DLT) охоплює те, як криптографія та відкритий розподілений реєстр можуть бути інтегровані в цифровий бізнес. DLT відноситься до технічної інфраструктури та протоколів, які забезпечують одночасний доступ, перевірку та оновлення записів у незворотній спосіб через мережу, що охоплює кілька організацій або місць. Це може бути блокчейн, спрямований ациклічний граф, хеш-граф, голограма.

1. Блокчейн - це тип DLT, в якому записи про транзакції зберігаються в реєстрі у вигляді ланцюжка блоків. Всі вузли ведуть спільний реєстр, тому загальна обчислювальна потужність розподіляється між ними, що забезпечує кращий результат. Підвищена безпека: Всі дані глибоко зашифровані (хешовані), що забезпечує вищий рівень безпеки. Це унеможливлює злам.[4]

Підтримує широкий спектр алгоритмів консенсусу, які допомагають вузлам прийняти правильне рішення. Як тільки транзакція відбувається, вузли мережі перевіряють її. Після перевірки транзакція отримує унікальний хеш-ідентифікатор і зберігається в реєстрі. Після того, як її додано до реєстру, ніхто не може змінити або видалити транзакцію.

2. Спрямований ациклічний граф. Кожна транзакція вноситься до реєстру у послідовному порядку. Однак, щоб вважатися дійсною, транзакція повинна підтвердити дві попередні транзакції, щоб вважатися дійсною. Послідовність транзакцій називається "гілкою", і чим довшою вона є, тим більш дійсними стають всі транзакції. Практично нескінченна масштабованість: Чим більше користувачів використовують мережу, тим менше часу потрібно для перевірки. Саме так вона може досягти нескінченного рівня масштабованості. Використання одноразових підписів робить її квантово-стійкою.Всі транзакції вирівнюються в паралельну лінію після валідації. [4]

3. Хеш-граф використовує протокол Gossip для передачі інформації про

транзакцію. Як тільки транзакція відбувається, сусідні вузли діляться цією інформацією з іншими вузлами, і через деякий час всі вузли будуть знати про транзакцію. За допомогою протоколу "Віртуального голосування" кожен вузол підтверджує транзакцію, після чого транзакція додається до реєстру. У реєстрі може зберігатися декілька транзакцій з однією і тією ж позначкою часу. Всі транзакції зберігаються в паралельній структурі. Тут кожен запис у книзі називається "подією".

Журнал реєструє кожну послідовність пліток в мережі в упорядкованому порядку. Вузли випадковим чином пліткують про те, що вони знають, з іншими вузлами, щоб поширити інформацію до тих пір, поки кожен вузол не дізнається про неї.

4. Голохейн. У Holochain кожен вузол мережі веде власний реєстр. Хоча система не передбачає жодного глобального протоколу перевірки, мережа підтримує набір правил під назвою "ДНК" для перевірки кожного окремого реєстру. Холочейн переходить від структур, орієнтованих на дані, до структур, орієнтованих на агентів. Вузли, орієнтовані на агентів, можуть перевіряти індивідуально без будь-якого протоколу примусового консенсусу. Енергоефективність: Інша природа реєстру робить систему більш енергоефективною, ніж інші її аналоги. Справжній DLT: кожен вузол в мережі веде свій власний реєстр, що створює справжню розподілену систему.[4]

5. Логічний годинник Tempo (Radix): Для досягнення консенсусу система покладається на послідовність транзакцій, а не на часові мітки. Шардинг: Кожен вузол мережі зберігає шард (найменший розділ глобального реєстру) з унікальним ідентифікатором. Протокол пліток: Вузли передають всю свою інформацію для синхронізації власних шардів. З наведеного вище порівняння випливає, що блокчейн, завдяки деяким своїм властивостям, таким як децентралізація, прозорість інформації, відкритість і захищеність від несанкціонованого втручання, добре підходить для управління ідентифікаційними даними.

2.2 Історія розвитку технології Blockchain

Блокчейн здійснив революцію в багатьох сферах з моменту свого виникнення (Рисунок 2.1).

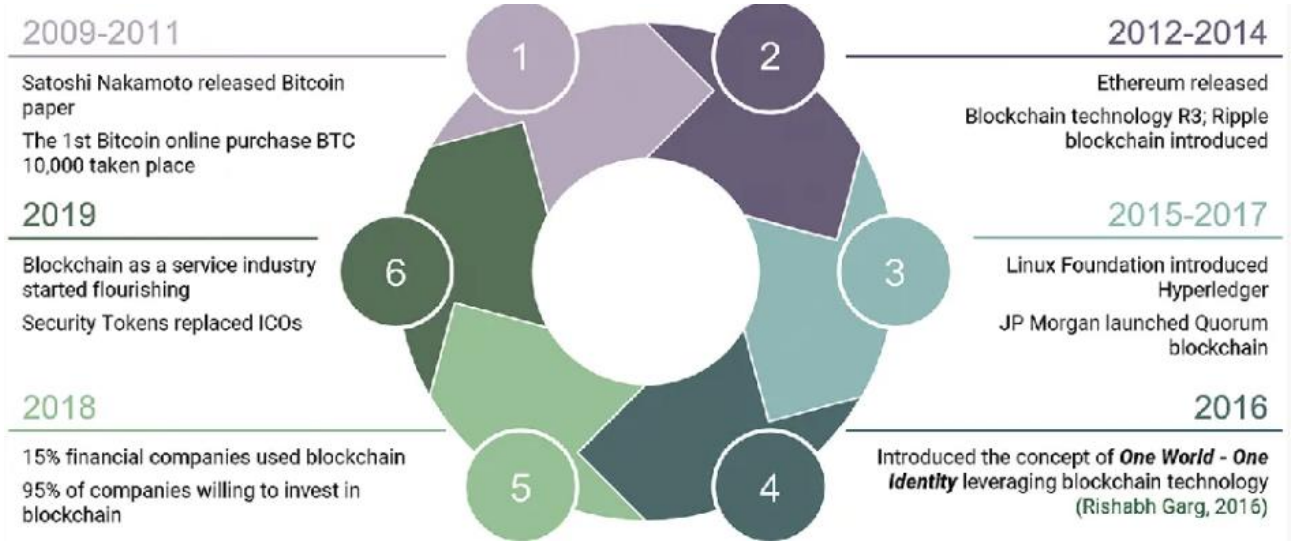


Рисунок 2.1 - Хронологія blockchain

Огляд літератури дає зрозуміти, що сфера застосування блокчейну розширилася від віртуальних валют до фінансових додатків, освіти, охорони здоров'я, науки, транспорту та державного управління. На сьогодні визначено три покоління блокчейну, а саме: блокчейн 1.0 для цифрової валюти, блокчейн 2.0 для цифрових фінансів, блокчейн 3.0 для цифрового суспільства та блокчейн 4.0 як сприятлива для бізнесу екосистема для реального світу.

Блокчейн 1.0 обмежувався віртуальними валютами, такими як біткойн, який був першою і найбільш поширеною цифровою валютою. Більшість додатків Blockchain 1.0 були цифровими валютами, які використовувалися в комерційних транзакціях і покладалися на криптовалютну екосистему для здійснення дрібних платежів, обміну валют, азартних ігор та відмивання грошей.

Спочатку віртуальні валюти покладалися на методи шифрування для генерації, транзакцій і перевірки їхньої вартості. Криптовалютні транзакції записуються в блокчейні, який перевіряє і затверджує кожну транзакцію,

використовуючи ресурси великої однорангової мережі. Основна перевага використання блокчейну для цифрової валюти полягає в тому, що він забезпечує надійну платформу для передачі активу без залучення довірених посередників або контрагентів.

Перший блок був створений Накамото (2008), який представив першу криптовалюту - біткойн. Його поточна ринкова капіталізація становить понад 140 мільярдів доларів США. Інші криптовалюти - Litecoin (2011), Namecoin (2011), Dogecoin (2013) та Peercoin (2012) - зберігають свою присутність із загальною ринковою капіталізацією понад 30 мільярдів доларів США. На даний момент на ринку існує понад 18 000 криптовалют із загальною ринковою капіталізацією 3,2 трильйона доларів США. [5]

Блокчейн 2.0. Цей рівень блокчейну в першу чергу включає Біткоїн 2.0, смарт-контракти, децентралізовані додатки (DApps), децентралізовані автономні організації (DAO) та децентралізовані автономні корпорації (DAC). Однак її використовували для підриву традиційних валютних і платіжних систем у виняткових сферах фінансів, насамперед у банківській справі, біржовій торгівлі, кредитних системах, фінансуванні ланцюгів поставок, платіжному клірингу, боротьбі з підробками та взаємному страхуванні. З'явилися певні криптовалюти, засновані на програмованих смарт-контрактах, такі як Ethereum, Codius і Hyperledger.[5]

Блокчейн 3.0. Блокчейн розглядається як проект нової економіки. Він може застосовуватися в таких секторах, як освіта, охорона здоров'я, наука, транспорт і логістика, окрім валюти і фінансів. Масштаби цього типу блокчейну та його потенційні застосування свідчать про те, що технологія блокчейн є постійною метою. Вона охоплює більш досконалу форму смарт-контрактів для створення розподіленої організаційної одиниці, яка формулює власні закони і працює з високим ступенем автономії.

Злиття блокчейну з токенами є життєво важливою комбінацією Blockchain 3.0. Токен - це засвідчення цифрових прав, і тому токени блокчейну отримали широке визнання завдяки Ethereum та його стандарту ERC20.

На основі цього стандарту будь-хто може випустити власний токен в Ethereum, і цей токен може позначати будь-яке право або цінність. Токени позначають економічну діяльність, створену за допомогою зашифрованих токенів, які в основному, але не виключно, базуються на стандарті ERC20.

Токени можуть слугувати формою підтвердження будь-якої кількості прав, включаючи особисту ідентичність, дипломи про освіту, валюту, квитанції, ключі, дисконтні бали, ваучери, акції та облігації.

Можна стверджувати, що токени є його зовнішнім економічним обличчям, тоді як блокчейн - це внутрішня технологія нового часу.

Блокчейн 4.0. Блокчейн 4.0 зараз розвивається як зручна для бізнесу екосистема для створення та експлуатації додатків у реальному світі. Система може досягти нескінченної масштабованості, досліджуючи можливості віртуального блокчейну всередині блокчейну. Blockchain 4.0 описує рішення та підходи, які раціоналізують технологію для потреб бізнесу, зокрема Індустрії 4.0 та комерції.

Індустрія 4.0 втілює в собі інтеграцію автоматизації, планування ресурсів підприємства і різних систем виконання, і саме це робить блокчейн корисним для реальних додатків майбутнього. Об'єднання блокчейну з промисловістю та бізнесом дозволяє здійснювати міжсистемні бізнес-процеси, забезпечуючи безпеку та конфіденційність даних за допомогою автоматизації, виключення контрагентів та прозорості.

Інтеграція блокчейну з новими технологіями, такими як Інтернет речей (IoT), хмарні технології, штучний інтелект і робототехніка, є однією з найбільших обіцянок майбутнього. Блокчейн автоматизує процеси, завойовуючи довіру, передаючи.[5]

2.3 Архітектура технології Blockchain

Блокчейн - це послідовність блоків, яка містить повний перелік записів про транзакції, подібно до звичайної публічної книги. Завдяки хешу попереднього блоку, що міститься в заголовку блоку, блок має лише один батьківський блок

(Рисунок 2.2).

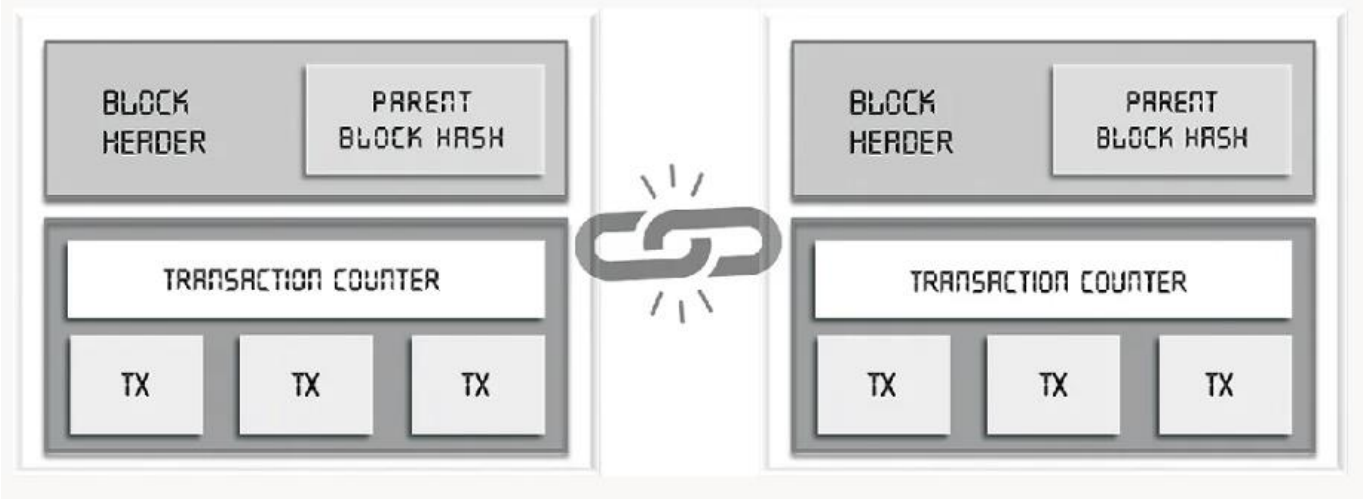


Рисунок 2.2 - Блоки в ланцюгу Blockchain

Перший блок блокчейну називається "генезисним блоком", який не має батьківського блоку. Блокчейн - безперервна послідовність блоків.

Блок складається із заголовка та тіла блоку (Рисунок 2.3).

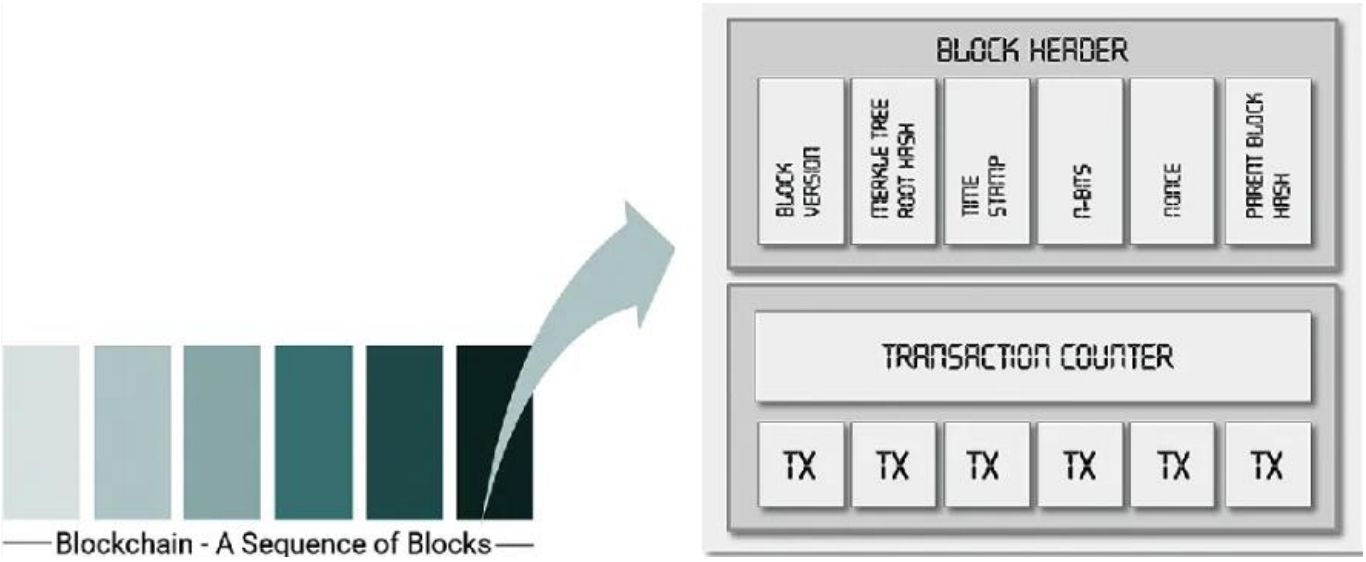


Рисунок 2.3 - Ілюстрація конкретного блоку Blockchain

Заголовок блоку містить:

- Версію блоку, яка позначає набір правил перевірки блоку, яких слід дотримуватися.

- Кореневий хеш дерева Меркла, який вказує на хеш-значення всіх транзакцій в блоці.
- Мітка часу, яка відображає поточний час за всесвітнім часом з 01 січня 1970 року.
- nБіти, які вказують на поріг допустимого хешу блоку.
- Nonce, що позначає 4-байтне поле, яке зазвичай починається з 0 і збільшується для кожного обчислення хешу.
- Хеш батьківського блоку з 256-бітовим хеш-значенням, яке вказує на попередній блок.

Тіло блоку - це блок, що складається з лічильника транзакцій і самої транзакції. Максимальна кількість транзакцій, яку може вмістити блок, залежить від розміру блоку і розміру кожної транзакції. Блокчейн використовує механізм асиметричної криптографії для підтвердження автентичності транзакцій (NRI, 2016). Цифрові підписи, засновані на асиметричній криптографії, використовуються в ненадійних середовищах.[6]

Тіло блоку складається з (Рисунок 2.4):

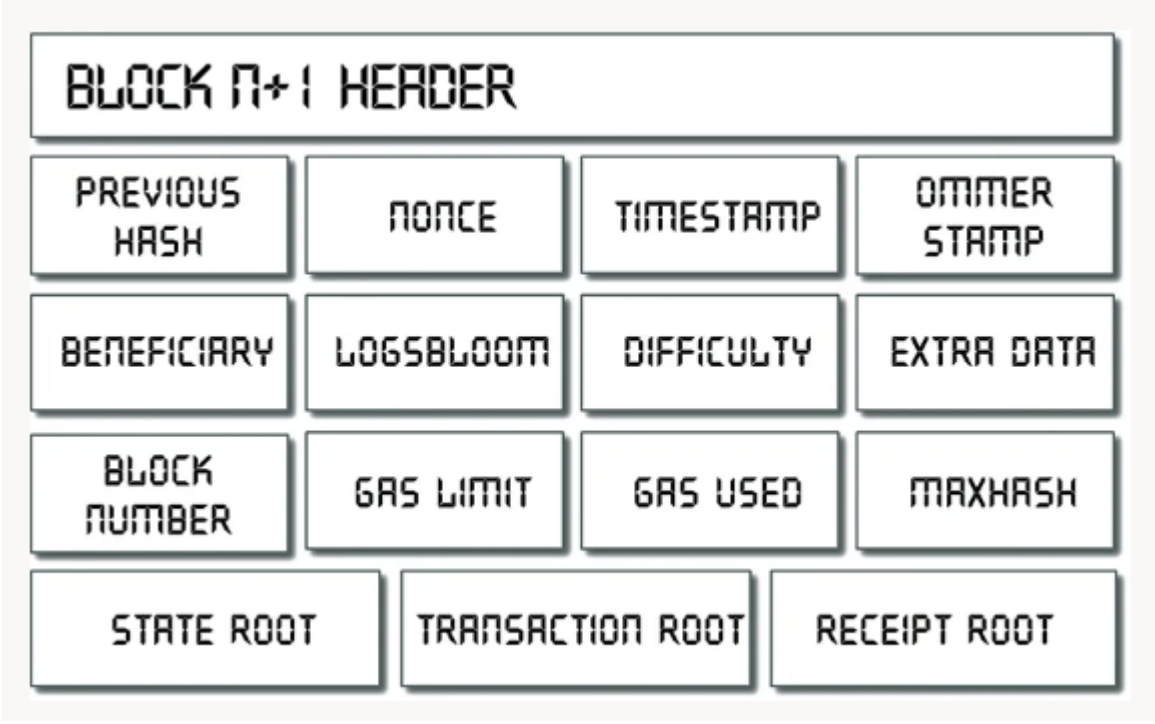


Рисунок 2.4 - Компоненти блоку

- Хеш транзакції
- Корінь транзакції
- Хеш стану
- State Root

Для перевірки цілісності блоку використовуються хеші всіх компонентів. Для валідації блоку перевіряються додаткові умови.

Як описано вище, кожен блок містить хешовану інформацію (Рисунок 2.5).

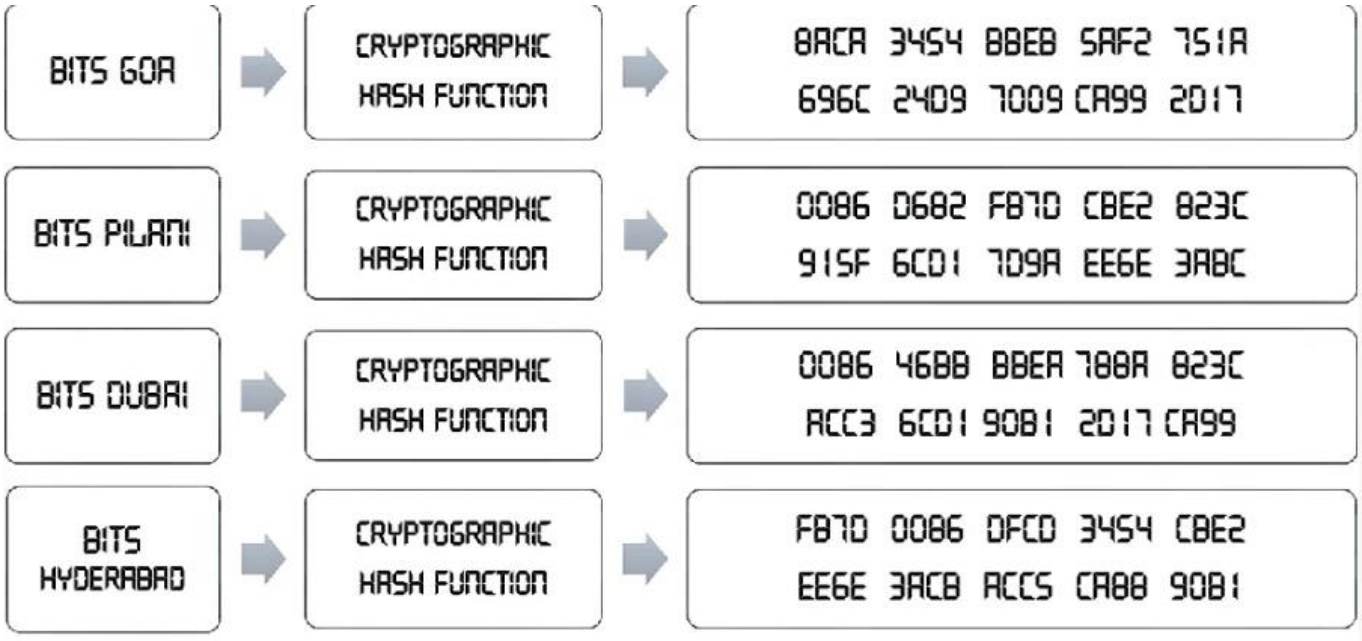


Рисунок 2.5 - Криптографічна хеш-функція

Хеш-функція, що використовується в криптографії, відображає математичний алгоритм, який перетворює біти інформації в рядок алфавітно-цифрових значень. Унікальний сам по собі, хеш вказує на те, чи збігається інформація на вході та виході. Хеш-функції мають високий лавинний ефект, тобто невелика зміна вхідного рядка призводить до зовсім іншого вихідного рядка. Це робить майже неможливим зворотний інжиніринг, якщо не застосовувати складні алгоритми машинного навчання.

Найпоширенішими функціями хешування є SHA-3, SHA-256 і Keccak. Існує

два типи методів хешування: Просте хешування: хешування виконується методом грубої сили. Він підходить для документів/блоків з фіксованою кількістю вмісту/транзакцій, наприклад, заголовка блоку.

Хешування деревом Меркла: Дані розташовуються у вигляді вузлів листків (бінарне дерево), а хешування виконується наборами з двох листків. Використовується для змінного розміру аргументів у хеш-функції. Наприклад, цілісність складеного блоку; O(n) = log n.

Шифрування. Перший випадок шифрування був представлений як шифрування з симетричним ключем. У цьому випадку той самий ключ (правило) використовується для шифрування і розшифрування. Наприклад, шифрування Цезаря, в якому алфавіті (значення ASCII) зсуваються на фіксовану величину.[6]

Важко передати ключ для розшифрування. Для вирішення вищезгаданих проблем була введена криптографія з відкритим ключем або асиметричне шифрування. У цьому кожен користувач має пару ключів. Повідомлення, зашифроване за допомогою закритого ключа відправника разом з відкритим ключем одержувача, розшифровується за допомогою закритого ключа одержувача і відкритого ключа відправника. Це дозволяє автентифікувати як відправника, так і одержувача. Алгоритм Rivest Shamir Adelman (RSA) використовує концепцію асиметричного шифрування, щоб допомогти користувачам увійти до віртуальної машини в хмарі, наприклад, Amazon Web Services. Однак у блокчейні для побудови пари публічного та приватного ключів використовуються алгоритми криптографії еліптичних кривих (сімейство ECC), оскільки вони набагато стійкіші за RSA (256-бітна пара ключів ECC ≡ 3072-бітна пара RSA).

Для захисту децентралізованих ідентифікаторів приватні ключі відомі лише власнику, в той час як публічні ключі широко розповсюджуються. Таке поєднання досягає двох цілей. Перша - автентифікація, коли публічний ключ підтверджує, що власник парного приватного ключа надіслав повідомлення. Друга - шифрування, коли тільки власник парного закритого ключа може розшифрувати повідомлення, зашифроване за допомогою відкритого ключа. Цей процес називається "криптографія".

Вузли в мережі ідентифікуються за допомогою адрес облікових записів, які генеруються наступним чином: Генерується 256-бітове випадкове число, яке позначається як приватний ключ.

Алгоритм ECC застосовується до приватного ключа для генерації публічного ключа.

До відкритого ключа застосовується хеш-функція для створення 20-байтової адреси облікового запису. Дані транзакції, які хешуються і шифруються, діють як цифровий підпис. Одержувач отримує оригінальну копію даних і захищений хеш цих даних. Одержувач розшифровує захищений хеш і повторно обчислює хеш отриманих оригінальних даних. Якщо обидва хеші збігаються, це означає, що отримані дані не були змінені.

Децентралізований ідентифікатор (DID) - це псевдо анонімний ідентифікатор особи або компанії. Кожен DID захищений приватним ключем. Тільки власник приватного ключа може встановити, що він володіє або контролює свою ідентичність. Одна особа може мати кілька DID, що обмежує ступінь, до якого її можна відстежити в різних сферах її життєдіяльності.

Кожен DID часто асоціюється з низкою посвідчень (атестатів), виданих іншими DID, які підтверджують відмінні характеристики цього DID (наприклад, вік, дипломи, платіжні квитанції, адреса), які можна перевірити. Ці посвідчення мають криптографічний підпис їхніх емітентів, що дозволяє власникам DID зберігати ці посвідчення самостійно як альтернативу єдиному провайдеру профілю.

DID зберігається в публічному реєстрі разом з документом DID, який містить публічний ключ для DID, будь-які інші публічні облікові дані (які власник посвідчення бажає оприлюднити) та мережеві адреси для взаємодії (Рисунок 2.6).

Рисунок 2.6 - Децентралізований ідентифікатор (DID)

Загальна схема роботи Blockchain (Рисунок 2.7).

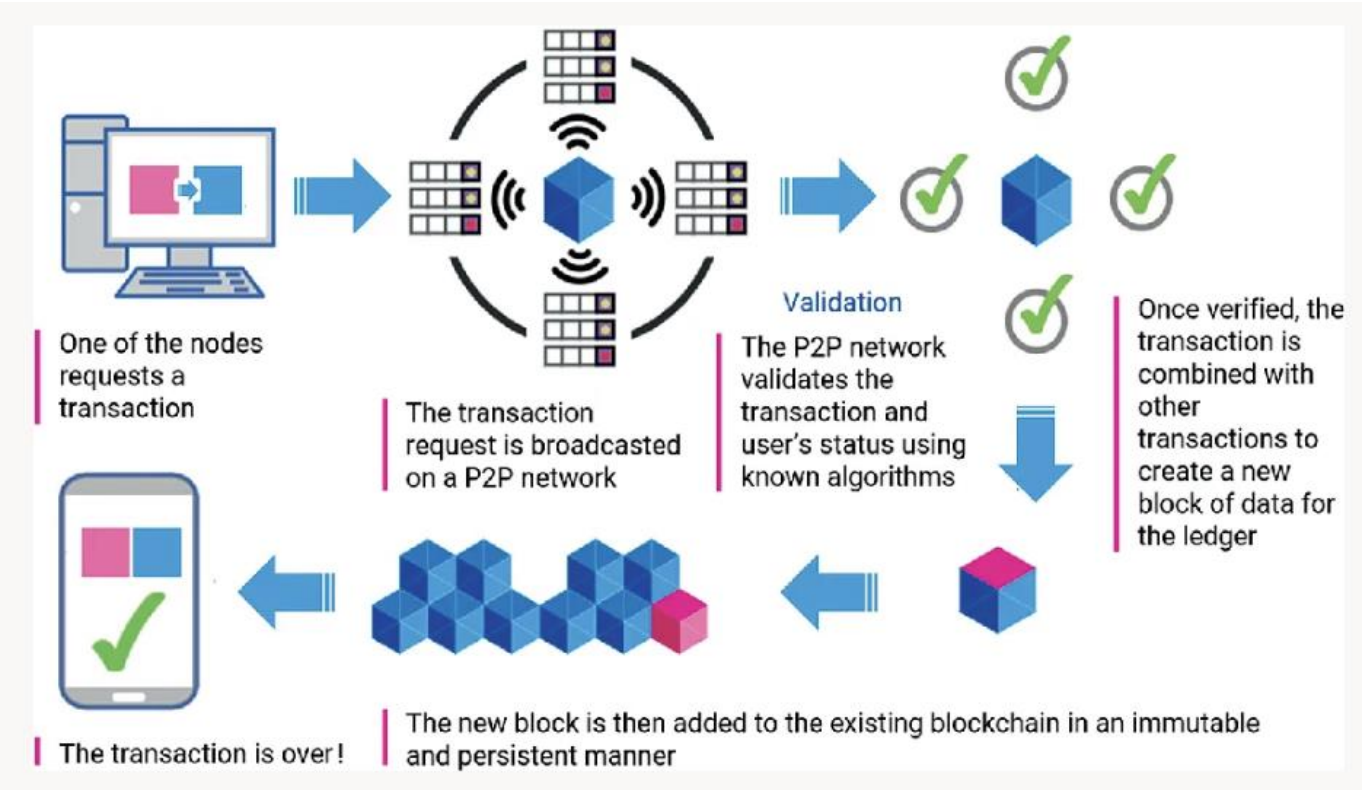


Рисунок 2.7 - Загальна схема роботи Blockchain

2.4 Ключові характеристики Blockchain

Блокчейн має такі ключові характеристики (Рисунок 2.8):

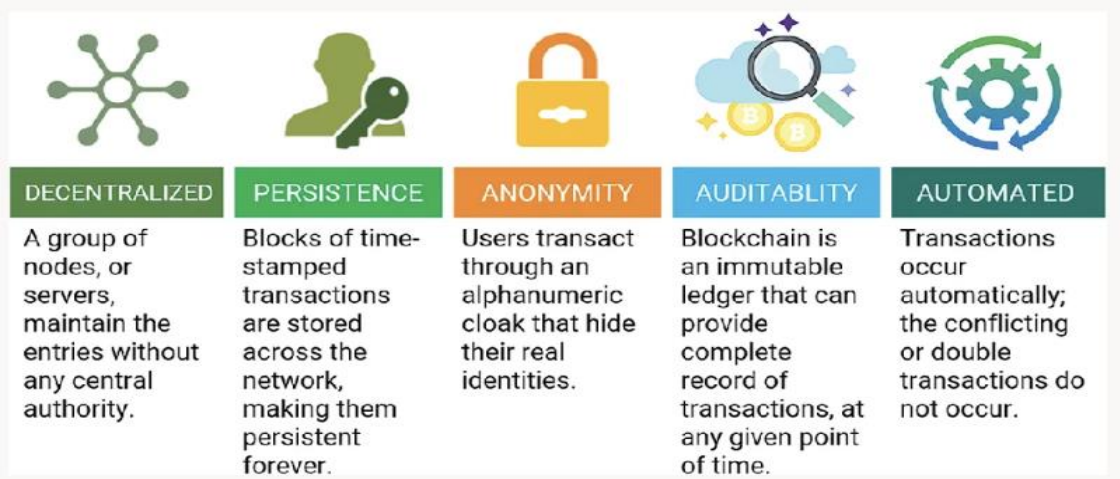


Рисунок 2.8 - Ключові характеристики блокчейну

- Децентралізація. У блокчейні для підтвердження транзакцій не потрібен централізований орган. Алгоритми консенсусу підтримують узгодженість даних у розподілених мережах.
- Незмінність. Практично неможливо видалити або відкотити транзакцію після її додавання до блокчейну. Транзакції перевіряються швидко, а недійсні транзакції не допускаються чесними майнерами. Тому блоки, які містять недійсні транзакції, можна виявити негайно.
- Анонімність. Кожен користувач може здійснювати транзакції в блокчейні, використовуючи згенеровану адресу, яка не розкриває справжню особу користувача.
- Можливість аудиту. Блокчейн зберігає дані про баланси користувачів на основі моделі Unspent Transaction Output (UTXO). Будь-яка транзакція повинна посилатися на деякі попередні невитрачені транзакції.

Сучасні блокчейн-системи зазвичай поділяють на три великі категорії: публічний блокчейн, приватний блокчейн і консорціумний блокчейн.

Публічний блокчейн. У публічному блокчейні всі записи є видимими для громадськості, і кожен вузол може брати участь у процесі досягнення консенсусу. Публічні блокчейни також відомі як блокчейни без прав доступу. У публічному блокчейні будь-який користувач, який бажає здійснювати транзакції з мережею, може брати участь і писати в блокчейні. Це дуже корисно для тих, хто не

покладається на єдиний центральний орган влади. Технологія блокчейн використовує децентралізований механізм консенсусу для забезпечення надійності та узгодженості даних і транзакцій. У неліцензійному блокчейні майнери використовують різні алгоритми для перевірки транзакцій. Транзакції записуються і додаються до блокчейну, коли більшість вузлів досягають консенсусу і схвалюють їх.

Загальнодоступний блокчейн використовує доказ роботи, оскільки він гарантує, що транзакції не можуть бути підроблені, доки жоден майнер не контролює більше 50% хеш-потужності мережі. Автоматизований механізм консенсусу дозволяє оновлювати кожну копію реєстру майже миттєво. Як тільки транзакція додається до блокчейну, оскільки всі вузли мають спільну копію реєстру, всі реєстри відображають цю зміну.

Приватний блокчейн. На противагу цьому, приватні блокчейни дозволяють брати участь у процесі консенсусу лише вузлам, що належать до певної організації. Їх також називають "дозволеними блокчейнами". Оскільки вони повністю контролюються однією організацією, їх часто сприймають як централізовану мережу. У дозволеному блокчейні повноваження доступу, перевірки та додавання транзакцій до реєстру делеговані обмеженій групі осіб. Існує також можливість того, що в дозволеному блокчейні деяким користувачам може бути дозволено переглядати всі транзакції в реєстрі, але вони можуть не мати права записувати жодної транзакції. Користувачі можуть переглядати інформацію і транзакції в реєстрі залежно від рівня доступу. У дозволеному блокчейні рівні доступу і ролі користувачів заздалегідь визначені. Після того, як сторони, що беруть участь у транзакції, відправляють дані, вони підтверджуються іншими дозволеними учасниками блокчейну. Оскільки приватні механізми блокчейну обмежуються лише організацією або групою людей, які беруть участь у транзакції, легко досягти консенсусу.[6]

Для того, щоб найкращим чином використовувати як публічні, так і приватні блокчейн-рішення, можна також застосувати гібридну модель блокчейну.

Консорціумний блокчейн. Консорціумний блокчейн дозволяє групі

попередньо відібраних вузлів брати участь у процесі консенсусу. Оскільки для визначення консенсусу обирається лише невелика частина вузлів, консорціумний блокчейн, створений кількома організаціями, є дещо децентралізованим.

2.5 Алгоритм консенсусу

Механізм консенсусу - це відмовостійкий механізм, який використовується для досягнення згоди щодо стану блокчейну з метою забезпечення дійсності та автентичності транзакцій.

Для того, щоб зрозуміти ідеологію, що лежить в основі створення блокчейну, необхідно почати з класичної задачі в розподіленій системі, широко відомої як "Проблема візантійських генералів" (BGP). Згідно зі схемою BGP, кілька армій збираються для нападу на замок. Замок може бути захоплений лише у тому випадку, якщо всі армії атакують в один і той же момент часу. Припустимо, що головна армія наказує всім іншим арміям атакувати в заздалегідь визначений час через гінця. Гонець може бути захоплений під час перевезення, і таким чином наказ про атаку замку ніколи не буде поширений. Щоб переконатися, що повідомлення було доставлено, відправник може попросити про підтвердження, але і тут є ймовірність, що того, хто це зробить, також можуть затримати.

Тому необхідно досягти консенсусу, який би засвідчив, що (i) відправник повідомлення про напад знає, що всі інші армії отримали це повідомлення; і (ii) кожна армія, яка отримує це повідомлення, підтверджує, що всі інші армії отримали це повідомлення. Тепер припустимо, що замість того, щоб відправити гінця, генерал надсилає це повідомлення про атаку через блокчейн. Атака запланована на шість годин. Доказ роботи, який використовується в цьому блокчейні, полягає в тому, що якщо всі армії будуть працювати над вирішенням проблеми в один і той же момент часу, то для появи першого рішення знадобиться близько десяти хвилин. Як тільки генерал, який відправив повідомлення про атаку, знайде дійсні рішення Proof-of-Work, що з'являються майже кожні десять хвилин, він може бути впевнений, що всі інші армії отримали це повідомлення. Малоймовірно, що кілька

армій можуть створювати правильні рішення з такою швидкістю. В той же час, всі інші армії будуть повністю впевнені, що всі інші армії бачили повідомлення про напад, враховуючи швидкість, з якою створюються рішення, що підтверджують достовірність роботи.

Така ж логіка застосовується і до ідеї розподіленого реєстру. Подібно до того, як блокчейн-рішення для BGP гарантує, що всі сторони знають, що всі інші сторони бачили повідомлення, ця логіка може бути використана для перевірки того, що всі сторони згодні з поточним станом реєстру.

Для досягнення консенсусу використовуються такі підходи:

Proof-of-Work (PoW) - це стратегія досягнення консенсусу, яка використовується в мережі Біткоїн (Nakamoto, 2008). У децентралізованій мережі для запису транзакцій повинна бути обрана певна стратегія, для якої найпростішим способом є випадковий вибір. Однак, випадковий вибір вразливий до атак. Отже, якщо вузол хоче опублікувати блок транзакцій, потрібно виконати багато роботи (обчислень), щоб довести, що від цього вузла не очікується атаки на мережу.

У PoW кожен вузол мережі обчислює хеш-значення заголовка блоку. Заголовок блоку містить nonce, і майнери будуть постійно змінювати nonce, щоб отримати різні хеш-значення. Консенсус вимагає, щоб обчислене значення дорівнювало або було меншим за певне задане значення. Коли один вузол досягає цільового значення, він транслює блок іншим вузлам, і всі інші вузли повинні взаємно підтвердити точність хеш-значення. Якщо блок підтверджено, інші майнери приєднують цей новий блок до свого блокчейну. Вузли, які обчислюють хеш-значення, називаються "майнерами", а процедура PoW - "майнінгом". У децентралізованій мережі дійсні блоки можуть генеруватися одночасно, коли кілька вузлів знаходять відповідний нонс майже одночасно. В результаті можуть утворюватися гілки.

Навіть у цьому випадку малоймовірно, що дві конкуруючі гілки сформують наступний блок одночасно. У протоколі PoW ланцюжок, який стає довшим, вважається автентичним. Розглянемо два форки, створені одночасно валідованими блоками U4 і B4. Майнери продовжують видобувати свої блоки до тих пір, поки не

згенерується довша гілка. Оскільки B4-B5 утворює довший ланцюжок, майнери на U4 переключаться на довшу гілку. Оскільки майнери повинні робити багато комп'ютерних обчислень в PoW, ці роботи витрачають багато ресурсів (Рисунок 2.9).

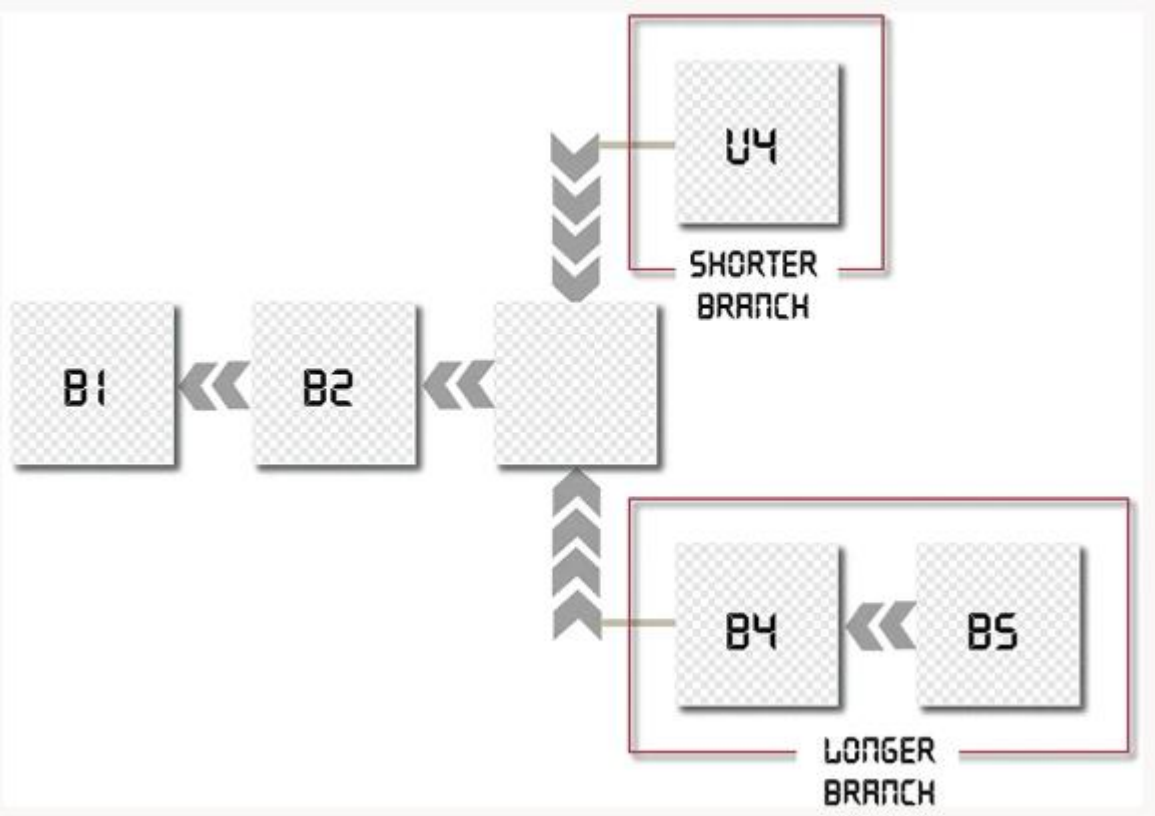


Рисунок 2.9 - Механізм відмовостійкості PoW

Proof-of-Stake (PoS) - це енергозберігаюча альтернатива PoW. Майнери в PoS повинні довести право власності на валюту. Вважається, що люди з більшою кількістю валюти менш схильні до атак на мережу. Однак це припущення не здається розумним, оскільки одна найбагатша людина може домінувати в мережі. Отже, було запропоновано кілька рішень з комбінацією розміру частки, щоб вирішити, кому з них підробляти наступний блок. Наприклад, блеккоїн використовує рандомізацію для прогнозування наступного генератора. Він використовує формулу, яка шукає найнижче хеш-значення в поєднанні з розміром стейка. Багато блокчейнів спочатку використовують PoW, а потім поступово переходять на PoS.[6]

Peercoin сприяє відбору на основі монет. Тут старіший і більший набір монет має більшу ймовірність видобути наступний блок. Але, оскільки вартість майнінгу майже нульова, не можна виключати можливості атак.

Практична візантійська відмовостійкість (PBFT) - це алгоритм реплікації, стійкий до візантійських збоїв. Новий блок визначається за один раунд. У кожному раунді за певними правилами обирається первинний блок, який відповідає за впорядкування транзакцій. Весь процес можна розділити на три фази: попередня підготовка, підготовка і фіксація. На кожній фазі вузол переходить до наступної фази, якщо він отримує більше двох третин голосів всіх вузлів. Обов'язковою умовою є те, що кожен вузол відомий мережі. Подібно до PBFT, Stellar Consensus Protocol (SCP) також є протоколом візантійської згоди. PBFT зобов'язує кожен вузол запитувати інші вузли, в той час як SCP надає учасникам право вибирати, якій групі інших учасників довіряти.

Делегований доказ частки (Delegated Proof-of-Stake, DPoS) є представницькою демократією і відрізняється від PoS, яка є прямою демократією. Зацікавлені сторони обирають своїх представників для створення і перевірки блоку. Оскільки для перевірки блоку потрібно менше вузлів, блок може бути підтверджений швидше, що призводить до більш швидкого підтвердження транзакцій. Параметри мережі, такі як розмір блоку та інтервал між блоками, можуть бути налаштовані делегатами. Крім того, користувачам не потрібно турбуватися про недобросовісних представників, оскільки їх можна легко відсторонити від участі в голосуванні.

Ripple - це алгоритм консенсусу, який використовує довірені підмережі колективно в рамках більшої мережі. У мережі вузли поділяються на два типи: сервери для участі в процесі консенсусу і клієнти, які лише переказують кошти. Кожен сервер має унікальний список вузлів (UNL), який є важливим для сервера. Приймаючи рішення, чи поміщати транзакцію в реєстр, сервер запитує вузли в UNL, і якщо отримана згода досягла 80%, транзакція буде упакована в реєстр. Для одного вузла реєстр буде коректним, враховуючи, що відсоток несправних вузлів в UNL становить менше 20%.

Тендермінт - це алгоритм візантійського консенсусу. У першому раунді визначається новий блок і обирається учасник, який транслюватиме непідтверджений блок. Його можна розділити на три фази: попереднє голосування, попередня фіксація і фіксація. На етапі попереднього голосування валідатори вирішують, чи транслювати попереднє голосування за запропонований блок. На етапі попередньої фіксації визначається, що якщо нода отримала більше двох третин попередніх голосів за запропонований блок, то вона транслює попередню фіксацію для цього блоку. Якщо нода отримує більше двох третин пре-фіксів, вона переходить до фази фіксації. На завершення, вузол перевіряє блок і публікує комміт для цього блоку (фаза фіксації). Якщо вузол отримує дві третини комміту, він приймає блок. На відміну від PBFT, вузли повинні заблокувати свої монети, щоб стати валідаторами. Якщо верифікатор виявиться нечесним, він буде оштрафований.

Добрий алгоритм консенсусу означає ефективність, безпеку і зручність. Останнім часом спостерігається величезний сплеск зусиль по вдосконаленню алгоритмів консенсусу в блокчейні. Розробляється все більше алгоритмів консенсусу для вирішення деяких специфічних проблем блокчейну.

Однією з таких нових ідей є пірінговий консенсус. Його мета - розділити створення блоків і підтвердження транзакцій, щоб збільшити швидкість досягнення консенсусу. Крім того, Крафт (2016) запропонував нову схему консенсусу, яка гарантує, що блок генерується з досить стабільною швидкістю, припускаючи, що висока швидкість генерації блоків може поставити під загрозу безпеку Біткоїна.

У подальшому Сомполінський і Зохар (2013) рекомендували для вирішення цієї проблеми правило вибору ланцюжка Greedy Heaviest-Observed Subtree (GHOST) (Жадібне найважче спостережуване піддерево - GHOST). У цьому методі замість схеми з найдовшою гілкою, GHOST зважує гілки, і майнери можуть вибрати кращу з них, щоб слідувати їй. Чепурной, Ларангейра та Оджиганов (2016) також представили новий алгоритм консенсусу для однорангових блокчейн-систем, де кожен, хто надає неінтерактивні докази відновлюваності для минулих

знімків стану, може згенерувати блок. У такому протоколі майнери повинні зберігати лише старі заголовки блоків, а не цілі блоки.

Проста перевірка платежів (СПП) - це процес, який дозволяє легкому клієнту перевірити, чи є транзакція в блокчейні Біткоїн, без необхідності завантажувати весь блокчейн. Клієнту SPV потрібно завантажити лише заголовки блоку, які набагато менші за весь блок. SPV-клієнт запитує гілку Merkle, щоб перевірити, чи була транзакція включена в блок. SPV-клієнти забезпечують кращий рівень безпеки, ніж веб-гаманці.Ключові особливості легко і практично дізнатися найдовший ланцюжок, не стаючи майнером.

Користувачеві потрібно зберегти лише копію заголовків блоків найдовшого ланцюжка з підтвердженням роботи і отримати гілку Merkle, яка з'єднує транзакцію з блоком.

Хоча користувач не може перевірити транзакцію самостійно, він все ж може побачити, чи була вона схвалена вузлом мережі. Він може прив'язати транзакцію до точки в ланцюжку і зупинити її.

Таким чином, поки мережа контролюється чесними вузлами, верифікація є надійною, хоча вона стає вразливою, як тільки зловмисник отримує контроль над мережею.

Спрощений підхід може бути спотворений або зруйнований, доки зловмисник успішно утримує домінування над мережею.

Єдиний спосіб захистити процес - приймати попередження від вузлів мережі, коли вони виявляють неправильний блок, змушувати програму користувача завантажувати весь блок або повідомляти про транзакції, щоб підтвердити невідповідність.

Повна перевірка платежу вимагає наявності повної копії блокчейну, яку часто називають грубим або важким гаманцем. Це допомагає перевірити, чи біткоїни, використані в транзакції, походять з видобутого блоку. Тут блокчейн сканується транзакція за транзакцією в ретроспективі, поки не буде знайдено джерело. Ці програми-гаманці діють як активні гравці в мережі Біткоїн. Вони не тільки керують транзакціями користувача, але також перевіряють і передають

транзакції інших осіб. У таких ситуаціях комп'ютери, які виконують ці програми, називаються повними вузлами.

Повні вузли - це всі майнери Біткоїна, а це означає, що для видобутку їм потрібна повна копія блокчейну.

На сьогоднішній день блокчейн виріс до розміру ≥15 ГБ і включає 35 мільйонів транзакцій протягом п'яти років. У наступні п'ять років ця технологія, ймовірно, зросте в сотні разів.

Залежно від пропускної здатності, завантаження мережі блокчейн через додаток гаманця Bitcoin може зайняти кілька днів.

Для досягнення консенсусу необхідно, щоб усі учасники (всі вузли) мережі були підключені, щоб представити свої індивідуальні висновки і визначити, який блокчейн має найбільше доказів роботи.

З постійно зростаючим обсягом транзакцій блокчейн стає важким. Через внутрішні обмеження на розмір блоку і часовий лаг для генерації нового блоку, блокчейн Біткоїн може обробляти близько семи транзакцій в секунду, що не відповідає потребі обробляти мільйони транзакцій в реальному часі. Тому може бути лише дві альтернативи: оптимізація сховища і редизайн.

Брюс запропонував нову криптовалютну схему, в якій старі записи про транзакції видаляються (або забуваються) мережею. Це важливо, оскільки вузлу не так просто оперувати повною копією реєстру.

Еяль запропонував Bitcoin-Next Generation розділити звичайний блок на дві частини: основний блок для обрання лідера та мікроблоки для зберігання транзакцій. Час ділиться на епохи, і в кожній епосі майнери повинні хешувати, щоб створити основний блок. Після того, як ключовий блок створено, вузол стає лідером, відповідальним за створення мікроблоку. Bitcoin-NG також розширив стратегію найдовшого ланцюжка з мікроблоками, які не мають ваги. Таким чином, шляхом редизайну блокчейну вирішується проблема компромісу між розміром блоку і безпекою мережі.

2.6 Смарт-контракти

Смарт-контракт - це код програми, який зберігається в блокчейні за певною адресою, відомою як адреса контракту. Додатки можуть викликати функції смарт-контракту, змінювати його стан та ініціювати транзакції. Смарт-контракти пишуться мовами програмування, такими як Solidity (об'єктно-орієнтована мова програмування) і Viper. Для виконання смарт-контрактів на будь-якому обладнанні чи програмному забезпеченні потрібен рівень абстракції. Його забезпечує EVM, який перетворює мову високого рівня в байт-код EVM.

Смарт-контракти задовольняють умови, виконуючи прості оператори "якщо... то", які запрограмовані в блокчейні. Ці умови можуть включати оплату, передачу товарів або виставлення рахунків при виконанні певної умови. Після завершення транзакції блокчейн оновлюється, після чого транзакція не може бути змінена. Результат може бачити лише уповноважена сторона.

На практиці учасники не пишуть новий код щоразу, коли запитується обчислення на EVM, натомість розробники додатків завантажують у сховище EVM фрагменти коду багаторазового використання, а потім користувачі роблять запит на виконання цих фрагментів коду відповідно до різних параметрів.

Смарт-контракти використовуються для автоматизації поширених централізованих процесів, таких як умовна передача цифрових активів, обмін активами з декількома підписами або очікування певного часу для виконання транзакції.

Смарт-контракт представлений контрактним рахунком. Це може бути реалізовано за допомогою зовнішнього облікового запису (EOA), який необхідний для участі в блокчейні Ethereum. Процес запрошення здійснюється за допомогою транзакцій, що надсилаються EOA у вигляді ефіру та газу. Коли цільовою адресою в транзакції є смарт-контракт, виконання смарт-контракту відбувається після верифікації (наприклад, перевірки комбінації nonce і комісії) і валідації транзакції.[7]

У 1994 році комп'ютерний вчений Нік Сабо запропонував ідею смарт-

контрактів як спосіб для комп'ютера санкціонувати транзакції за допомогою рядків коду, які формують протокол, який він застосовує. Ідея натхненна системою торгових точок (POS) та іншими електронними системами транзакцій: вводиться команда, функція безперешкодно виконується, і обмін завершується. У дусі всього, що означає криптовалюта, смарт-контракти пропонують децентралізований спосіб запису транзакцій, який виключає будь-якого посередника. Дві сторони укладають угоду, і смарт-контракт запускає код, який поміщає цю транзакцію в блокчейн. Уявіть собі весь цей процес як торговий автомат. Блокчейн - це власне торговий автомат, в якому зберігається все, що вам може знадобитися, наприклад, фінансова звітність або пачка чіпсів Doritos, а продукти виступають в якості другої сторони, з якою ви ведете бізнес, і ваша криптовалюта (в даному випадку ETH) виступає в якості оплати. Смарт-контракт - це код, який виконується для того, щоб повідомити машині, що транзакція була здійснена, повинна бути записана, а продукт повинен бути виданий. Смарт-контракти автоматично виконують код, вбудований в умови угоди, і саме це допомагає скоротити час транзакцій в блокчейні. Інша аналогія - уявити смарт-контракти як світлофори на перехресті. Без них машинам довелося б спілкуватися з усіма іншими на перехресті, щоб знати, хто і коли їде. Це працює, але неймовірно повільно. Натомість, будь-яка команда, що надходить на світлофор, диктує потік транспорту так, що все це, в теорії, відбувається безперебійно. Уявіть собі, що кожен обмін криптовалютою потрібно було б схвалювати вручну. Це надзвичайно уповільнило б трафік у блокчейні, що вплинуло б на роботу всіх інших додатків. Смарт-контракти також визначають, за яких умов гроші можуть переходити з рук в руки, наприклад, коли, скільки і кому. Знову ж таки, банк або інший посередник зазвичай диктує ці умови, і гроші повинні спочатку пройти через них. Але завдяки смарт-контрактам це не так. Це також означає, що вам не потрібно буде проходити через сторонні додатки, такі як PayPal або Venmo, щоб обміняти свої гроші, тому що ви можете обміняти їх безпосередньо з тим, з ким захочете. Звучить знайомо? Так, децентралізовані обміни можливі завдяки смарт-контрактам. Вони допомагають вам купувати і продавати криптовалюту напряму, а умови транзакції кодуються, затверджуються і виконуються контрактом, який

потім надсилається і записується в блокчейн.

Користувачі блокчейну зберігають копію кожного смарт-контракту, щоб записи були точними, і вони можуть переконатися, що користувачі, які беруть участь у транзакції, дотримуються правил. Існує така гармонійна система між смарт-контрактами та блокчейном, яка постійно вдосконалюється та оновлюється, і люди починають використовувати смарт-контракти не лише для обміну грошима, але й для автоматизації певних частин своїх додатків. Зрештою, якщо смарт-контракти можуть запускати автомати з продажу криптовалюти, то які ще автомати вони можуть запускати? Ефіріум має глибокий зв'язок зі смарт-контрактами, оскільки розробка цих контрактів була одним з перших конкретних застосувань Ефіріуму. Це ще раз показує, як Ethereum взяв існуючі інструменти, надані для криптовалют, і перевірив, наскільки багато можна з ними зробити. Ethereum взяв, здавалося б, обмежувальну мову програмування, яку біткоїн використовував для смарт-контрактів, і замінив її на мову, яка дозволяє проводити ширші обчислення, що дало можливість розробникам використовувати смарт-контракти поза межами криптовалют. Завдяки цій зміні мови смарт-контракти через Ethereum можуть зберігати незмінну інформацію з додатків, таку як реєстрації та записи про членство, дозволяти більш ніж двом сторонам домовлятися про розподіл коштів, допомагати іншим смарт-контрактам у роботі, схожій на їхнє кодування, а також керувати записами та угодами між користувачами, які можуть бути витягнуті за потреби.

Як і все, що пов'язано з Ethereum, смарт-контракти демонструють значний потенціал. Насправді, спрощена і постійно вдосконалювана мова цих контрактів - це те, що в даний час допомагає DeFi стати більш потужною рушійною силою змін. Час покаже, як розвиватимуться його застосування і якою буде мова, на якій він писатиметься.

2.7 Мережа Ethereum

Ethereum - це децентралізована платформа, яка дозволяє розгортати на ній

DApps. Смарт-контракти пишуться за допомогою мови програмування solidity.

контракти пишуться за допомогою мови програмування Solidity. DApps створюються з використанням одного або декількох смарт-контрактів. Смарт-контракти - це програми, які працюють точно так, як запрограмовано без будь-якої можливості простою, цензури, шахрайства або стороннього інтерфейсу. В Ethereum смарт-контракти можуть бути написані кількома мовами програмування, зокрема Solidity, LLL та Serpent. Solidity - найпопулярніша з цих мов. Ethereum має внутрішню валюту, яка називається ефір. Щоб розгорнути смарт-контракти або викликати їх методи, нам потрібен ефір. Як і будь-який інший DApp, смарт-контракт може мати декілька екземплярів, і кожен екземпляр ідентифікується своєю унікальною адресою. Як облікові записи користувачів, так і смарт-контракти контракти можуть зберігати ефір. Ethereum використовує структуру даних блокчейну і протокол консенсусу з доказом роботи. Метод смарт-контракту може бути викликаний через транзакцію або іншим способом. У мережі існує два типи вузлів у мережі: звичайні вузли та майнери. Звичайні вузли - це ті, які просто мають копію блокчейну, в той час як майнери створюють блокчейн, видобуваючи блоки.

Ефіріум часто називають "світовим комп'ютером". Давайте почнемо з опису, орієнтованого на комп'ютерні науки, а потім спробуємо розшифрувати його за допомогою більш практичного аналізу можливостей і характеристик Ефіріуму, порівнюючи його з біткоіном та іншими платформами децентралізованого обміну інформацією.[7]

З точки зору комп'ютерних наук, Ethereum - це детермінований, але практично необмежений автомат, що складається з глобально доступного синглетного стану і віртуальної машини, яка застосовує зміни до цього стану.

З більш практичної точки зору, Ethereum - це глобально децентралізована обчислювальна інфраструктура з відкритим вихідним кодом, яка виконує програми, що називаються смарт-контрактами. Вона використовує блокчейн для синхронізації та зберігання змін стану системи, а також криптовалюту під назвою ефір для вимірювання та обмеження витрат ресурсів на виконання.

Платформа Ethereum дозволяє розробникам створювати потужні

децентралізовані додатки з вбудованими економічними функціями. Забезпечуючи високу доступність, можливість аудиту, прозорість і нейтральність, вона також зменшує або усуває цензуру і знижує певні ризики контрагентів.

Багато людей прийдуть до Ефіріуму з певним попереднім досвідом роботи з криптовалютами, зокрема з біткоїном. Ethereum має багато спільних елементів з іншими відкритими блокчейнами: однорангова мережа, що з'єднує учасників, візантійський відмовостійкий алгоритм консенсусу для синхронізації оновлень стану (блокчейн з доказом роботи), використання криптографічних примітивів, таких як цифрові підписи та хеші, і цифрова валюта (ефір).

Проте багато в чому і мета, і побудова Ефіріуму разюче відрізняються від відкритих блокчейнів, які йому передували, в тому числі і від Біткоїна.

Мета Ефіріуму не полягає в тому, щоб бути платіжною мережею цифрової валюти. Хоча цифрова валюта ефір є невід'ємною частиною і необхідною для роботи Ethereum, ефір призначений як утилітарна валюта для оплати за використання платформи Ethereum як світового комп'ютера.

На відміну від Біткоїна, який має дуже обмежену мову сценаріїв, Ефіріум розроблений як універсальний програмований блокчейн, який працює на віртуальній машині, здатний виконувати код довільної та необмеженої складності. В той час як мова сценаріїв Біткоїна навмисно обмежена простою оцінкою істинності/хибності умов витрачання коштів, мова Ефіріуму є повною мовою Тьюринга, що означає, що Ефіріум може прямо функціонувати як комп'ютер загального призначення.

Оригінальний блокчейн, а саме блокчейн Біткоїна, відстежує стан одиниць біткоїна та права власності на них. Ви можете уявити собі біткоїн як розподілений консенсусний автомат, де транзакції викликають глобальну зміну стану, що змінює право власності на монети. Переходи станів обмежені правилами консенсусу, що дозволяє всім учасникам (в кінцевому підсумку) прийти до спільного (консенсусного) стану системи після того, як буде видобуто кілька блоків.

Ethereum також є розподіленою машиною стану. Але замість того, щоб відстежувати лише стан володіння валютою, Ethereum відстежує переходи станів

сховища даних загального призначення, тобто сховища, яке може зберігати будь-які дані, виражені у вигляді кортежу ключ-значення. Сховище даних "ключ-значення" зберігає довільні значення, кожне з яких пов'язане з певним ключем; наприклад, значення "Освоєння Ethereum" пов'язане з ключем "Назва книги". В деякому сенсі, це служить тій же меті, що і модель зберігання даних в оперативній пам'яті (RAM), яка використовується в більшості комп'ютерів загального призначення. Ethereum має пам'ять, яка зберігає як код, так і дані, і використовує блокчейн Ethereum, щоб відстежувати, як ця пам'ять змінюється з часом. Як і звичайний комп'ютер із запам'ятовуючим пристроєм, Ethereum може завантажувати код у свою машину станів і запускати його, зберігаючи отримані зміни стану в своєму блокчейні. Дві найважливіші відмінності від більшості комп'ютерів загального призначення полягають в тому, що зміни стану Ethereum регулюються правилами консенсусу, а сам стан розподіляється глобально (Рисунок 2.10).

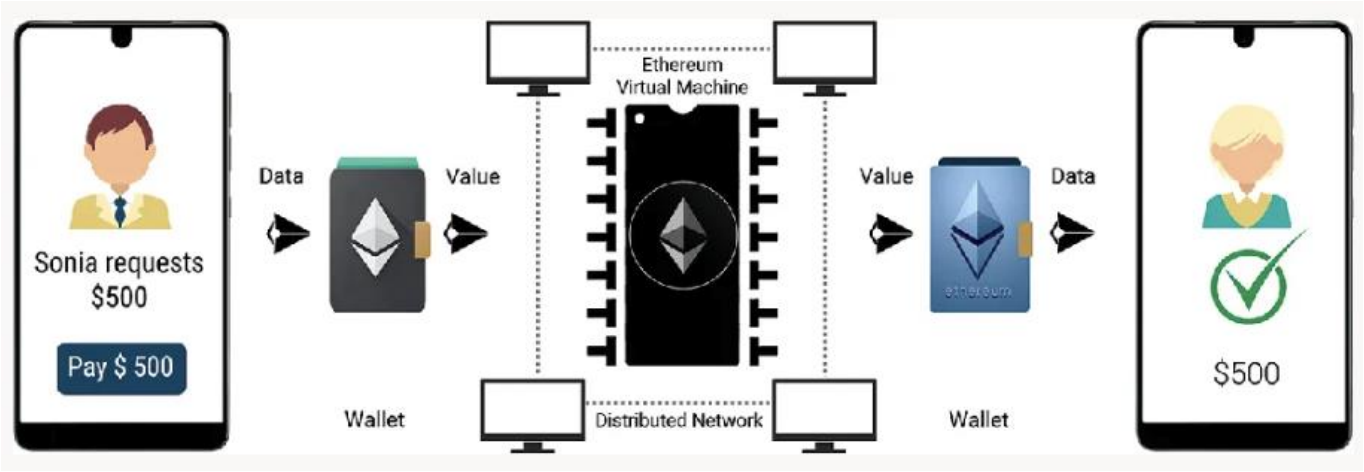


Рисунок 2.10 - Схема роботи мережі Ethereum

Всі великі інновації вирішують реальні проблеми, і Ethereum не є винятком. Ethereum був задуманий в той час, коли люди визнали силу моделі біткоїна і намагалися вийти за рамки криптовалютних додатків. Але розробники зіткнулися з головоломкою: їм потрібно було або будувати на основі Біткоїна, або створювати

новий блокчейн. Розбудова на основі Біткоїна означала життя в рамках навмисних обмежень мережі та пошук обхідних шляхів. Обмежений набір типів транзакцій, типів даних і розмірів сховищ даних, здавалося, обмежував типи додатків, які могли б працювати безпосередньо на Біткоїні; все інше потребувало додаткових поза мережевих рівнів, а це одразу ж зводило нанівець багато переваг використання публічного блокчейну. Для проектів, які потребували більшої свободи та гнучкості, залишаючись при цьому в ланцюжку, єдиним варіантом був новий блокчейн. Але це означало багато роботи: перезавантаження всіх елементів інфраструктури, вичерпне тестування тощо.

Наприкінці 2013 року Віталій Бутерін, молодий програміст та ентузіаст біткоїна, почав думати про подальше розширення можливостей біткоїна та Mastercoin (протокол накладення, який розширив можливості біткоїна, пропонуючи елементарні смарт-контракти). У жовтні того ж року Віталік запропонував команді Mastercoin більш узагальнений підхід, який дозволив би гнучким і скриптовим (але не повним за Тьюрінгом) контрактам замінити спеціалізовану мову контрактів Mastercoin. Хоча команда Mastercoin була вражена, ця пропозиція була занадто радикальною зміною, щоб вписатися в їх дорожню карту розвитку.

У грудні 2013 року Віталій почав ділитися технічним документом, в якому була викладена ідея Ethereum: повного за Тьюрінгом блокчейну загального призначення. Кілька десятків людей побачили цей ранній проект і надали свої відгуки, допомагаючи Віталіку розвивати пропозицію.

Обидва автори цієї книги отримали ранню версію білої книги і прокоментували її. Андреас М. Антонопулос був заінтригований ідеєю і поставив Віталіку багато запитань про використання окремого блокчейну для забезпечення дотримання правил консенсусу при виконанні смарт-контрактів, а також про наслідки використання мови, повної за Тьюрінгом. Андреас продовжував з великим інтересом стежити за розвитком Ethereum, але перебував на ранніх стадіях написання своєї книги "Освоєння біткоїна" і не брав безпосередньої участі в роботі над Ethereum до набагато пізнішого часу. Доктор Гевін Вуд був одним з перших,

хто звернувся до Віталіка і запропонував допомогти з його навичками програмування на C++. Гевін став співзасновником, розробником та технічним директором Ethereum.

2.8 Особливості децентралізованих додатків

Децентралізовані додатки, також відомі як децентралізовані додатки або DApps (зазвичай вимовляються як де-apps), є частиною нової хвилі веб-додатків, покликаних підвищити прозорість комерційних транзакцій, державних процесів, ланцюжків поставок і всіх тих систем, які наразі вимагають взаємної довіри між замовником і постачальником, користувачем і провайдером. Мета DApps - мінімізувати або усунути необхідність будь-якої довіри між учасниками системної взаємодії, щоб розширити можливості користувачів за межами того, що надала Веб 2.0. Дехто стверджує, що DApps можуть стати основою Веб 3.0.

Централізований додаток або система контролюється єдиним або центральним суб'єктом: особою, компанією, установою, державним органом тощо. Суб'єкт розміщує систему безпосередньо на своїй території або через постачальника послуг чи хмарного сервісу і має повний контроль над усіма компонентами та рівнями архітектури системи. Користувач довіряє добросовісності центрального органу і вирішує, чи отримувати доступ до його системи, залежно від його репутації. З точки зору користувача, системі або довіряють, або ні. Саме так сьогодні влаштовано більшість веб-додатків та корпоративних додатків. Ви не повинні знайти в цьому нічого дивного. Централізований додаток тісно пов'язаний з єдиним суб'єктом, який його контролює. Отже, користувачі вирішують, чи отримувати до нього доступ, залежно від того, наскільки вони йому довіряють. Перейдемо до децентралізованих додатків. Якщо ви на мить замислитесь над альтернативним додатком для електронної комерції, який я представив раніше, ви погодитесь, що він має переваги перед SmallWebRetailer.com: Вигідні умови транзакції - транзакція буде завершена і гроші будуть повністю перераховані продавцю тільки тоді, коли він

виконає всі умови, пов'язані з транзакцією, наприклад, отримає ваше підтвердження безпечної доставки. Це усунуло б одне з найбільших застережень щодо SmallWebRetailer.com: невпевненість у тому, чи отримаєте ви доставку, і що станеться з вашими грошима, якщо ні. Незалежне виконання та перевірка транзакцій - транзакція буде оброблятися не роздрібним продавцем або окремою третьою стороною, а одним з багатьох учасників платформи, що підтримує додаток для електронної комерції, а потім всі учасники платформи незалежно перевірятимуть її. Механізм, за допомогою якого всі сторони домовляються про перевірку транзакції, називається консенсусом (визначення наведено в інформаційному листі). Механізм консенсусу гарантує, що обіцяні умови транзакції будуть виконані і перевірені багатьма незалежними сторонами, а не невідомим продавцем.

Визначення Консенсус - це розподілена і ненадійна форма угоди про перевірку транзакції. Розподілена означає, що незалежний центральний орган не здійснює перевірку транзакції; натомість усі сторони беруть участь у її перевірці та погоджуються на неї. Без довіри означає, що сторонам не потрібно довіряти одна одній, щоб домовитися про результати перевірки. Консенсус досягається, коли кваліфікована більшість учасників погоджується з результатом транзакції. Прозорість - ви зможете перевірити код, який обробляє транзакцію, і переконатися, що він дотримується визначених умов, перш ніж переказати свої гроші продавцю. Це дасть вам додатковий рівень впевненості в тому, що додаток виконується відповідно до обіцяних умов. Ви можете виконати всі ці вимоги, побудувавши альтернативний додаток для електронної комерції як мережу рівнозначних за важливістю та функціональністю обробних вузлів, кожен з яких належав би окремій стороні. Кожен вузол зможе обробляти транзакції так само, як це роблять інші вузли, перевіряти всі транзакції так само, як інші вузли роблять рівний внесок у результат транзакції. Наслідком такої архітектури буде те, що обробка буде децентралізована до мережі незалежних вузлів, а не централізована на певному наборі серверів, якими володіє конкретний суб'єкт господарювання. Така децентралізація звільнила б користувача від необхідності довіряти конкретному

суб'єкту: користувач мав би довіряти лише структурі мережі в цілому. Додатки, побудовані на такій архітектурі, відомі як децентралізовані додатки.

DApps, які також називають децентралізованими додатками, - це цифрові додатки, побудовані на блокчейні. Вони поєднують графічний інтерфейс користувача (GUI) зі смарт-контрактами (комп'ютерною програмою). Основна відмінність між додатками і DApps полягає в тому, що DApps працюють на децентралізованих мережах блокчейнів P2P. І навпаки, додаток працює на одному або декількох централізованих серверах. Крім того, DApps не контролюються жодною третьою стороною або органом влади. Вони не мають єдиної точки відмови, тому вважаються дуже безпечними, оскільки всі дані зберігаються на блокчейні, який є незмінним.

Спочатку DApps з'явилися на Ethereum. DApps, розроблені на основі Ethereum, не потрібно створювати з нуля. Замість цього, розробники DApp можуть побудувати його на основі існуючого блокчейну. Однак кастомні розробники DApp створюють DApp на новому блокчейні. Серед популярних DApps - Uniswap, Yearn Finance, Opensea, Curve і PancakeSwap (Рисунок 2.11).

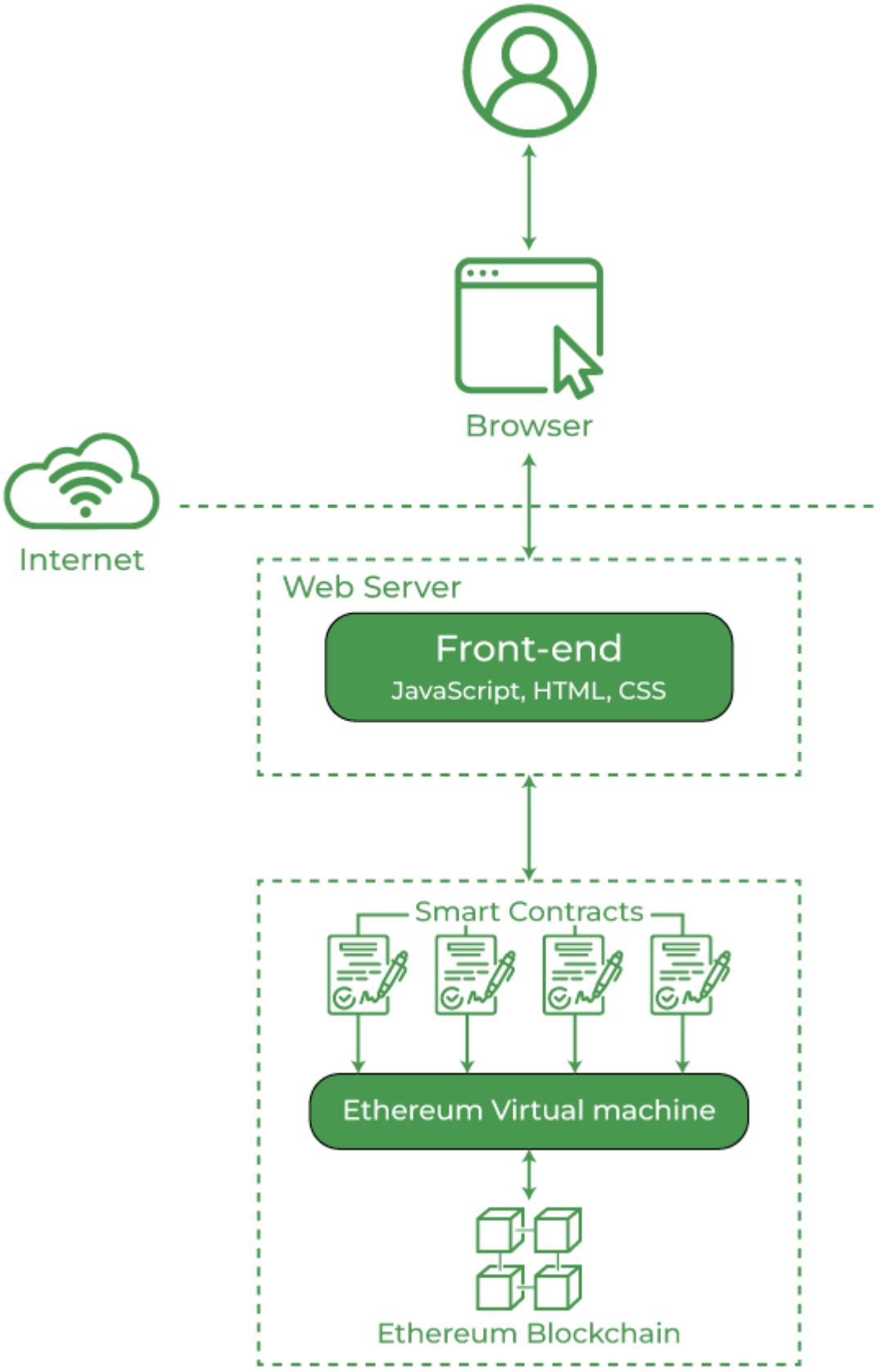


Рисунок 2.11 - Архітектура децентралізованих додатків

З поширенням смарт-контрактів DApps стають все більш поширеними. Кожна галузь, будь то ланцюжок поставок, охорона здоров'я, нерухомість або фінанси, з нетерпінням чекає на розробку DApp для своїх індивідуальних організаційних потреб і для того, щоб залишатися конкурентоспроможними на дуже щільному ринку.

Давайте розглянемо деякі ключові відмінності між звичайним додатком і DApp.

Найбільша різниця між додатком і DApp полягає в тому, що DApp децентралізований, а додаток - централізований. Додатки контролюються розробниками, командами підтримки, центральними органами влади або будь-якою третьою стороною. З іншого боку, DApp працює за моделлю одноранговогo блокчейну, де немає центрального органу, який би здійснював моніторинг або контроль за діяльністю. Отже, DApp не має єдиної точки відмови, в той час як додаток має.

DApps надійні і вважаються надійними серед усіх учасників блокчейну. Причина в тому, що DApp надає рівне право власності та контролю кожному учаснику.

Як ми всі знаємо, додатки на основі блокчейну є більш безпечними, ніж інші. Аналогічно, DApps пропонують першокласний захист від несанкціонованого доступу та різних спроб кібератак. Оскільки DApps децентралізовані, а записи незмінні, зламати мережу блокчейну майже неможливо. Однак звичайні додатки мають певні лазівки і потребують додаткових заходів для захисту від злому або потенційних кібератак.

Додаток має різні потреби та витрати на розробку. Наприклад, вартість окремих хмарних серверів, сторонніх інструментів та обслуговування. Проте, розробка DApp є досить економічно вигідною.

Мобільні та десктопні додатки здебільшого безкоштовні. Лише деякі додатки стягують плату за додаткові функції або ресурси. З іншого боку, DApp стягує невелику плату щоразу, коли користувачі використовують додаток. Плата залежить від масштабованості та складності додатку для децентралізації.

Якщо порівнювати швидкість роботи DApps і звичайних додатків, то DApps працюють повільніше. За словами розробників, DApps можуть обробляти лише 15 транзакцій на секунду. Однак у найближчому майбутньому DApps можна буде масштабувати, а отже, проблема швидкості буде вирішена. Звичайні додатки досить швидкі і мають високу швидкість обробки.

Переваги розробки DApps:

- Стійкість до цензури - оскільки жодна третя сторона або окремий суб'єкт не контролює DApp, авторизований доступ гарантує, що жодна організація або уряд не зможуть змінити транзакції в блокчейні
- Безпека - після розгортання смарт-контракту послуга стає доступною для учасників блокчейну. Жодна зловмисна організація не зможе здійснити кібератаку на окремі DApps
- Відкритий код - DApps мають відкритий код, що дозволяє швидше розвивати екосистему
- Цілісність - DApps є незмінними завдяки надійному криптографічному захисту
- Конфіденційність - DApps не потребують вашої реальної ідентичності для взаємодії, тому конфіденційність залишається недоторканою
- Сумісність - DApps сумісні з криптоплатформами і можуть приймати криптовалюти в якості платежів
- Універсальність - такі платформи, як Ethereum, дають можливість розробникам зосередитися на інноваційних ідеях. DApps можна легко розгортати в різних галузях, трансформуючи реальні додатки в іграх, соціальних мережах, фінансах, торгівлі та електронній комерції.

DApps мають потенціал для самодостатності. У найближчі роки ми можемо очікувати на появу більшої кількості стейблкоінів, що використовують передові методи цінової стабільності криптовалют. Очікується, що провідні компанії-розробники DApp розроблятимуть більш складні додатки на основі стейблкоінів.

Децентралізовані біржі (DEX) також будуть розвиватися повним ходом у майбутньому. Служби розробки DApp створюватимуть більше DEX, щоб вирішити

проблему сумісності різних технологій блокчейну.

Аналогічно, ще одним потенційним майбутнім варіантом використання DApps буде криптокредитування. Послуги з розробки DeFi DApp революціонізують процес кредитування для позичальників, яким доводилося проходити через традиційні і громіздкі процеси подачі заявок в декількох фінансових установах. Коротше кажучи, у найближчому майбутньому ми можемо побачити безмежні можливості для DApps, свідками яких ми можемо стати.

DApps у фінансовому світі здаються очевидною річчю, але насправді вони можуть бути інноваційними в усіх галузях. Давайте розглянемо деякі з цих переваг у таких галузях, як фінанси, соціальні мережі, ігри тощо.

Лихварі та позичальники можуть використовувати DApps для ведення свого бізнесу. У банках кредитори заробляють певні відсоткові ставки на основі заощаджених грошей. Чим більше людина заощаджує, тим більше банк може позичити, і тим більше обидві сторони заробляють на відсотках. Однак банк, який діє як централізована установа, бере більший відсоток, ніж хотілося б кредиторам, просто за те, що надає місце для зберігання коштів.

На DApp кредитори отримують 100% своїх відсотків, оскільки немає посередника, якому потрібно платити. Крім того, вони мають більше контролю над позиками, заробляючи токени на платформі, яку вони обрали для кредитування.

Що стосується позичальників, то вони мають більше можливостей впливати на розмір відсотків, а також на час їх сплати. Дійсно, деякі платформи дозволяють позичальникам виплачувати відсотки місяцями або навіть роками, за умови, що вони відповідають мінімальному порогу виплат. Позичальник також може обговорити ставки з кредитором, забезпечуючи справедливе рішення для обох сторін.

Коли все сказано і зроблено, кошти можуть бути отримані негайно завдяки технології смарт-контрактів. Немає необхідності залучати юристів та інших третіх осіб, що робить процес підтвердження довшим і дорожчим для обох сторін.

Користувачі отримують значну вигоду від DApps для соціальних мереж. По-перше, ніхто не може цензурувати пости, що означає свободу слова. Однак, якщо

деякі дописи стають проблемою, спільнота може проголосувати за їх видалення.

Інфлюенсери також можуть заробляти більше. На традиційних платформах, таких як Twitter, компанія отримує найбільше прибутку від популярних твітів. Вона отримує дохід від реклами з усіх відвідувань сайту, а автор не отримує нічого в грошовому еквіваленті.

DApps для соціальних мереж можуть мати вбудовану систему чайових, що використовує токен, і користувачі можуть розміщувати рекламу і отримувати повну оплату, а не відрахування компанії.

Ігри завжди були цікавим варіантом використання DApp. Наразі ігри вимагають десятки годин, вкладених у розвиток персонажа, в якого, ймовірно, вкладено реальні гроші, лише для того, щоб він сидів, коли гравець рухається далі.

DApps представляють більш цікаве рішення з точки зору цінності. Візьмемо, наприклад, таку гру, як CryptoKitties. Гравці купують токенізований актив, в даному випадку - кота. Згодом цей кіт росте з часом, збільшуючи свою вартість, якщо його правильно вирощувати. Потім користувач може продати цього кота за будь-яку суму, якщо знайдеться покупець, який заплатить за нього.

Крім того, деякі коти можуть потенційно розмножуватися з іншими котами, створюючи ще більш рідкісних, потенційно більш цінних котів. Гравці можуть торгувати або колекціонувати котів, робити з ними все, що їм заманеться. Їхні часові інвестиції стають по-справжньому цінними. Зараз таких ігор небагато, але уявіть собі цю концепцію в більш розвиненій грі з багатогодинним ігровим процесом. Можливо, повноцінні ігри - це наше майбутнє.

У більшості випадків голосування є болісним процесом. Воно часто включає в себе різні етапи перевірки - деякі з них недоступні для громадян без належного житла або тих, хто страждає від інших проблем. Не кажучи вже про фальсифікації та подібні незаконні дії.

Голосуючий DApp може зробити цю процедуру доступною для всіх завдяки смарт-контрактам. По суті, спільнота може проголосувати за список пропозицій. Потім вони можуть встановити часовий проміжок, скажімо, 24 години, протягом якого користувачі можуть "заставити" свій голос токенами. Це відкриває участь для

всіх, дозволяючи будь-кому голосувати анонімно.

Голоси зберігаються в децентралізованій мережі, що робить їх незмінними і непідробними. Крім того, смарт-контракти можуть винагороджувати виборців відповідним токеном за їхні зусилля, стимулюючи голосувати більше людей, ніж будь-коли раніше.

Багато користувачів користуються блокуванням реклами під час перегляду сайтів в Інтернеті. Очевидно, що це є проблемою для веб-сайтів, які намагаються генерувати дохід, але в деякому сенсі зрозумілою, оскільки реклама стала досить неприємною у багатьох відношеннях. DApp для браузера може це виправити.

Коли користувачі переглядають веб-сторінки, вони роблять це за допомогою інтегрованого в браузер блокувальника реклами та трекерів, заробляючи при цьому криптовалюту. Тепер, коли користувачі знаходять авторів і веб-сайти, які вони хотіли б підтримати, вони можуть дозволити їм робити внески. Це означає, що чим довше користувач переглядає сайт, тим більше він платить цьому сайту з часом. Користувачі можуть навіть увімкнути рекламу для цих сайтів, що допоможе їм у довгостроковій перспективі.

2.9 Інструменти для розробки децентралізованих додатків

Для розробки проекту децентралізованого додатку файлообмінника, були використані:

- Solidity
- React.js
- Ether.js та Web3.js
- Metamask

Solidity - це абсолютно нова мова програмування, розроблена Ethereum, другим за капіталізацією криптовалютним ринком. Solidity - це об'єктно-орієнтована мова програмування, створена спеціально командою Ethereum Network для побудови та дизайну смарт-контрактів на платформах Blockchain.

Вона використовується для створення смарт-контрактів, які реалізують

бізнес-логіку і генерують ланцюжок записів транзакцій в системі блокчейн.

Це інструмент для створення машинного коду та його компіляції на віртуальній машині Ethereum (EVM).

Вона має багато спільного з C та C++ і є досить простою у вивченні та розумінні. Наприклад, "main" в C еквівалентний "контракту" в Solidity.

Як і в інших мовах програмування, у Solidity також є змінні, функції, класи, арифметичні операції, маніпуляції з рядками та багато інших понять.

Solidity - відносно нова мова, яка швидко розвивається.

Наразі Solidity є основною мовою в Ethereum та інших приватних блокчейнах, що працюють на конкуруючих платформах, таких як Monax та його блокчейн Hyperledger Burrow, який використовує Tendermint для досягнення консенсусу.[8]

SWIFT створив перевірку концепції, яка працює на Burrow і використовує Solidity.Віртуальна машина Ethereum (EVM) забезпечує середовище виконання для смарт-контрактів Ethereum.

Вона в першу чергу спрямована на забезпечення безпеки та виконання ненадійних програм за допомогою міжнародної мережі публічних вузлів.

EVM спеціалізується на запобіганні атакам типу "відмова в обслуговуванні" і засвідчує, що програми не мають доступу до стану одна одної, а також встановлює зв'язок без будь-якого можливого втручання.

Смарт-контракти - це високорівневі програмні коди, які компілюються в EVM перед тим, як вони публікуються в блокчейні Ethereum для виконання.

Це дозволяє здійснювати надійні транзакції без участі третьої сторони; ці транзакції можна відстежити і вони незворотні. Мови програмування, які зазвичай використовуються для створення та написання смарт-контрактів: Serpent, Solidity, Mutan та LLL.

Типи даних для програмування на Solidity:

- Булевий - Булевий тип даних повертає "1", коли умова істинна, і "0", коли вона хибна, в залежності від статусу умови.
- Ціле число - у Solidity ви можете додавати та видаляти знаки цілих чисел.

Вона також підтримує винятки часу виконання та ключові слова uint8 і uint256.

- Рядок - Одинарні або подвійні лапки можуть позначати рядок.
- Модифікатор - перед виконанням коду смарт-контракту модифікатор часто перевіряє, чи є будь-яка умова раціональною.
- Масив - синтаксис програмування Solidity подібний до синтаксису інших мов ООП і підтримує як одновимірні, так і багатовимірні масиви.

Крім того, програмування на Solidity дозволяє "мапувати" структури даних за допомогою перелічень, операторів та хеш-значень для повернення значень, що зберігаються в певних місцях зберігання.

Переваги Solidity для створення DApps:

- Інтеграція з екосистемою Ethereum: Solidity розроблений спеціально для Ethereum, що робить його добре придатним для розробки додатків, які працюють на блокчейні Ethereum. Він легко інтегрується з екосистемою інструментів, бібліотек та інфраструктури Ethereum.
- Підтримка смарт-контрактів: Solidity в першу чергу використовується для написання смарт-контрактів, які є самодостатніми контрактами, де умови угоди безпосередньо вписані в код. Це дозволяє розробникам створювати децентралізовані та надійні додатки, використовуючи можливості смарт-контрактів.
- Спільнота та документація: Solidity має велику і активну спільноту розробників. Підтримка спільноти виражається у великій документації, навчальних посібниках та форумах, де розробники можуть звертатися за допомогою, обмінюватися знаннями та співпрацювати над проектами.
- Функції безпеки: Solidity має функції, призначені для підвищення безпеки смарт-контрактів. Наприклад, він включає такі функції, як модифікатори контролю доступу, і дозволяє розробникам впроваджувати кращі практики для мінімізації вразливостей, такі як захист від повторного входу і належна перевірка вхідних даних.
- Сумісність зі стандартами ERC: Solidity сумісний з різними стандартами

Ethereum Request for Comments (ERC), такими як ERC-20 (взаємозамінні токени) і ERC-721 (не взаємозамінні токени). Це полегшує створення токенів і забезпечує сумісність з іншими проектами, які дотримуються цих стандартів.

- Активний розвиток: Solidity активно розвивається і вдосконалюється. Регулярно випускаються оновлення і нові функції для вирішення проблем, підвищення безпеки і розширення функціональності. Це гарантує, що розробники мають доступ до найновіших інструментів та вдосконалень.

Мінуси Solidity для створення додатків:

- Крива навчання: Solidity має криву навчання, особливо для розробників, які є новачками в розробці блокчейну. Розуміння таких понять, як газ, транзакції та децентралізована природа DApps може бути складним для початківців.
- Ризики безпеки: Написання безпечних смарт-контрактів вимагає ретельного підходу, і навіть досвідчені розробники можуть допустити вразливості. Поширені ризики безпеки включають атаки на повторний вхід, цілочисельне переповнення/недоповнення та некоректний контроль доступу.
- Проблеми незмінності: Після розгортання смарт-контракти на блокчейні Ethereum є незмінними, тобто їх неможливо змінити. Це може стати проблемою, якщо після розгортання будуть виявлені помилки або проблеми з безпекою, що вимагає ретельного планування на етапі розробки.
- Проблеми з масштабуванням: Поточні обмеження масштабованості Ethereum можуть вплинути на продуктивність Dapp, особливо в періоди високої мережевої активності. Це може призвести до збільшення вартості транзакцій і уповільнення часу підтвердження.
- Залежність від Ethereum: Solidity спеціально розроблена для Ethereum, а це означає, що якщо Dapp побудований на Ethereum, він залежить від успішності та масштабованості мережі Ethereum.
- Розвиток технології: Простір блокчейну швидко розвивається, і можуть з'явитися нові технології та мови програмування, які можуть кинути виклик або перевершити Solidity в майбутньому. Розробники повинні йти в ногу з

розвитком блокчейн-простору.

Таким чином, Solidity є потужним інструментом для розробки додатків на блокчейні Ethereum, але розробники повинні усвідомлювати його проблеми, особливо з точки зору безпеки та кривої навчання, пов'язаної з розробкою блокчейну. Постійні вдосконалення Solidity та екосистеми Ethereum спрямовані на те, щоб з часом вирішити деякі з цих проблем.

React.js - це популярна бібліотека JavaScript з відкритим вихідним кодом. Вона допомагає створювати вражаючі веб-додатки, які вимагають мінімальних зусиль та кодування. Основна мета ReactJS - розробка користувацьких інтерфейсів (UI), які покращують швидкість роботи додатків.[9]

Переваги React.js:

1. Легкість у вивченні та використанні. ReactJS набагато простіший у вивченні та використанні. Він поставляється з великою кількістю документації, підручників та навчальних ресурсів. Будь-який розробник, який має досвід роботи з JavaScript, може легко зрозуміти і почати створювати веб-додатки за допомогою React за кілька днів. Це частина V (view) в моделі MVC (Model-View-Controller), яку називають "одним з фреймворків JavaScript". Він не є повнофункціональним, але має перевагу у вигляді бібліотеки JavaScript User Interface (UI) з відкритим вихідним кодом, яка допомагає краще виконати завдання.

2. Створення динамічних веб-додатків стає простішим. Створити динамічний веб-додаток саме з HTML-рядків було складно, оскільки це вимагає складного кодування, але React JS вирішив цю проблему і робить його простішим. Він забезпечує менше кодування і надає більше функціональності. Він використовує JSX (розширення JavaScript), яке є особливим синтаксисом, що дозволяє використовувати HTML-цитати та синтаксис HTML-тегів для відображення певних підкомпонентів. Він також підтримує створення машинозчитуваних кодів.

3. Багаторазові компоненти. Веб-додаток на ReactJS складається з декількох компонентів, кожен з яких має власну логіку та елементи керування. Ці компоненти відповідають за виведення невеликого, багаторазового фрагменту HTML-коду, який можна використовувати повторно, де вам потрібно. Багаторазовий код

допомагає полегшити розробку та підтримку ваших додатків. Ці компоненти можуть бути вкладені в інші компоненти, що дозволяє створювати складні додатки з простих будівельних блоків. ReactJS використовує механізм віртуального DOM для заповнення даних в HTML DOM. Віртуальний DOM працює швидко, оскільки він змінює лише окремі елементи DOM замість того, щоб кожного разу перезавантажувати весь DOM.

4. Підвищення продуктивності. ReactJS покращує продуктивність завдяки віртуальному DOM. DOM - це крос-платформний API для програмування, який працює з HTML, XML або XHTML. Більшість розробників стикалися з проблемою, коли DOM оновлювався, що сповільнювало роботу додатку. ReactJS вирішив цю проблему, запровадивши віртуальний DOM. Віртуальний DOM React існує повністю в пам'яті і є представленням DOM веб-браузера. Завдяки цьому, коли ми пишемо React-компонент, ми не пишемо безпосередньо в DOM. Натомість ми пишемо віртуальні компоненти, які React перетворить на DOM, що призводить до більш плавної та швидкої роботи.

5. Підтримка зручних інструментів. React JS також набув популярності завдяки наявності зручного набору інструментів. Ці інструменти роблять завдання розробників зрозумілішими та простішими. React Developer Tools були розроблені як розширення для Chrome та Firefox і дозволяють переглядати ієрархію React-компонентів у віртуальному DOM. Це також дозволяє вибирати певні компоненти, переглядати та редагувати їхні поточні пропси та стан.

6. Дружній до пошукової оптимізації. Традиційні JavaScript фреймворки мають проблеми з пошуковою оптимізацією. Пошукові системи, як правило, мають проблеми з читанням JavaScript-додатків. Багато веб-розробників часто скаржилися на цю проблему. ReactJS долає цю проблему, що допомагає розробникам легко орієнтуватися в різних пошукових системах. Це відбувається тому, що React.js додатки можуть працювати на сервері, а віртуальний DOM буде рендеритися і повертатися в браузер як звичайна веб-сторінка.

7. Переваги використання бібліотеки JavaScript. Сьогодні ReactJS обирає більшість веб-розробників. Це тому, що він пропонує дуже багату бібліотеку

JavaScript. Бібліотека JavaScript забезпечує більшу гнучкість для веб-розробників у виборі способу, який вони хочуть.

8. Можливості для тестування коду. Додатки на ReactJS надзвичайно легко тестувати. Він пропонує простір, де розробник може тестувати та налагоджувати свій код за допомогою нативних інструментів.

Недоліки ReactJS:

1. Високий темп розробки. Високий темп розробки має як переваги, так і недоліки. Недоліком є те, що оскільки середовище постійно змінюється так швидко, деякі розробники не відчувають себе комфортно, регулярно вивчаючи нові способи виконання речей. Їм може бути важко прийняти всі ці зміни з урахуванням постійних оновлень. Їм потрібно постійно вдосконалювати свої навички та вивчати нові способи роботи.

2. Погана документація. Це ще один мінус, який характерний для технологій, що постійно оновлюються. React-технології оновлюються та прискорюються так швидко, що не вистачає часу на створення належної документації. Щоб подолати цю проблему, розробники пишуть інструкції самостійно з появою нових версій та інструментів у своїх поточних проектах.

3. Переглянути частину. ReactJS покриває лише рівні інтерфейсу користувача програми і нічого більше. Тому вам все одно потрібно вибрати деякі інші технології, щоб отримати повний набір інструментів для розробки в проекті.

4. JSX як бар'єр. ReactJS використовує JSX. Це синтаксичне розширення, яке дозволяє змішувати HTML з JavaScript. Такий підхід має свої переваги, але деякі члени спільноти розробників вважають JSX бар'єром, особливо для нових розробників. Розробники скаржаться на його складність у процесі навчання.

Web3.js та Ethers.js - популярні JavaScript-бібліотеки для розробки Ethereum, які дозволяють інтерфейсним додаткам взаємодіяти з блокчейном

Web3.js має більш широкий набір функцій, але працює повільніше, в той час як Ethers.js легша, швидша і має простіший API.

Вибір правильної бібліотеки залежить від аналізу вимог вашого проекту та оцінки сильних і слабких сторін кожної бібліотеки

Web3.js і Ethers.js - це дві бібліотеки, які відрізняються своїм синтаксисом, продуктивністю і підтримкою функцій, тому для розробників Ethereum важливо розуміти сильні і слабкі сторони кожної з них, перш ніж вибирати одну з них для свого проекту.

Хоча web3.js є більш усталеною і популярною бібліотекою JavaScript для Ethereum з більшою кількістю зірок GitHub, ethers.js набирає популярність як альтернатива web3.js. Ethers.js є більш модульною та надає чистіший та сучасніший API для розробників.[10]

Завдяки бібліотекам JavaScript, таким як Web3.js та Ethers.js, розробка Ethereum ще ніколи не була такою доступною. Ці інструменти забезпечують безперешкодну взаємодію з блокчейном Ethereum і мають вирішальне значення для створення децентралізованих додатків (DApps).

Якщо ви новачок у світі криптовалют або досвідчений розробник, який прагне дослідити різні варіанти, розуміння відмінностей між цими двома популярними бібліотеками може бути життєво важливим для вашого наступного проекту.

Якщо ви розробляєте для блокчейну Ethereum, швидше за все, ви стикалися з двома популярними бібліотеками JavaScript: Web3.js та Ethers.js. Ці бібліотеки Web3 дозволяють інтерфейсним додаткам взаємодіяти з блокчейном Ethereum, включаючи смарт-контракти. Але в чому різниця між цими бібліотеками і яку з них вибрати? У цій статті ми розглянемо ключові відмінності між Web3.js та Ethers.js щодо синтаксису, продуктивності, підтримки функцій тощо. Наприкінці цього керівництва ви будете краще розуміти, яка бібліотека найкраще підходить для ваших потреб як розробника Ethereum.

Web3.js - це відома бібліотека JavaScript, призначена для спрощення розробки Ethereum шляхом надання розробникам широкого набору інструментів та функцій.

По суті, вона дозволяє інтерфейсним додаткам безперешкодно взаємодіяти з блокчейном Ethereum - чи то розгортання смарт-контрактів, чи то запит деталей транзакцій.[11]

Як невід'ємна частина екосистеми Ethereum, Web3.js допоміг стимулювати розвиток децентралізованих додатків (DApps), дозволяючи розробникам втілювати свої ідеї в реальність на цій потужній платформі.

Наприклад, використовуючи Web3.js, ви можете пересилати Ефір між користувачами, не покладаючись на центральні сервери, або створювати повнофункціональні DApps, такі як CryptoKitties, які використовують технологію блокчейн для безпечних транзакцій і відстеження прав власності на цифрові активи.

Ethers.js - це популярна легка web3-бібліотека для розробки Ethereum, яка спрощує процес створення децентралізованих додатків і взаємодії зі смарт-контрактами.[10]

Вона пропонує простіший API, об'єктно-орієнтований синтаксис і переваги в продуктивності в порівнянні з Web3.js. Ethers.js також обробляє управління ключами інакше, ніж Web3.js, усуваючи необхідність ручного входу/виходу з системи.

Однією з найбільших переваг використання Ethers.js є його швидкий час виконання, оскільки він займає лише 77 КБ у стислому вигляді та 284 КБ у розпакованому, порівняно з більшим розміром Web3.js. Крім того, розробники вважають, що Ethers.js легше використовувати, оскільки він має меншу криву навчання і вимагає менше коду завдяки ефективній структурі дизайну.

Web3.js та Ethers.js мають помітні відмінності у синтаксисі та структурі коду. Ось деякі з них:

- Об'єктно-орієнтований vs функціональний синтаксис: Ethers.js має об'єктно-орієнтований синтаксис, що означає, що він використовує об'єкти для представлення структур даних і функції для виконання завдань. Web3.js, з іншого боку, використовує функціональний синтаксис, що означає, що він зосереджений на виконанні дій за допомогою ряду функцій.
- Відмінності в просторах імен: Ethers.js працює як єдиний простір імен, де доступ до всіх методів API здійснюється через об'єкт ethers. Для порівняння, Web3.js розділяє API на дві ролі - провайдера та підписувача.

- Відмінності в іменах методів: Імена методів, що використовуються в обох бібліотеках, відрізняються, тому розробникам необхідно звертатися до документації для правильного використання.
- Відмінності в абстракції контрактів: Обидві бібліотеки використовують різні методи абстракції контрактів: Web3.js використовує JSON RPC, а Ethers.js - Contract ABI.
- Відмінності в загальній складності: Як зазначалося, завдяки своїй об'єктно-орієнтованій природі та легкій архітектурі, розробники вважають використання Ethers.js менш складним, ніж Web3.js, при розробці DApps, враховуючи його простіший API та використання.

В цілому обидві бібліотеки дозволяють інтерфейсним додаткам ефективно взаємодіяти зі смарт-контрактами на основі блокчейну Ethereum, проте вони відрізняються за підходом до структурування коду та синтаксису.

Розробка Ethereum вимагає швидких і ефективних бібліотек для взаємодії з блокчейном. Ось деякі відмінності в продуктивності і швидкості між Web3.js і Ethers.js:

- Ethers.js є більш легкою бібліотекою, ніж Web3.js, що забезпечує кращу продуктивність.
- Ethers.js використовує менше ресурсів і споживає менше пам'яті, що робить її швидшою за Web3.js для деяких завдань.
- Ethers.js обробляє транзакції швидше, ніж Web3.js, завдяки спрощеному API та використанню.
- Web3.js може працювати повільніше, оскільки має більше функцій і можливостей, що може збільшити складність і сповільнити виконання.
- Ethers.js розроблений для використання з сучасними інструментами JavaScript, такими як TypeScript, що полегшує написання високоякісного коду, який добре працює.

Підсумовуючи, Ethers.js перевершує за продуктивністю завдяки своєму об'єктно-орієнтованому синтаксису, простішому API та зосередженню на легкій функціональності. Однак, хоча Web3.js може бути повільнішим у порівнянні з ним

через більший набір функцій, він залишається широко використовуваним завдяки сформованій спільноті розробників та доступним ресурсам. Зрештою, вибір між цими бібліотеками зводиться до конкретних потреб конкретного проекту. [11]

Metamask відіграє ключову роль в архітектурі децентралізованих додатків (DApps), які працюють на блокчейні Ethereum (Рисунок 2.12).

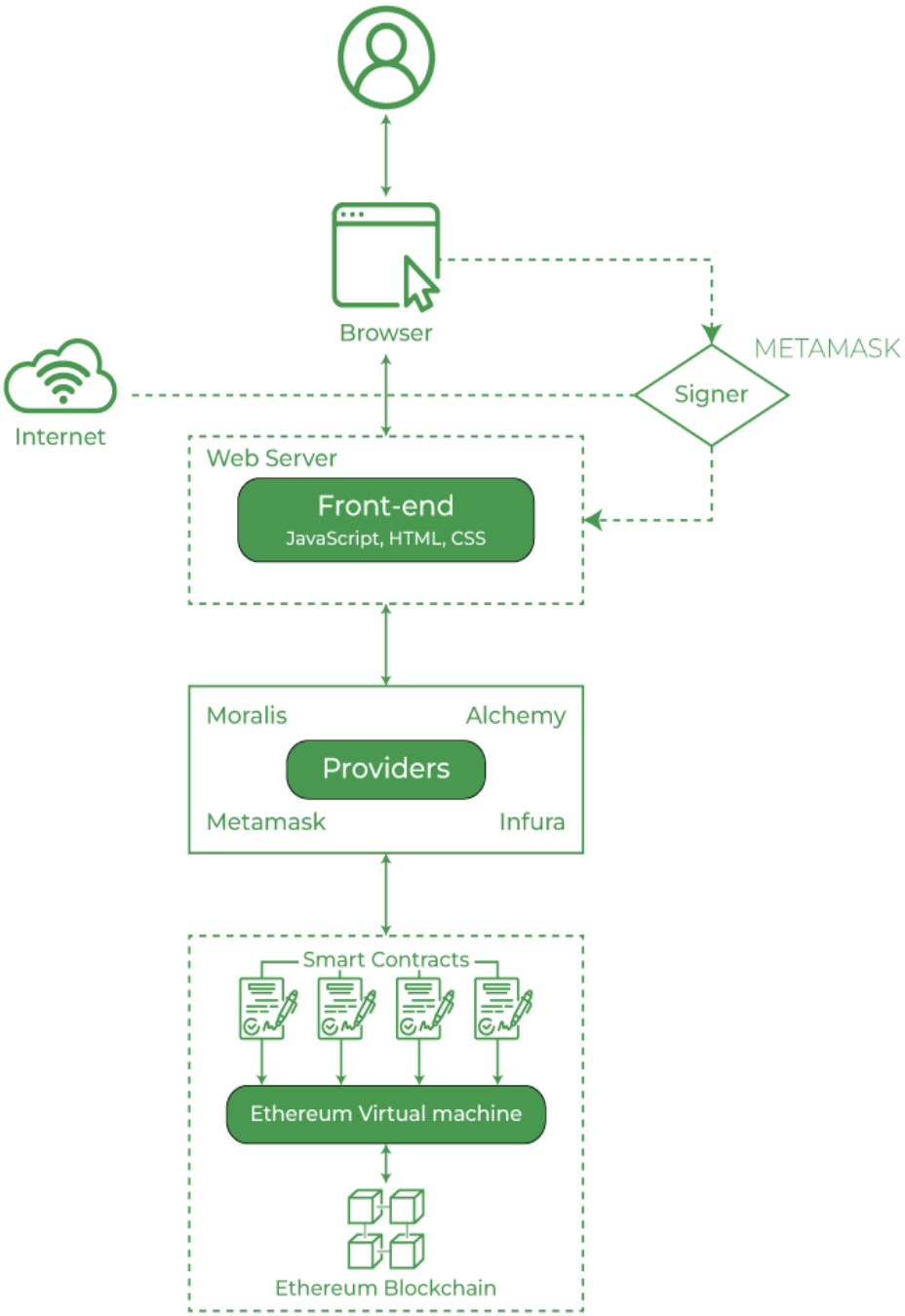


Рисунок 2.12 - Взаємодія Metamask та додатку

Як плагін для браузера та гаманець Ethereum, Metamask полегшує взаємодію між користувачами та мережею Ethereum. Ось огляд його ролі в архітектурі Dapp:

- Функціональність гаманця Ethereum: Metamask слугує гаманцем користувача Ethereum, що дозволяє йому безпечно зберігати та керувати своїми цифровими активами, включаючи Ефір (ETH) та різноманітні токени. Користувачі можуть переглядати баланс свого рахунку, історію транзакцій та керувати своїми активами безпосередньо через інтерфейс Metamask.
- Підключення до мережі Ethereum: Оскільки блокчейн Ethereum є децентралізованою мережею, потрібні спеціальні інструменти для підключення браузерів користувачів до блокчейну. Metamask виступає в ролі моста, надаючи розширення для браузера, яке встановлює з'єднання між браузером користувача і мережею Ethereum. Це з'єднання дозволяє користувачам безперешкодно взаємодіяти з децентралізованими додатками.
- Зберігання приватних ключів: Metamask зберігає приватні ключі користувача локально в його браузері. Закриті ключі необхідні для підписання транзакцій і підтвердження права власності на активи Ethereum. Безпечне зберігання приватних ключів Metamask гарантує, що користувачі мають контроль над своїми коштами під час взаємодії з DApps.
- Підписання транзакцій: Коли користувач ініціює транзакцію в Dapp (наприклад, відправляє Ефір або взаємодіє зі смарт-контрактом), Metamask пропонує користувачеві підписати транзакцію за допомогою свого приватного ключа. Цей підпис є криптографічним доказом того, що користувач авторизував транзакцію. Таким чином, Metamask виступає в ролі підписанта, сприяючи безпечному виконанню транзакцій в блокчейні Ethereum.
- Постачальник для DApps: Metamask не тільки зберігає приватні ключі, але й виступає в ролі провайдера для DApps. Він забезпечує необхідний зв'язок і комунікацію між Dapp і блокчейном Ethereum. Ця роль провайдера дозволяє DApps отримувати інформацію з блокчейну, таку як залишки на рахунках, статус транзакцій та інші важливі дані.

- Взаємодія користувача зі смарт-контрактами: Додатки часто передбачають взаємодію зі смарт-контрактами на блокчейні Ethereum. Metamask дозволяє користувачам взаємодіяти з цими смарт-контрактами, полегшуючи підписання транзакцій. Користувачі можуть запускати функції всередині смарт-контрактів, а Metamask забезпечує безпечне виконання цих взаємодій.

Таким чином, Metamask виконує подвійну роль в архітектурі Dapp (Рисунок 2.13).

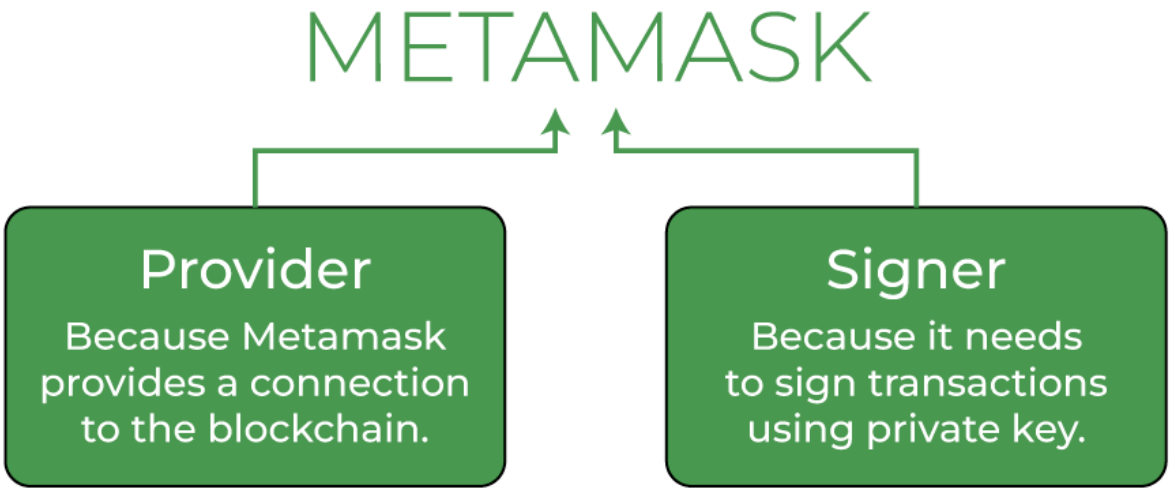


Рисунок 2.13 - Задачі Metamask в архітектурі

Вона функціонує як провайдер, забезпечуючи зв'язок між браузером і блокчейном Ethereum, і як підписувач, забезпечуючи безпечне підписання транзакцій за допомогою приватних ключів, що зберігаються локально в браузері. Таке поєднання функцій робить Metamask важливим компонентом для користувачів, які працюють з децентралізованими додатками в мережі Ethereum.[12]

3 РОЗРОБКА МЕТОДУ ДЕЦЕНТРАЛІЗОВАНОГО ОБМІНУ ФАЙЛАМИ

3.1 Опис архітектури проєкту

Створений проєкт додатку, який реалізує метод децентралізованого обміну файлами складається із трьох основних частин, які містяться в окремих директоріях:

- client - містить код клієнтської частини додатку, що реалізує весь користувацький інтерфейс, за допомогою мови програмування JavaScript та фреймворку React.
- contracts - містить код створеного смарт-контракту на мові програмування Solidity, для взаємодії додатку із блокчейн мережею Ethereum, для реалізації функціоналу файлообміну та доступів до файлів.
- scripts - містить код для розміщення (деплою) створеного смарт-контракту у відповідній блокчейн мережі.

Також існує файл hardhat.config.js - цей файл представляє конфігураційний файл для фреймворку Hardhat, який використовується для розробки і тестування смарт-контрактів на блокчейні Ethereum. Давайте розглянемо його елементи:

- require("@nomicfoundation/hardhat-toolbox") - використовується для підключення допоміжного пакету hardhat-toolbox, який надає додаткові інструменти для роботи з Hardhat.
- solidity: "0.8.9" - версія мови Solidity, яку використовуватиме проект.
- networks - налаштування для різних мереж (у даному випадку, лише для мережі Hardhat).
- hardhat: { chainId: 1337 } - налаштування для мережі Hardhat з ідентифікатором ланцюга 1337.
- paths - Налаштування шляхів для збереження артефактів (згенерованих ABI та бінарних кодів контрактів).
- artifacts: "./client/src/artifacts" - шлях для збереження артефактів у фронтенді проекту.

Цей файл дозволяє конфігурувати різні аспекти роботи Hardhat під час розробки смарт-контрактів на платформі Ethereum.

Розглянемо детальніше структуру створеного смарт контракту contracts/Upload.sol:

- Створена структура Access - містить адресу користувача та булеве значення доступу (true або false).
- Мапування value - Визначає мапування, яке пов'язує адресу користувача з масивом рядків (URL).
- Мапування ownership - визначає мапування, яке вказує, чи належить власник контракту іншому користувачеві.
- Мапування accessList - визначає мапування, яке зберігає список доступу до контракту для кожного користувача.
- Мапування previousData - визначає мапування для зберігання попередніх даних про доступ.
- Функція add - додає новий URL до масиву значень для вказаного користувача.
- Функція allow - надає дозвіл на доступ для вказаного користувача та вносить зміни в список доступу.
- Функція disallow - забороняє доступ для вказаного користувача та змінює список доступу.
- Функція display - повертає масив URL для вказаного користувача, якщо у викликаючого користувача є доступ.
- Функція shareAccess - повертає список доступу до контракту для викликаючого користувача.

Далі наведений опис клієнтської частини свореного додатку, який міститься в client/src:

- index.js - точка входу в додаток, місце де додаток вбудовується в HTML-сторінку.
- App.js - головний компонент додатку, що містить в собі основну логіку взаємодії із блокчейн мережею.

- FileUpload.js - компонент, який містить в собі логіку по завантаженню файлів зображень до додатку.
- Display.js - компонент, що відповідає за відображення файлів зображень, до яких користувач має доступ.
- Modal.js - компонент, що інкапсулює логіку по наданню доступу до файлів зображень для інших користувачів.

Головний файл App.js виконує такі функції:

- Здійснює імпорт ABI та адреси смарт-контракту з його JSON-представлення.
- Імпортує необхідні бібліотеки React та Ethers для роботи з фронтендом та блокчейном відповідно.
- Імпортує компоненти React для завантаження файлів (FileUpload), відображення вмісту (Display) та модального вікна (Modal).
- Визначає головний компонент React, який містить стани для рахунку користувача, контракту та провайдера, а також стан для відкриття модального вікна.
- Використовує функцію useEffect для ініціалізації, включаючи підключення до мережі Ethereum, отримання аккаунту та створення об'єкта контракту.
- Виводить кнопку Share, яка відкриває модальне вікно для поділу контракту.
- Виводить заголовок додатка, інформацію про обліковий запис користувача та компоненти для завантаження файлів та відображення вмісту контракту.

Компонент FileUpload.js, відповідає за завантаження файлів на IPFS за допомогою сервісу Pinata та їх подальше збереження у смарт-контракті Ethereum. Давайте розглянемо основні функції цього файлу:

- Здійснює імпорт бібліотек React (для useState), Axios (для виконання HTTP-запитів) та стилів.
- Оголошує компонент FileUpload, який приймає параметри contract, account і provider.
- Використовує стейти для відстеження обраного файлу та його назви.
- Містить функцію handleSubmit, яка викликається при поданні форми для завантаження файлу, вона виконує HTTP-запит для завантаження файлу на

IPFS, отримує хеш та зберігає інформацію про файл у смарт-контракті.

- Містить функцію retriveFile, яка викликається при виборі файлу користувачем, вона отримує обраний файл, читає його та оновлює стан компонента.
- Повертає JSX, який представляє форму для вибору та завантаження файлу.
- Дозволяє компоненту FileUpload бути експортованим для використання в інших частинах додатка.

Компонент Display.js Цей код представляє компонент React для відображення даних, отриманих з смарт-контракту Ethereum. Основні функції цього коду:

- Імпортує useState з бібліотеки React для роботи зі станом компонента та стилі з файлу Display.css.
- Визначає компонент Display, який приймає параметри contract (смарт-контракт) і account (адреса облікового запису користувача).
- Використовує стан для зберігання отриманих даних з контракту.
- Містить функцію, яка викликається при натисканні кнопки Get Data. Здійснює виклик методу контракту display для отримання даних.
- Містить логіку, коли після отримання даних перевіряється їхній наявність, і, якщо дані не порожні, вони розбираються та виводяться у вигляді зображень.
- Містить JSX, що представляє компонент для відображення даних. Включає контейнер для зображень, поле введення адреси та кнопку для виклику функції отримання даних.
- Дозволяє компоненту Display бути експортованим для використання в інших частинах додатка.

Останній компонент клієнтської частини додатку це Modal.js. Цей файл містить компонент React для модального вікна, яке використовується для реалізації можливості Share у додатку. Давайте розглянемо основні частини цього коду:

- Здійснює імпорт useEffect з React для використання ефекту під час відображення компоненту та стилів з файлу Modal.css.
- Визначає компонент Modal, який приймає параметри setModalOpen (функція

для закриття модального вікна) та contract (смарт-контракт).

- Містить функцію sharing, яка викликається при натисканні кнопки Share.
- Викликає метод контракту allow для обміну доступом.
- Містить функцію useEffect, який викликається при відображенні компоненту та при зміні смарт-контракту (contract). Отримує список доступних адрес і оновлює випадаючий список.
- Містить JSX, що представляє вікно для обміну доступом. Включає заголовок, поле введення адреси, випадаючий список доступних адрес та кнопки для скасування та обміну доступом.
- Дозволяє компоненту Modal бути експортованим для використання в інших частинах додатка.

Також існують файли стилів CSS для кожного і створений React компонентів App.css, FileUpload.css, Modal.css, Display.css.

3.2 Тестування

Для тестування створеного децентралізованого додатку для обміну файлів із використанням технології Blockchain необхідно спочатку створити криптогаменець Metamask та встановити відповідне розширення для браузера, що мати можливість використовувати додаток (Рисунок 3.1).

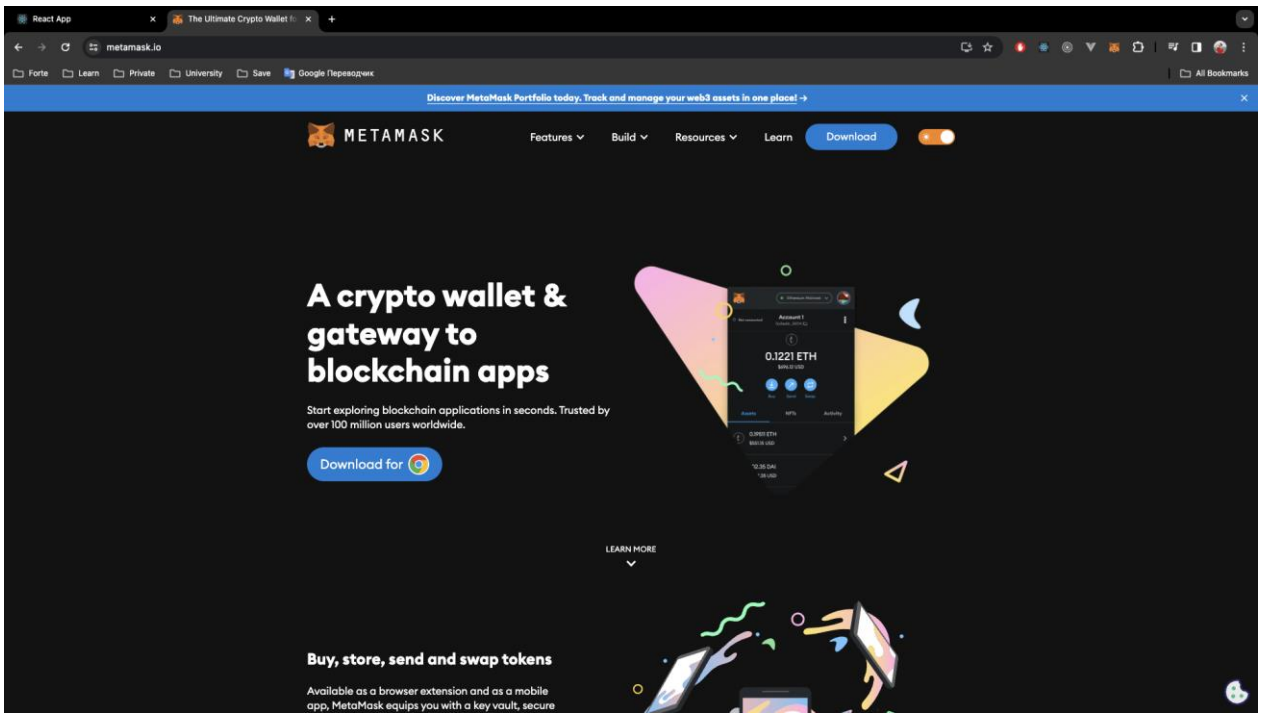


Рисунок 3.1 - Реєстрація та завантаження Metamask

Після того як розширення було встановлене необхідно залогінитись в криптогаманець, далі наведений стан налаштованого для тестування криптогаманця (Рисунок 3.2).

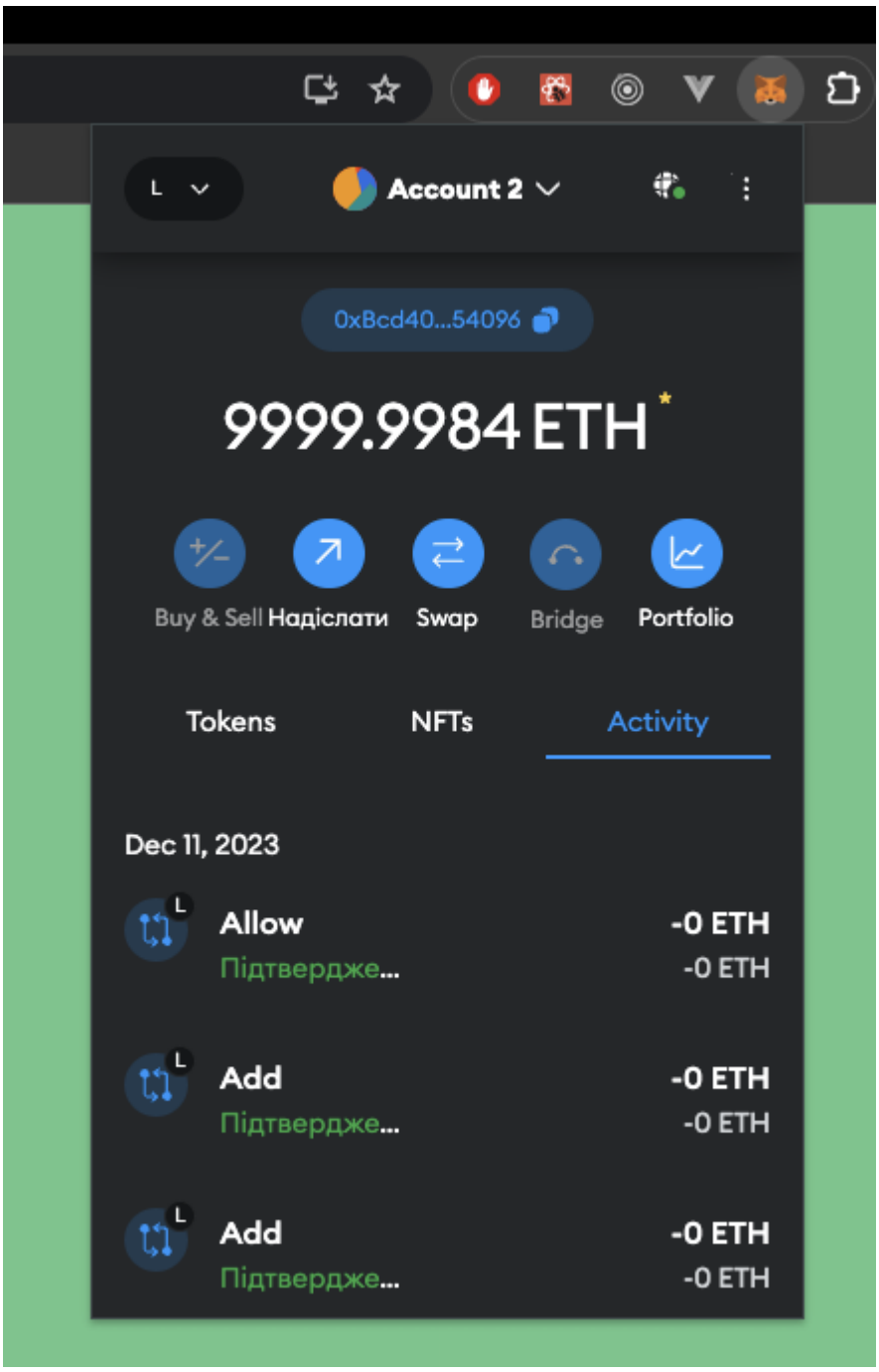


Рисунок 3.2 - Готовий до тестування криптогаманець

Крім того, необхідно вибрати відповідну мережу для тестування в якій існують створені аккаунти. В даному випадку тестування буде відбуватись на тестовій мережі Localhost 8545 (Рисунок 3.3).

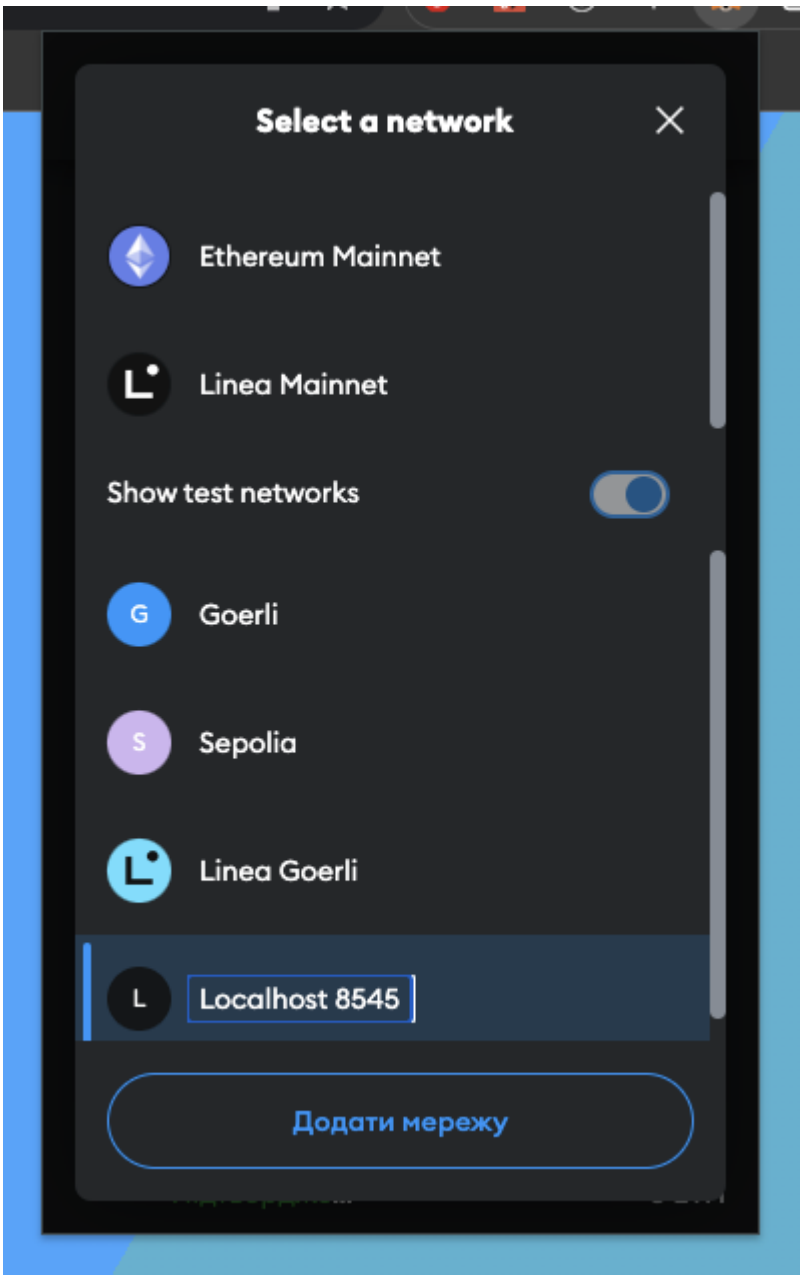


Рисунок 3.3 - Вибір тестової мережі

Для тестування необхідно додати мінімум два аккаунти із наявним мінімальним об’ємом криптовалюти на балансі, тому що кожна дія із криптогаманцем коштує відповідний об’єм комісії. В даному прикладі будуть брати участь аккаунти під номером під номером 4 та 5 (Рисунок 3.4).

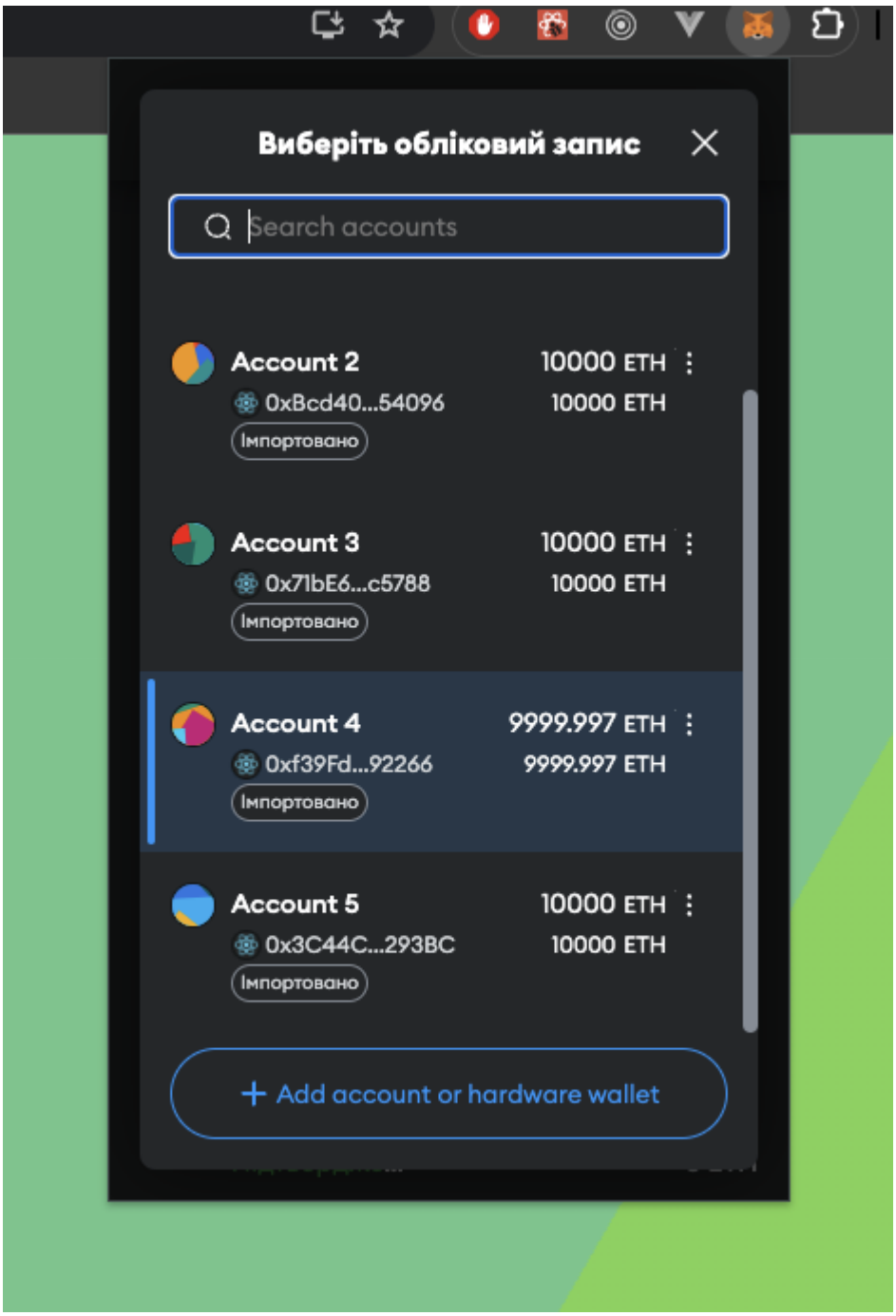


Рисунок 3.4 - Аккаунти для тестування

Далі наведений загальний вигляд додатку із підключеним криптогаманцем. На екрані можна побачити декілька елементів (Рисунок 3.5):

- назву додатку
- адресу акканту підключеного до додатку через криптогаманець
- кнопку для вибору завантажуваного зображення
- кнопку для завантаження зображення до блокчейн мережі

- поле адреси для отримання файлів зображень
- кнопку для отримання файлів зображень
- кнопку для надання доступу до завантажених файлів зображень

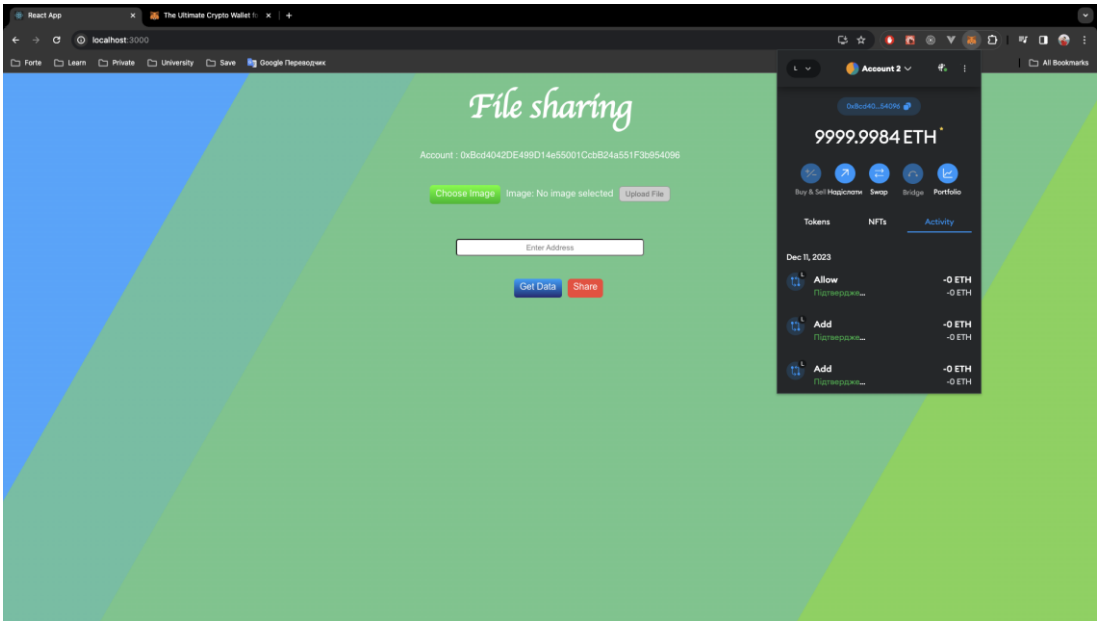


Рисунок 3.5 - Загальний вигляд додатку

Для того, щоб завантажити файл зображення до криптомережі необхідно натиснути кнопку Choose image та вибрати відповідний файл (Рисунок 3.6).

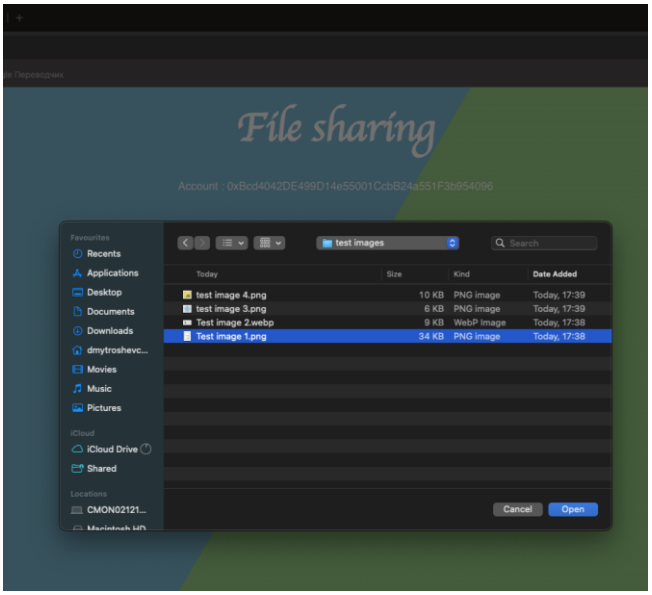


Рисунок 3.6 - Вибір зображення для завантаження

Після того як файл був успішно обраний на екрані з’явиться його назва та розширення із повідомленням про успішне додавання (Рисунок 3.7).

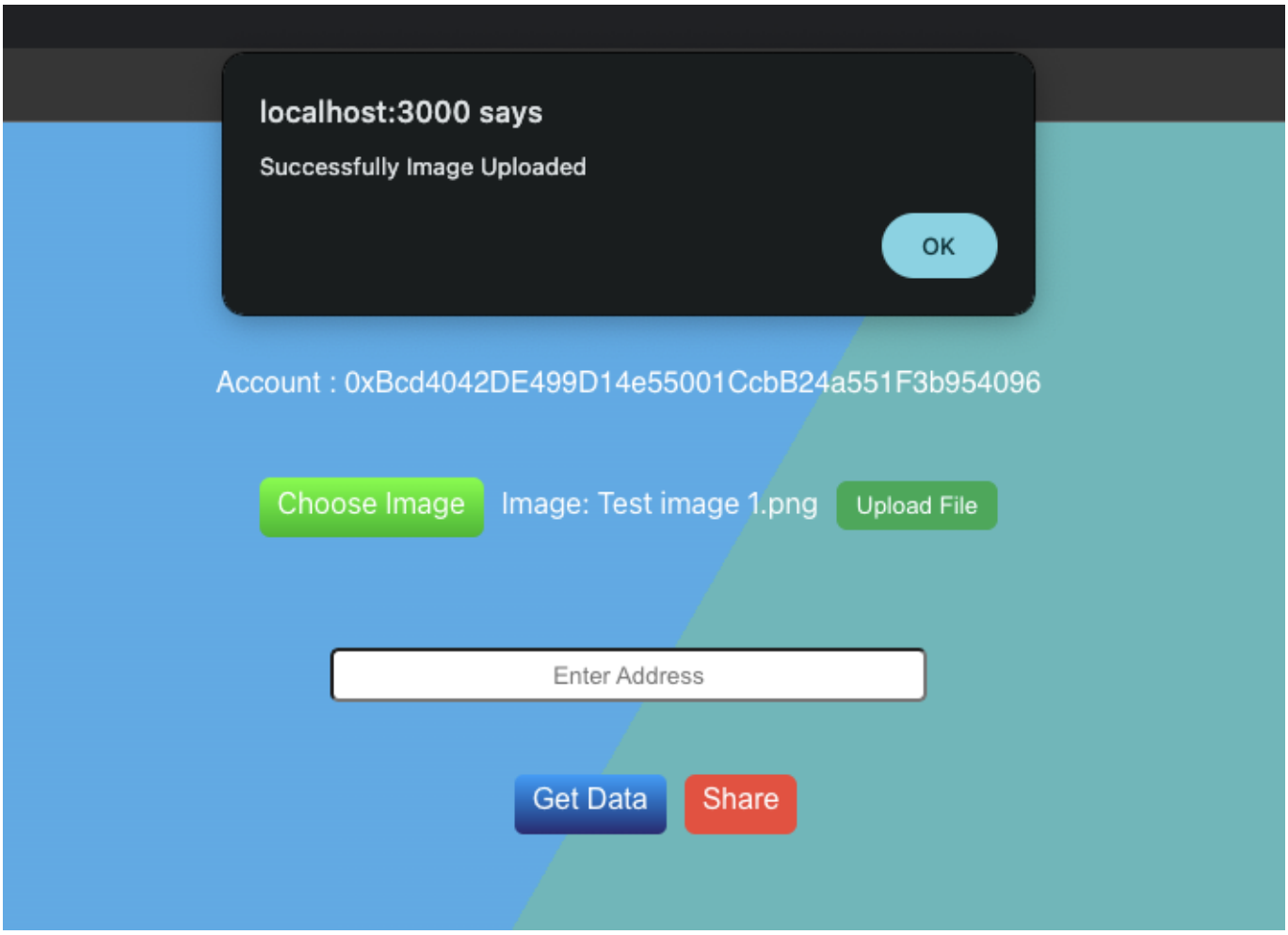


Рисунок 3.7 - Успішне додавання файлу зображення

Далі необхідно завантажити файл до блокчейн мережі. Для цього необхідно натиснути на кнопку Upload file. Після цього з’явиться вікно із підтвердженням проведення операції в блокчейн мережі. В цьому вікні буде вказано адресу гаманця який проводить операцію та кількість комісії, яка буде списана за її проведення (Рисунок 3.8).

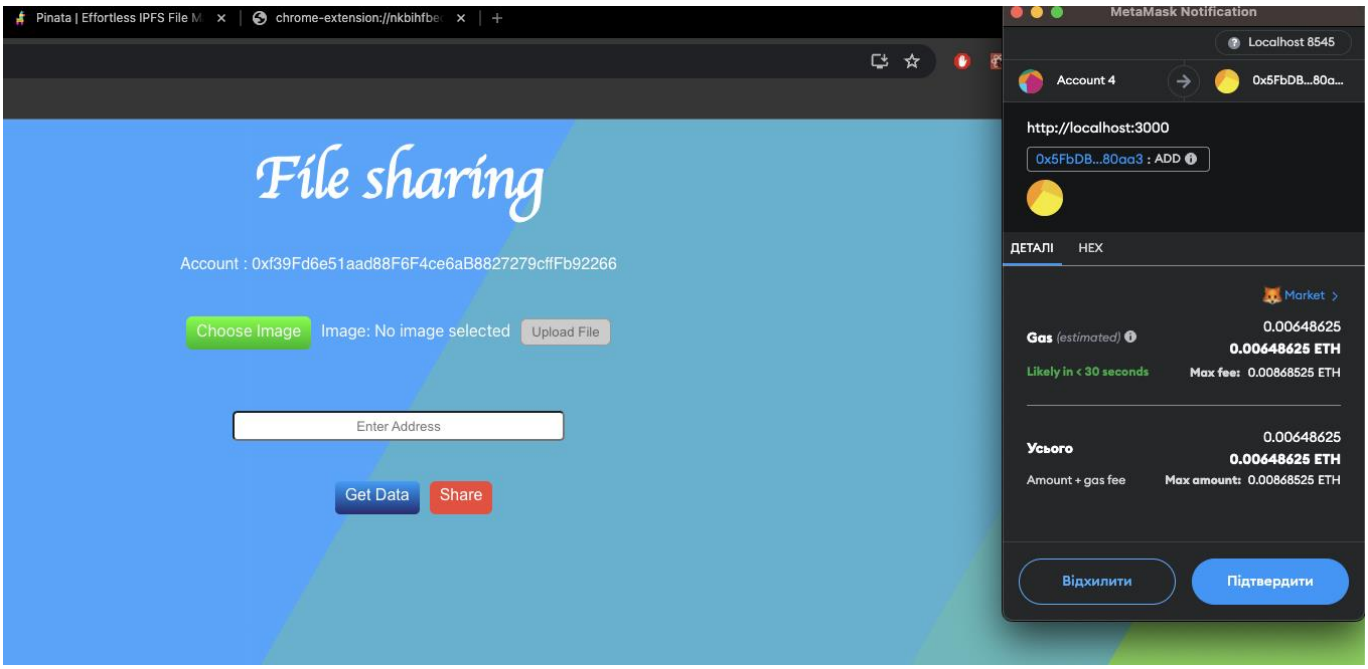


Рисунок 3.8 - Підтвердження операції в блокчейн мережі

Після підтвердження операції вона буде додана в історію виконаних операцій, із відповідним статусом в даному випадку - підтверджено, що означає що вона пройшла успішно (Рисунок 3.9).

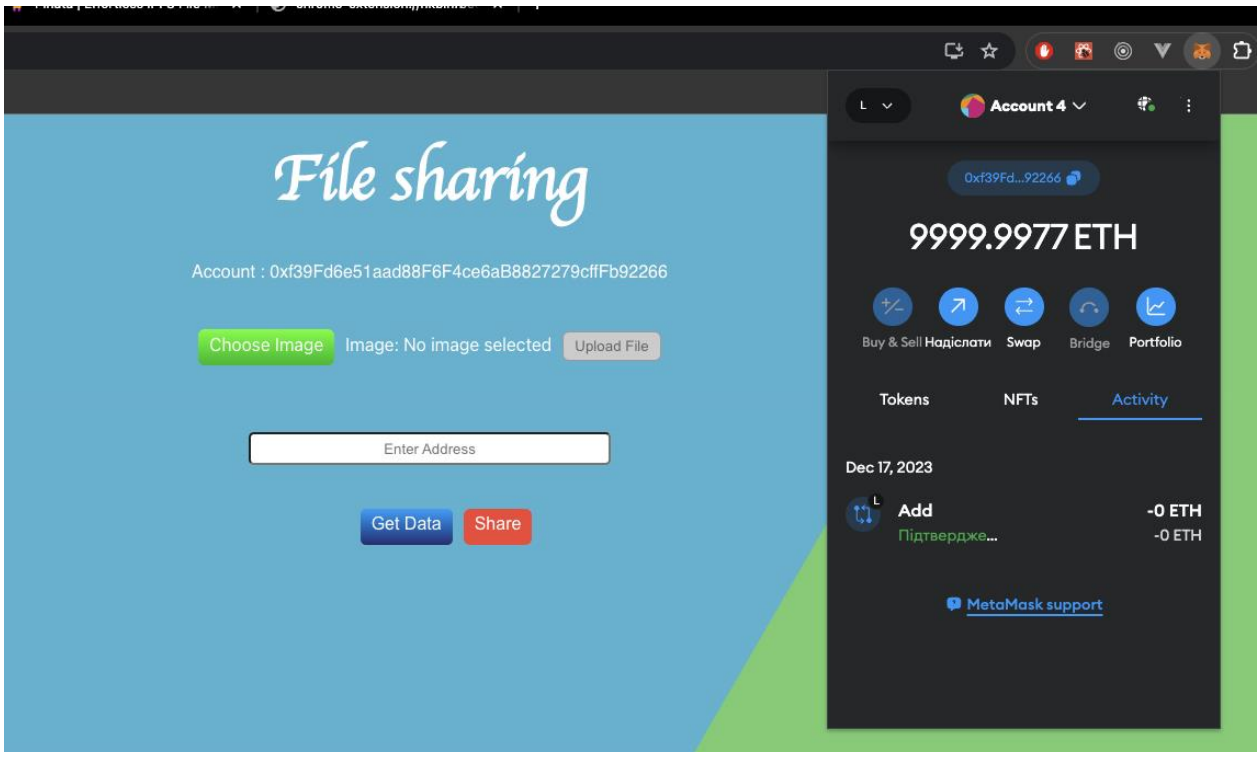


Рисунок 3.9 - Успішне підтвердження проведеної операції

Далі для більшої наглядності можна повторити наведені дії для іншого файлу зображення. Коли декілька зображень було завантажено до блокчейн мережі, можна тестувати їх отримання. Для того, щоб завантажити зображення, що вже знаходяться в мережі для обраного аккаунту, необхідно просто натиснути кнопку Get Data. Після цього отримані зображення, будуть відображатись на екрані. Як бачимо отримання відбулось успішно, без помилок, швидко та ефективно (Рисунок 3.10).

Рисунок 3.10 - Отримання завантажених файлів зображень із блокчейн мережі

Тепер необхідно надати доступ, до цих зображень іншому аккаунту за іншою адресою. Для цього, натискаємо кнопку Share. Після цього на екрані з’явиться модальне вікно, в яке необхідно ввести адресу аккаунту, якому необхідно надати доступ до файлів, які були завантажені для поточного аккаунту. Також в модальному вікні представлений випадаючий список в якому зазначені аккаунти,

які вже мають доступ до файлів, тобто яким був наданий доступ раніше (Рисунок 3.11).

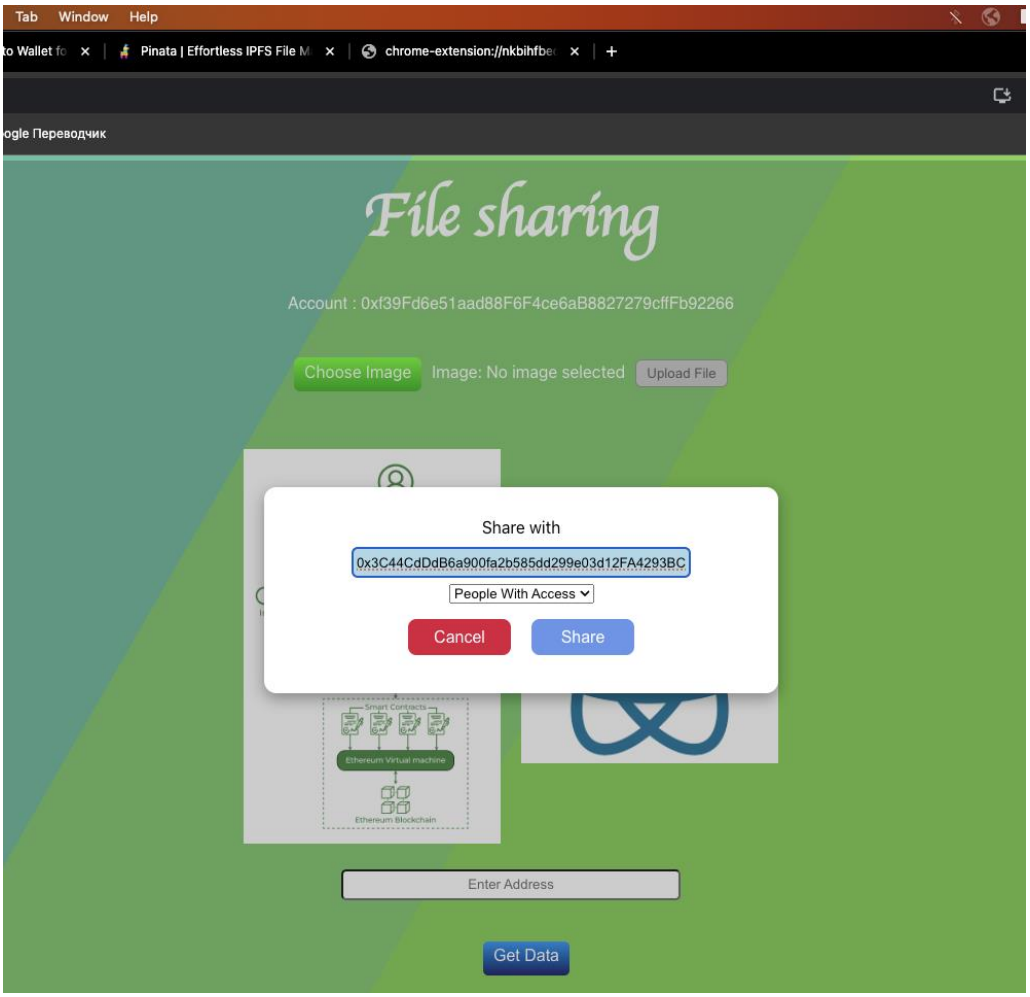


Рисунок 3.11 - Надання доступу до файлів іншому аккаунту

Після цього, необхідно знову підтвердити дію в блокчейн мережі через криптогаманець, за цю операцію також буде списано відповідний об’єм комісії (Рисунок 3.12).

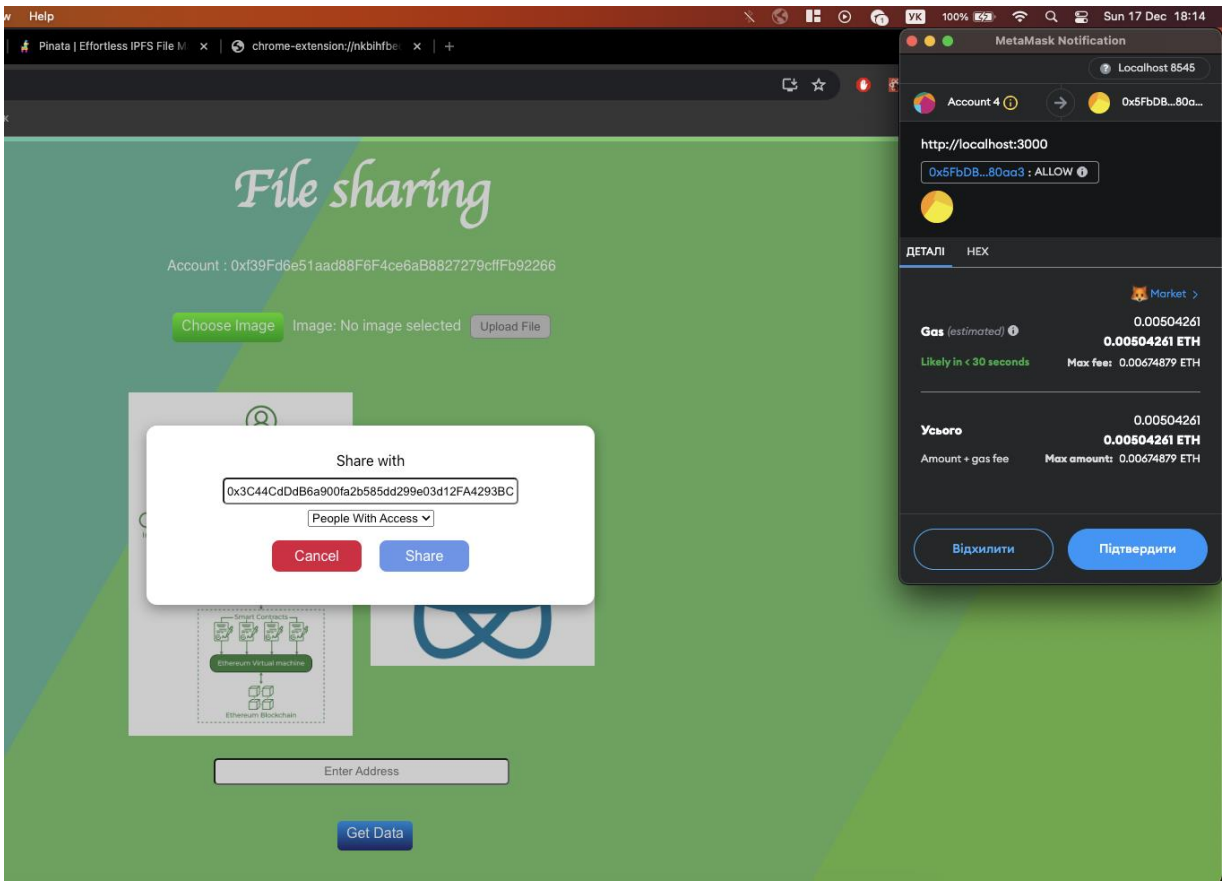


Рисунок 3.12 - Підтвердження надання доступу

Тепер протестуємо отримання зображень через аккаунт, якому тільки що був наданий доступ. Для цього за допомогою криптогаманця, необхідно змінити підключений до додатку аккаунт. Після того як, ми виконали цю дію, бачимо, що адреса аккаунту на екрані змінилась на поточну, а завантажені для минулого аккаунту файли зображень зникли (Рисунок 3.13).

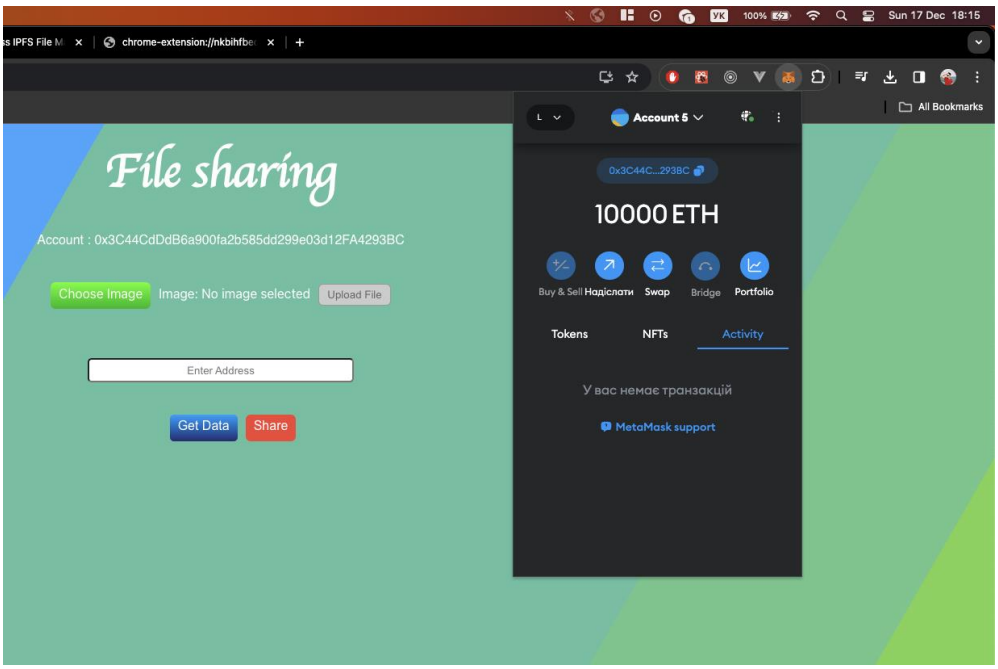


Рисунок 3.13 - Зміна аккаунту, на тей, якому надали доступ

Тепер для того, щоб знову їх отримати, та перевірити до них доступ із поточного аккаунту, необхідно скопіювати адресу аккаунту, якому належать ці файли та вставити її в поле Address на екрані та натиснути кнопку Get Data, після цього на екрані з’являється файли зображень першого аккаунту для яких був наданий доступ поточному. У випадку відсутності доступу або зображення буде показано відповідне повідомлення із помилкою (Рисунок 3.14).

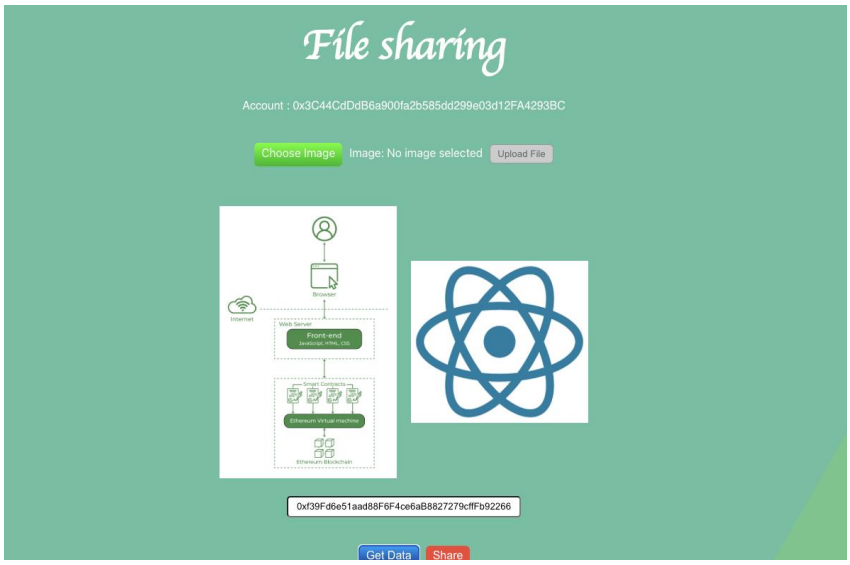


Рисунок 3.14 - Отримання файлів зображень для яких наданий доступ

Тепер можемо протестувати ті ж самі дії, але замінити аккаунти місцями, тобто тепер на поточний аккаунт завантажимо інший файл, та надамо до нього доступ першому аккаунту, який вже має свої зображення (Рисунок 3.15).

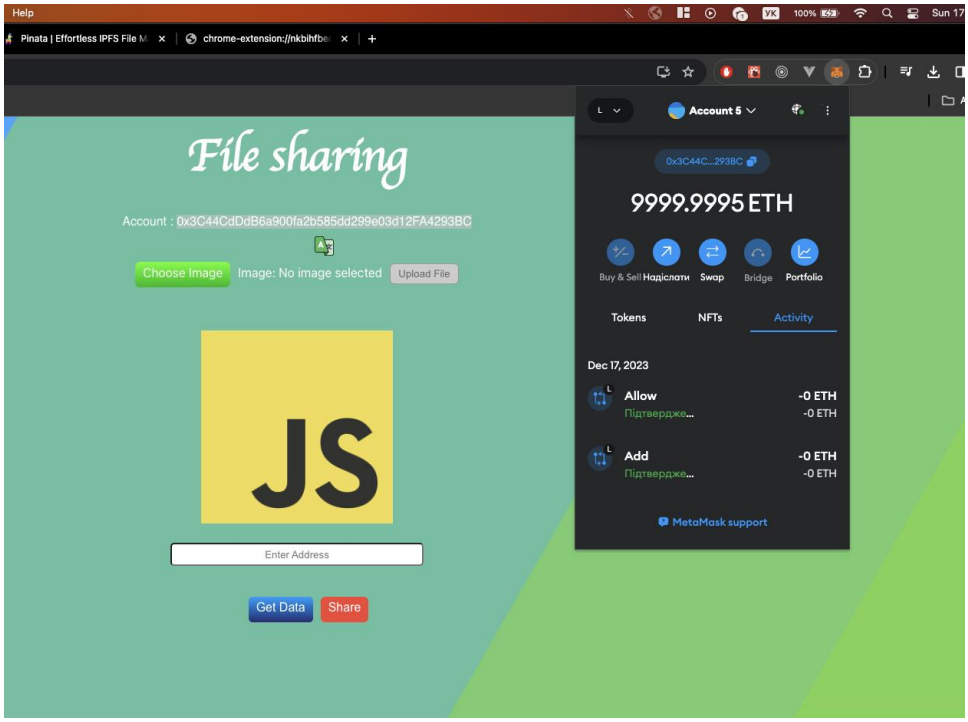


Рисунок 3.15 - Завантажений файл на другий аккаунт та надання доступу до нього першому

Після цього повертаємось на перший аккаунт, для цього пересвідчимось, що адреса аккаунту на екрані змінилась, та спробуємо отримати файли другого аккаунту, для цього вставляємо його адресу в поле та натискаємо кнопку Get Data. Бачимо, що файли другого аккаунту успішно отримано та вони відображаються на екрані (Рисунок 3.16).

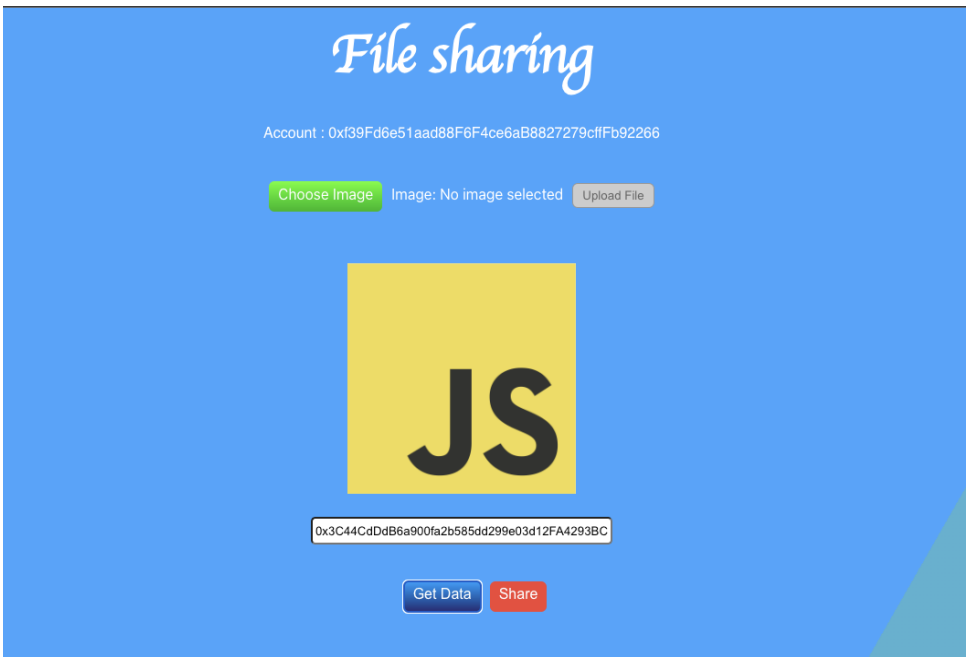


Рисунок 3.16 - Отримання файлів другого аккаунту на першому, якому було надано доступ

Також можемо перевірити, що власні файли першого аккаунту, ми також можемо отримати, просто очистивши поле адреси та знову натиснути кнопку GetData, бо за замовчуванням, якщо поле пусте, то додаток отримує зображення для адреси поточного аккаунту (Рисунок 3.17).

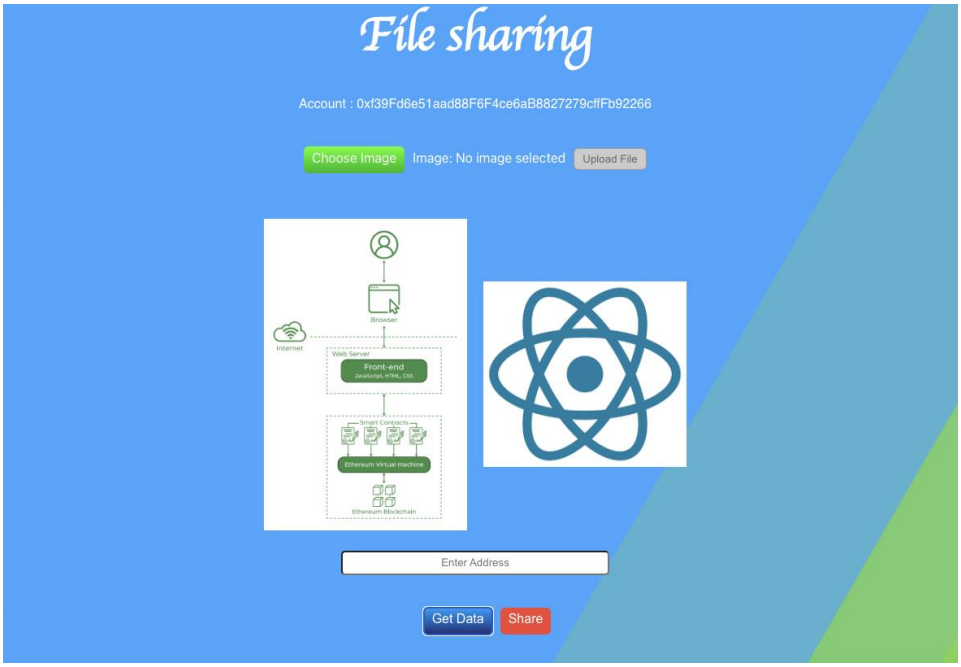


Рисунок 3.17 - Отримання власних зображень аккаунту

Як бачимо, додаток відпрацьовує абсолютно корректно, без збоїв та помилок. Реалізований функціонал дозволяє:

- отримати доступ до блокчейн мережі через криптогаманець Metamask
- завантажувати файли зображень до блокчейн мережі
- надавати доступ до своїх зображень іншим користувачам, із іншими аккаунтами в цій мережі
- отримувати, як свої, так файли зображень із блокчейн мережі, так і ті, для яких надали доступ

Слід зазначити, реалізований додаток, має потенціал для покращень та вдосконалень, наприклад додавання розширених функцій для керування доступом до файлів або додавання інших методів завантаження файлів.

Можна зробити висновок, що ціль розробити метод децентралізованого обміну файлами з використанням технології Blockchain, спрямованого на підвищення безпеки, ефективності та надійності обміну інформацією між користувачами було успішно виконано у вигляді децентралізованого додатку.

ВИСНОВКИ

Метою даної дипломної роботи була розробка методу децентралізованого обміну файлами з використанням технології Blockchain з метою підвищення безпеки, ефективності та надійності обміну інформацією між користувачами. Під час її виконання було досліджено та вивчено еволюцію технологій обміну файлами, проведено аналіз технології Blockchain та реалізовано відповідний метод децентралізованого обміну файлами.

Об'єктом дослідження були технології обміну файлами та їх розвиток в інформаційному середовищі. Предметом дослідження став метод децентралізованого обміну файлами з використанням технології Blockchain як інноваційного інструменту для підвищення безпеки та надійності обміну файловою інформацією.

У роботі проведено аналіз розвитку технологій для обміну файлами, визначено основні тенденції та характеристики. Розглянуто еволюцію цих технологій, враховано важливі аспекти їхнього використання в сучасному інформаційному середовищі.

Також було вивчено принципи та можливості технології Blockchain, досліджено її вплив на обмін файлами. Зазначено переваги та можливості використання Blockchain для підвищення безпеки та надійності обміну файловою інформацією.

Здійснено розробку та реалізацію методу децентралізованого обміну файлами на основі технології Blockchain. Проведено тестування функціональності, ефективності та безпеки запропонованого методу. Крім того, реалізований додаток, має потенціал для покращень та вдосконалень, наприклад додавання розширених функцій для керування доступом до файлів або додавання інших методів для завантаження файлів.

Отримані результати дозволяють зробити висновок, що розроблений метод може слугувати ефективним рішенням для децентралізованого обміну файлами в різних галузях, де важливо поєднати безпеку та надійність із сучасними

технологічними вимогами.

Дана дипломна робота вносить важливий внесок у галузь розробки методів обміну файлами, розширює розуміння можливостей технології Blockchain у цьому контексті та підкреслює її потенціал для подальших досліджень та впроваджень.

ПЕРЕЛІК ПОСИЛАНЬ

1. Michael J. Folk, Bill Zoellick, Greg Riccardi - File Structures - An Object-Oriented Approach With C++-Addison-Wesley (1998) | PDF | Hard Disk Drive | Computer File. Scribd. URL: https://www.scribd.com/document/514637904/Michael-J-Folk-Bill-Zoellick-Greg-Riccardi-File-Structures-an-Object-Oriented-Approach-With-C-Addison-Wesley-1998 (дата звернення: 06.12.2023).
2. "Designing & Deploying Server & Storage Solutions for Small and Medium Business First Edition.". URL: https://www.certiport.com/portal/common/htmllibrary/hp-test-fest/docs/HPATA-SS-Study-Guide.pdf
3. Distributed Systems 3rd edition (2017) - DISTRIBUTED-SYSTEMS.NET. URL: https://www.distributed-systems.net/index.php/books/ds3/ (дата звернення: 06.12.2023).
4. Bashir, Imran. "Mastering Blockchain.". URL: https://books.google.com.ua/books/about/Mastering_Blockchain.html?id=dMJbMQAACAAJ&redir_esc=y (дата звернення: 10.10.2023)
5. Drescher, Daniel. "Blockchain Basics: A Non-Technical Introduction in 25Steps.".URL:https://www.scribd.com/book/567559027/Blockchain-Basics-A-Non-Technical-Introduction-in-25-Steps (дата звернення: 10.10.2023)
6. Antonopoulos A. M., D G. W. P. Mastering Ethereum: building smart contracts and dapps. O'Reilly Media, 2018. 424 p.
7. Ethereum whitepaper. URL: https://ethereum.org/en/whitepaper/ (дата звернення: 27.10.2023).
8. Solidity documentation. URL: https://docs.soliditylang.org/en/v0.8.22/ (дата звернення: 27.10.2023).
9. React documentation. URL: https://react.dev/blog/2023/03/16/introducing-react-dev (дата звернення: 27.10.2023).
10. Ether documentation. URL: https://docs.ethers.org/v5/ (дата звернення:

27.10.2023).

11. Web3 documentation. URL: https://web3js.readthedocs.io/en/v1.10.0/ (дата звернення: 27.10.2023).
12. Metamask documentation. URL: https://web3js.readthedocs.io/en/v1.10.0/ (дата звернення: 27.10.2023).

ДОДАТКИ

ДОДАТОК А
Програмний код

Upload.sol

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract Upload {

 struct Access{
    address user;
    bool access; //true or false
 }
 mapping(address=>string[]) value;
 mapping(address=>mapping(address=>bool)) ownership;
 mapping(address=>Access[]) accessList;
 mapping(address=>mapping(address=>bool)) previousData;

 function add(address _user,string memory url) external {
     value[_user].push(url);
 }
 function allow(address user) external {
     ownership[msg.sender][user]=true;
     if(previousData[msg.sender][user]){
        for(uint i=0;i<accessList[msg.sender].length;i++){
            if(accessList[msg.sender][i].user==user){
                 accessList[msg.sender][i].access=true;
            }
        }
     }else{
         accessList[msg.sender].push(Access(user,true));
         previousData[msg.sender][user]=true;
     }

 }
 function disallow(address user) public{
     ownership[msg.sender][user]=false;
     for(uint i=0;i<accessList[msg.sender].length;i++){
         if(accessList[msg.sender][i].user==user){
             accessList[msg.sender][i].access=false;
         }
     }
 }

 function display(address _user) external view returns(string[] memory){
     require(_user==msg.sender || ownership[_user][msg.sender],"You don't have
access");
     return value[_user];
 }
```

```
 function shareAccess() public view returns(Access[] memory){
     return accessList[msg.sender];
 }
}
```

deploy.js

```
const hre = require("hardhat");

async function main() {
 const Upload = await hre.ethers.getContractFactory("Upload");
 const upload = await Upload.deploy();

 await upload.deployed();

 console.log("Library deployed to:", upload.address);
}

main().catch((error) => {
 console.error(error);
 process.exitCode = 1;
});
```

App.jsx

```
import Upload from "./artifacts/contracts/Upload.sol/Upload.json";
import { useState, useEffect } from "react";
import { ethers } from "ethers";
import FileUpload from "./components/FileUpload";
import Display from "./components/Display";
import Modal from "./components/Modal";
import "./App.css";

function App() {
 const [account, setAccount] = useState("");
 const [contract, setContract] = useState(null);
 const [provider, setProvider] = useState(null);
 const [modalOpen, setModalOpen] = useState(false);

 useEffect(() => {
   const provider = new ethers.providers.Web3Provider(window.ethereum);

   const loadProvider = async () => {
     if (provider) {
       window.ethereum.on("chainChanged", () => {
         window.location.reload();
       });

       window.ethereum.on("accountsChanged", () => {
         window.location.reload();
       });
       await provider.send("eth_requestAccounts", []);
       const signer = provider.getSigner();
       const address = await signer.getAddress();
       setAccount(address);
       let contractAddress = "0x5FbDB2315678afecb367f032d93F642f64180aa3";
```

```
        const contract = new ethers.Contract(
          contractAddress,
          Upload.abi,
          signer
        );
        //console.log(contract);
        setContract(contract);
        setProvider(provider);
      } else {
        console.error("Metamask is not installed");
      }
    };
    provider && loadProvider();
  }, []);
  return (
    <>
      {modalOpen && (
        <Modal setModalOpen={setModalOpen} contract={contract}></Modal>
      )}

      <div className="App">
        <h1 style={{ color: "white" }}>File sharing</h1>
        <div class="bg"></div>
        <div class="bg bg2"></div>
        <div class="bg bg3"></div>

        <p style={{ color: "white" }}>
          Account : {account ? account : "Not connected"}
        </p>
        <FileUpload
          account={account}
          provider={provider}
          contract={contract}
        ></FileUpload>
        <Display contract={contract} account={account}></Display>
        {!modalOpen && (
            <button className="share" onClick={() => setModalOpen(true)}>
              Share
            </button>
        )}
      </div>
    </>
  );
}

export default App;
```

FileUpload.jsx

```
import {useState} from "react";
import axios from "axios";
import "./FileUpload.css";

const FileUpload = ({contract, account, provider}) => {
   const [file, setFile] = useState(null);
```

```
    const [fileName, setFileName] = useState("No image selected");
    const handleSubmit = async (e) => {
        e.preventDefault();
        if (file) {
            try {
                const formData = new FormData();
                formData.append("file", file);

                const resFile = await axios({
                    method: "post",
                    url: "https://api.pinata.cloud/pinning/pinFileToIPFS",
                    data: formData,
                    headers: {
                        pinata_api_key: `567805efe302570567db`,
                        pinata_secret_api_key:
`725017b53c07fdfcc9b0996e1a058f9dd47d71eb78ecc012a2d344741a8512d5`,
                        "Content-Type": "multipart/form-data",
                    },
                });
                const ImgHash =
`https://gateway.pinata.cloud/ipfs/${resFile.data.IpfsHash}`;

                contract.add(account, ImgHash);
                alert("Successfully Image Uploaded");
                setFileName("No image selected");
                setFile(null);
            } catch (e) {
                alert("Unable to upload image to Pinata");
            }
        }
        alert("Successfully Image Uploaded");
        setFileName("No image selected");
        setFile(null);
    };
    const retrieveFile = (e) => {
        const data = e.target.files[0]; //files array of files object
        const reader = new window.FileReader();
        reader.readAsArrayBuffer(data);
        reader.onloadend = () => {
            setFile(e.target.files[0]);
        };
        setFileName(e.target.files[0].name);
        e.preventDefault();
    };
    return (
        <div className="top">
            <form className="form" onSubmit={handleSubmit}>
                <label htmlFor="file-upload" className="choose">
                    Choose Image
                </label>
                <input
                    disabled={!account}
                    type="file"
                    id="file-upload"
                    name="data"
                    onChange={retrieveFile}
```

```
                />
                <span className="textArea">Image: {fileName}</span>
                <button type="submit" className="upload" disabled={!file}>
                    Upload File
                </button>
            </form>
        </div>
    );
};
export default FileUpload;
```

Display.jsx

```
import { useState } from "react";
import "./Display.css";
const Display = ({ contract, account }) => {
 const [data, setData] = useState("");
 const getdata = async () => {
   let dataArray;
   const Otheraddress = document.querySelector(".address").value;
   try {
     if (Otheraddress) {
       dataArray = await contract.display(Otheraddress);
       console.log(dataArray);
     } else {
       dataArray = await contract.display(account);
     }
   } catch (e) {
     alert("You don't have access");
   }
   const isEmpty = Object.keys(dataArray).length === 0;

   if (!isEmpty) {
     const str = dataArray.toString();
     const str_array = str.split(",");
     const images = str_array.map((item, i) => {
       console.log(item);
       return (
         <a href={item} key={i} target="_blank">
           <img
             key={i}
             crossOrigin="anonymous"
             src={item}
             alt="new"
             className="image-list"
           ></img>
         </a>
       );
     });
     setData(images);
   } else {
     alert("No image to display");
   }
 };
 return (
   <>
```

```
      <div className="image-list">{data}</div>
      <input
        type="text"
        placeholder="Enter Address"
        className="address"
      ></input>
      <button className="center button" onClick={getdata}>
        Get Data
      </button>
    </>
  );
};
export default Display;
```

Modal.jsx

```
import { useEffect } from "react";
import "./Modal.css";
const Modal = ({ setModalOpen, contract }) => {
 const sharing = async () => {
   const address = document.querySelector(".address").value;
   await contract.allow(address);
   setModalOpen(false);
 };
 useEffect(() => {
   const accessList = async () => {
     const addressList = await contract.shareAccess();
     let select = document.querySelector("#selectNumber");
     const options = addressList;

     for (let i = 0; i < options.length; i++) {
       let opt = options[i];
       let e1 = document.createElement("option");
       e1.textContent = opt;
       e1.value = opt;
       select.appendChild(e1);
     }
   };
   contract && accessList();
 }, [contract]);
 return (
   <>
     <div className="modalBackground">
       <div className="modalContainer">
         <div className="title">Share with</div>
         <div className="body">
           <input
             type="text"
             className="address"
             placeholder="Enter Address"
           ></input>
         </div>
         <form id="myForm">
           <select id="selectNumber">
             <option className="address">People With Access</option>
           </select>
```

```
          </form>
          <div className="footer">
            <button
              onClick={() => {
                setModalOpen(false);
              }}
              id="cancelBtn"
            >
              Cancel
            </button>
            <button onClick={() => sharing()}>Share</button>
          </div>
        </div>
      </div>
    </>
  );
};
export default Modal;
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

ДИПЛОМНА РОБОТА
на ступінь вищої освіти магістр
із спеціальності 122 Комп'ютерні технології

Розробка методу децентралізованого обміну файлами із використанням технології Blockchain

Виконав: студент 6 курсу, групи КНДМ-63
Шевченко Дмитро Сергійович
Керівник: д.т.н., професор
Ільїн О.О.

Київ - 2023

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДИПЛОМНОЇ РОБОТИ

Тема	Розробка методу децентралізованого обміну файлами із використанням технології Blockchain
Мета дослідження	Розробити метод децентралізованого обміну файлами із використанням технології Blockchain, що спрямований на підвищення безпеки, ефективності та надійності обміну інформацією між користувачами.
Наукове завдання	Розробка децентралізованого додатку для обміну файлами використовуючи технологію Blockchain
Об’єкт дослідження	Технологія обміну файлами та їх розвиток в сучасному інформаційному середовищі
Предмет дослідження	Метод децентралізованого обміну файлами

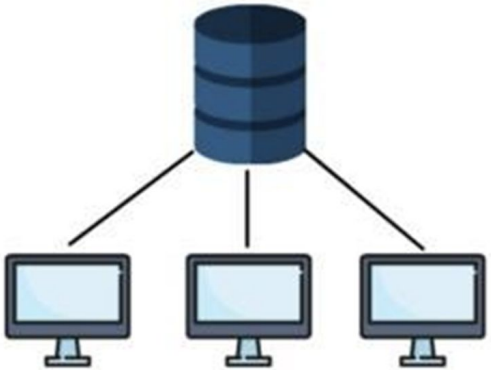
Аналіз файлообмінних мереж

Файлообмінна мережа - це система, в якій комп'ютери або інші пристрої підключені один до одного з метою обміну файлами та ресурсами. Ці мережі створюються для того, щоб дозволити користувачам надсилати, отримувати та обмінюватися даними, такими як тексти, зображення, відео, аудіофайли, програми тощо.

1.1 - Схема процесу обміну файлами в файлообмінних мережах

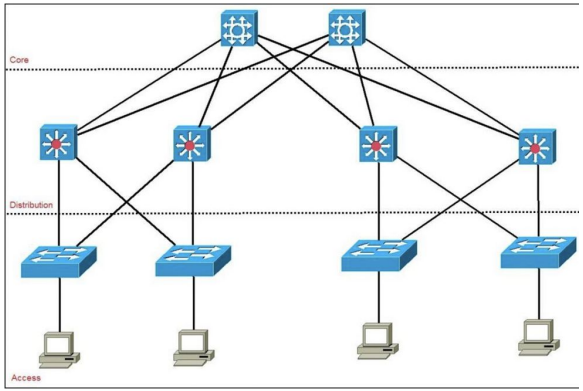
Види моделей обміну даними

- Централізована
- Децентралізована
- Ієрархічна



1.2 - Централізована

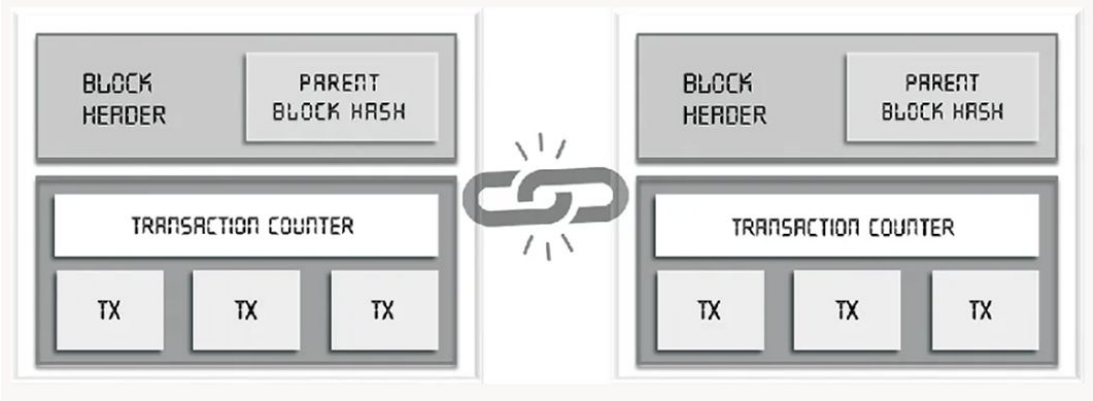
1.3 - Децентралізована



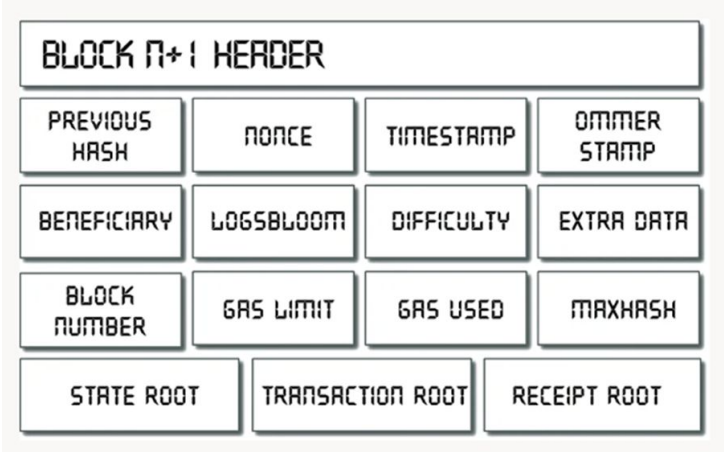
1.4 - Ієрархічна

Аналіз технології Blockchain

Blockchain - це тип технології розподіленого реєстру (DLT), в якому записи про транзакції зберігаються в реєстрі у вигляді ланцюжка блоків. Всі вузли ведуть спільний реєстр, тому загальна обчислювальна потужність розподіляється між ними, що забезпечує кращий результат. Всі дані глибоко зашифровані (хешовані), що забезпечує вищий рівень безпеки. Це унеможливлює злам.



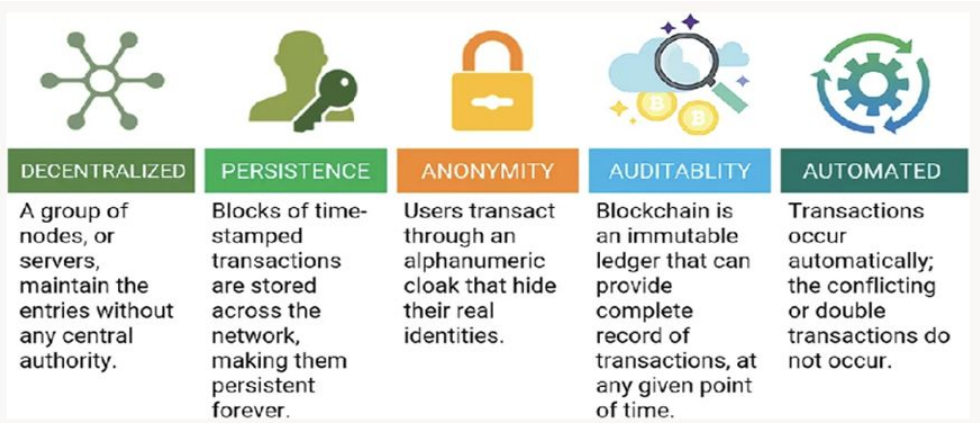
2.2 - Блоки в ланцюгу Blockchain



2.3 - Ілюстрація конкретного блоку Blockchain

Переваги Blockchain

- Децентралізація
- Незмінність
- Анонімність
- Можливість аудиту
- Автоматичність



2.8 - Ключові характеристики блокчейну

Алгоритм консенсусу

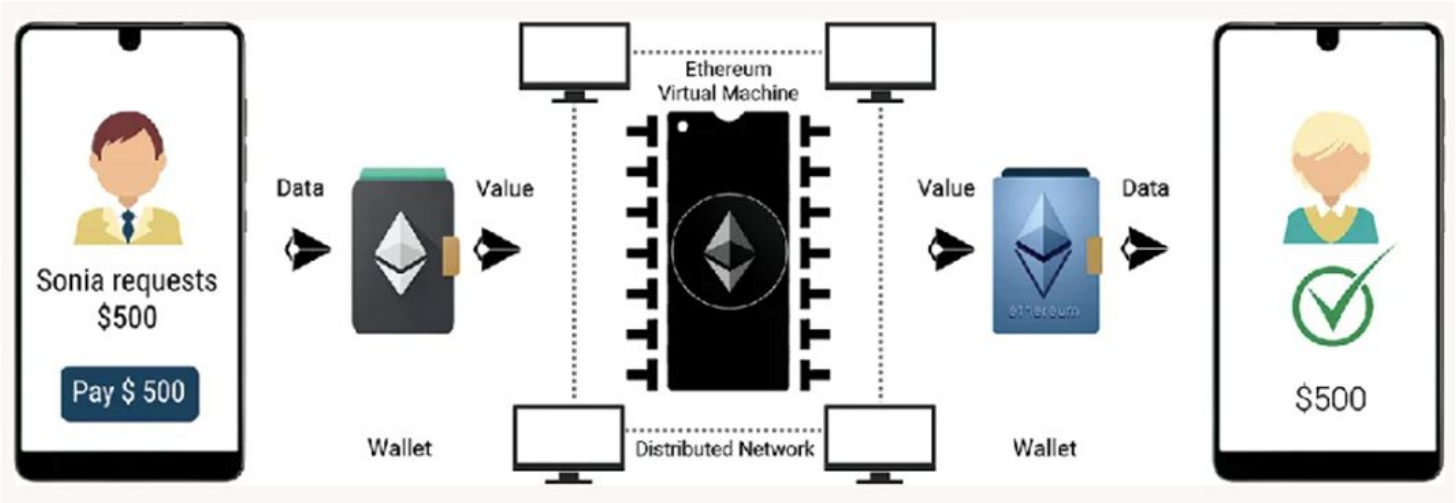
Механізм консенсусу - це відмовостійкий механізм, який використовується для досягнення згоди щодо стану блокчейну з метою забезпечення дійсності та автентичності транзакцій.

Типи алгоритму консенсусу:

- Proof-of-Work (PoW)
- Proof-of-Stake (PoS)
- Delegated Proof-of-Stake (DPoS)

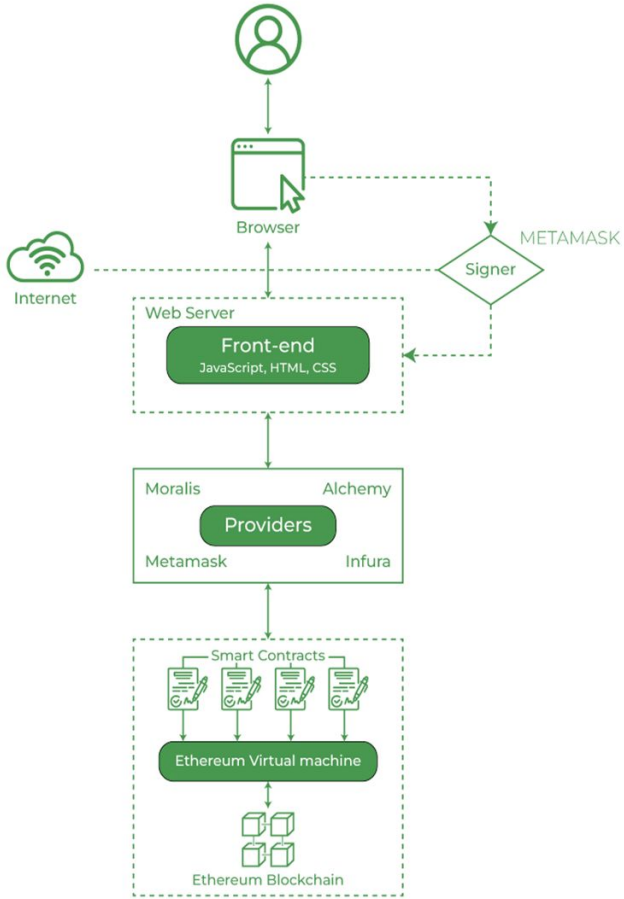
Ethereum та смарт-контракти

Ethereum - це глобально децентралізована обчислювальна інфраструктура з відкритим вихідним кодом, яка виконує програми, що називаються смарт-контрактами. Вона використовує блокчейн для синхронізації та зберігання змін стану системи, а також криптовалюту під назвою ефір для вимірювання та обмеження витрат ресурсів на виконання.



2.10 - Схема роботи мережі Ethereum

Архітектура децентралізованого додатку



Використані технології для розробки

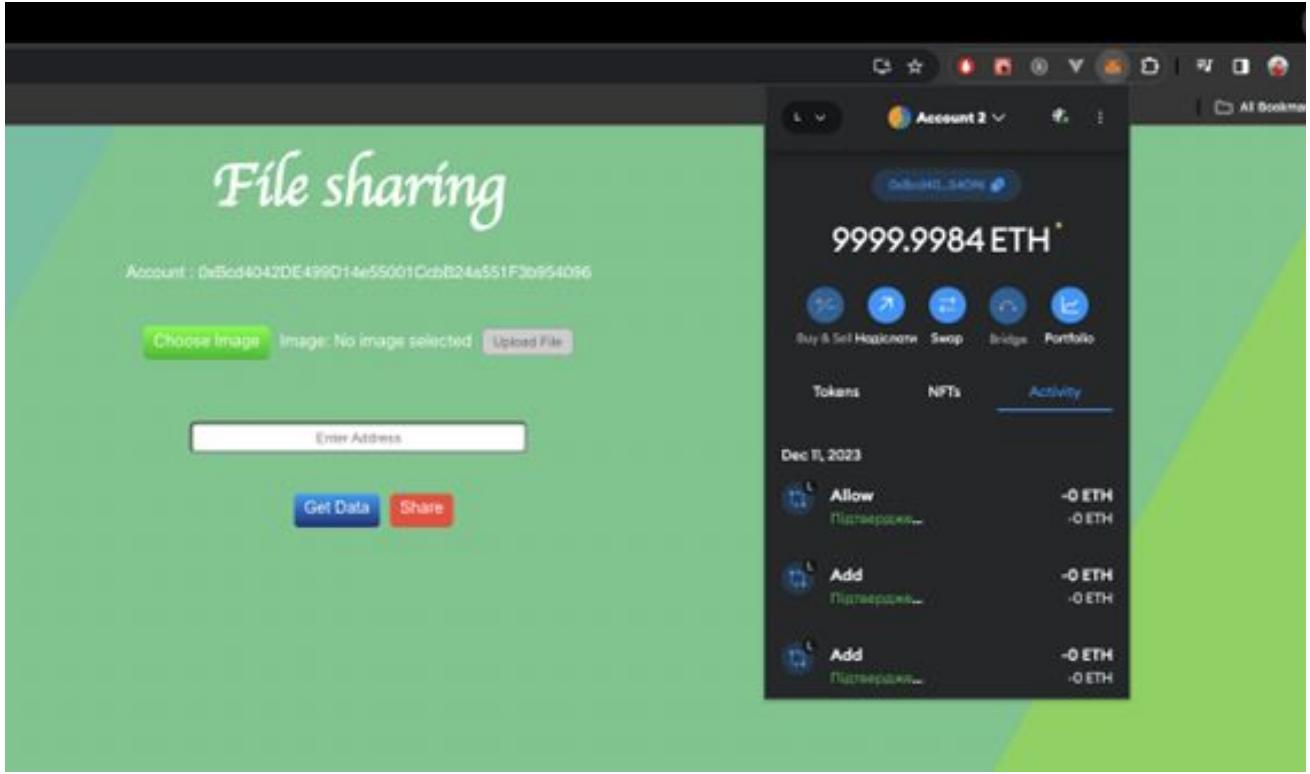
- Frontend:
  - JavaScript
  - React
  - HTML
  - CSS
- Smart-contract:
  - Solidity
  - Ether.js
- Provider-Signer:
  - Metamask

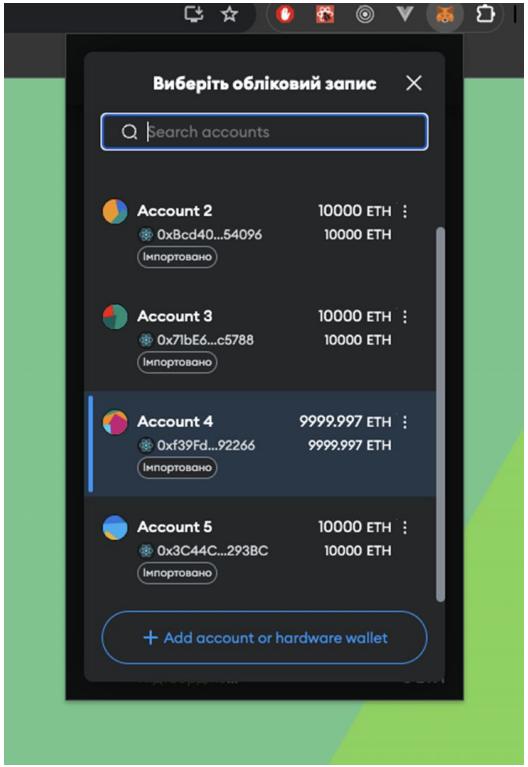
Функції реалізованого додатку

Реалізований функціонал дозволяє:

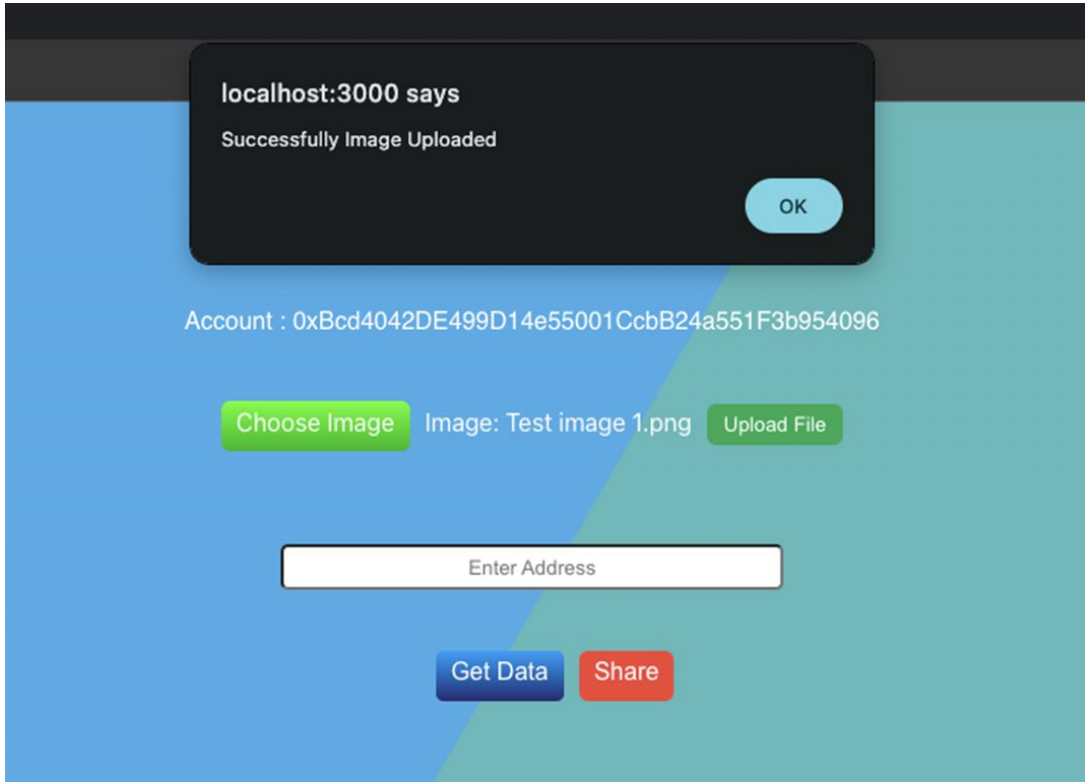
- Отримати доступ до блокчейн мережі через Metamask
- Завантажити файли зображень до блокчейн мережі
- Надавати доступ до своїх зображень іншим користувачам, із іншими аккаунтами в цій мережі
- Отримувати, як свої, так файли зображень із блокчейн мережі, так і ті, для яких надали доступ

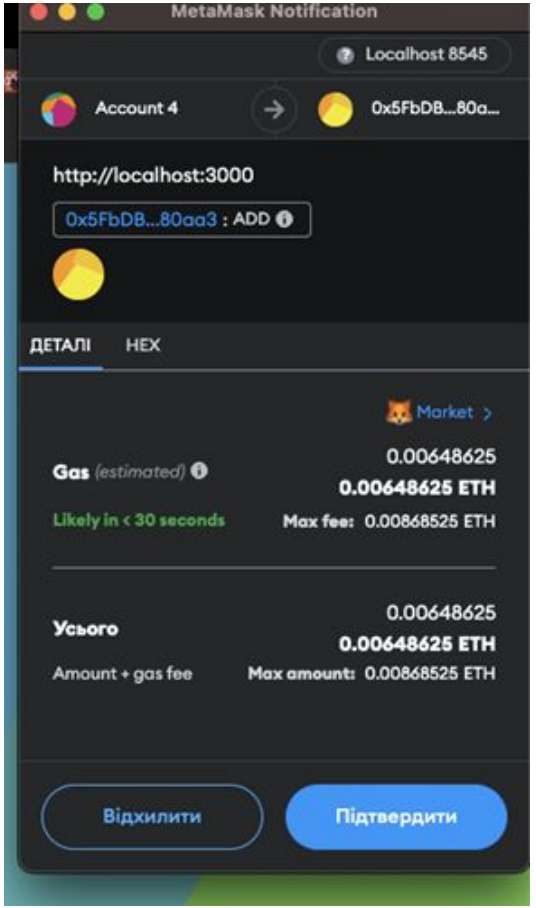
Тестування


File sharing
Account : 0xBcd4042DE499D14e55001CcbB24a551F3b954096
Choose Image
Image: No image selected
Upload File
Enter Address
Get Data
Share
Account 2
9999.9984 ETH
Buy & Sell
Надіслати
Swap
Bridge
Portfolio
Tokens
NFTs
Activity
Dec 11, 2023
Allow
-0 ETH
Add
-0 ETH
Add
-0 ETH

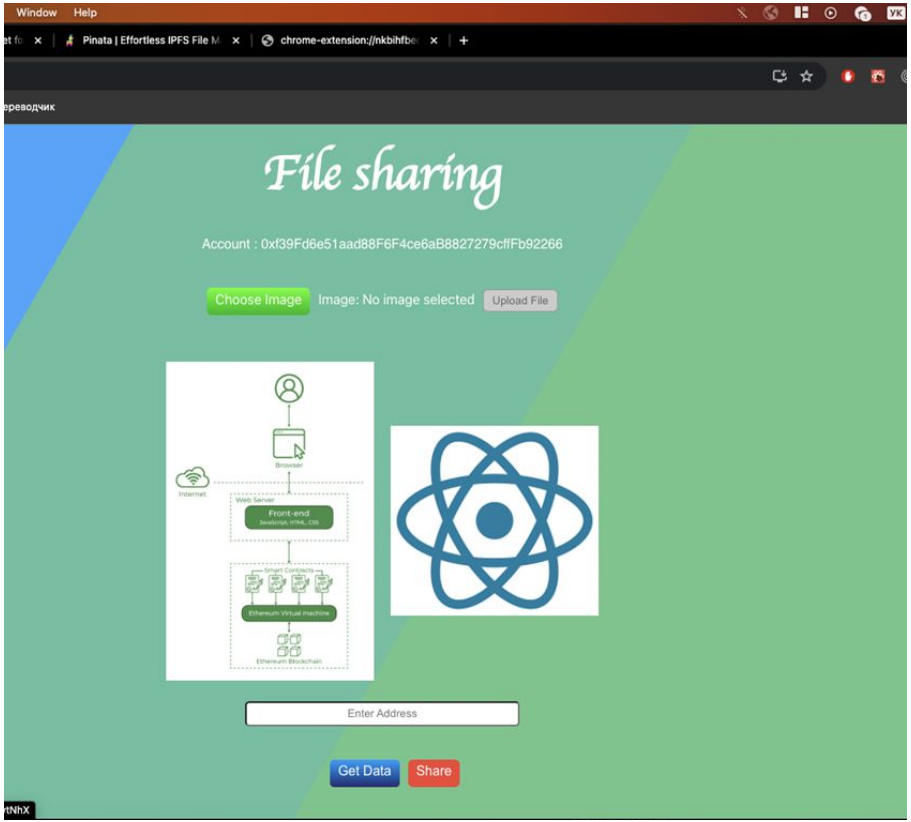

Виберіть обліковий запис
Search accounts
Account 2
10000 ETH
0xBcd40...54096
10000 ETH
Імпортовано
Account 3
10000 ETH
0x71bE6...c5788
10000 ETH
Імпортовано
Account 4
9999.997 ETH
0xf39Fd...92266
9999.997 ETH
Імпортовано
Account 5
10000 ETH
0x3C44C...293BC
10000 ETH
Імпортовано
+ Add account or hardware wallet

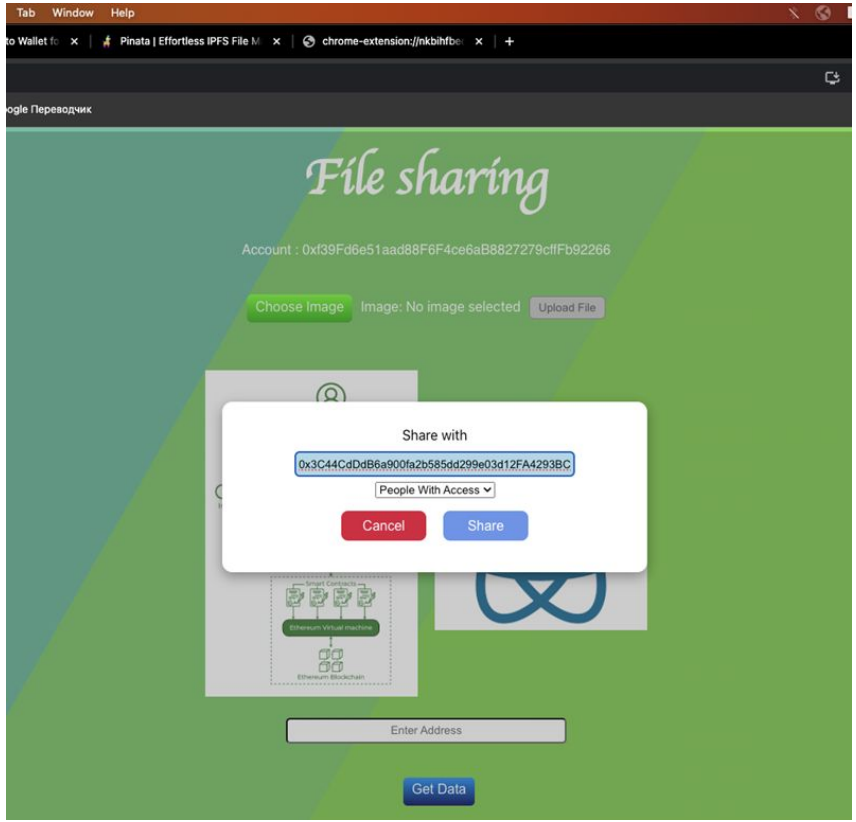
Тестування


localhost:3000 says
Successfully Image Uploaded
OK
Account : 0xBcd4042DE499D14e55001CcbB24a551F3b954096
Choose Image
Image: Test image 1.png
Upload File
Enter Address
Get Data
Share

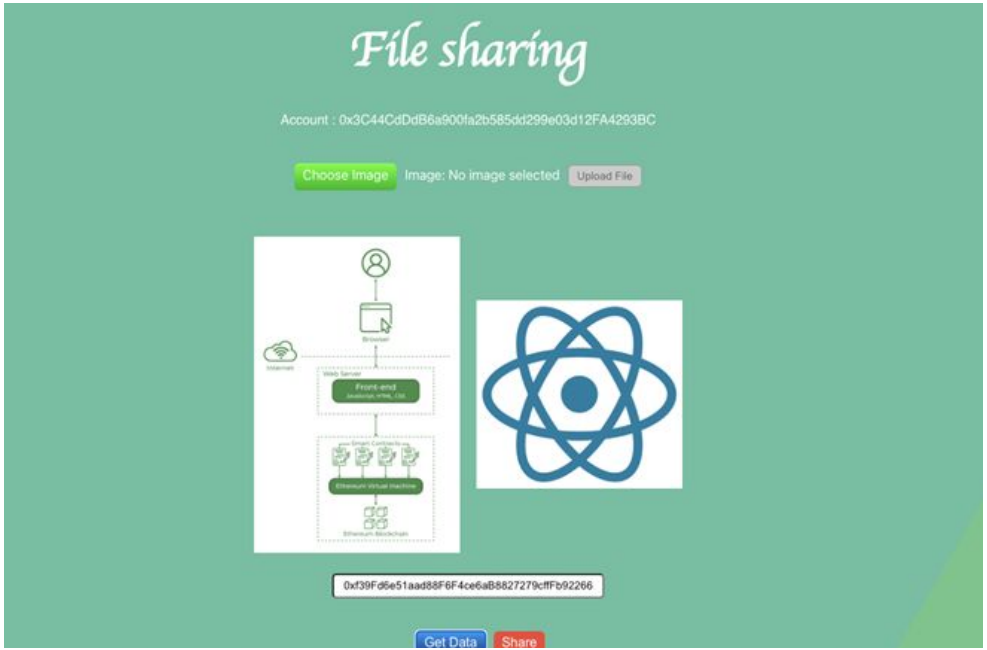

MetaMask Notification
Localhost 8545
Account 4
0x5FbDB...80a...
http://localhost:3000
0x5FbDB...80aa3 : ADD
ДЕТАЛІ
HEX
Market
Gas (estimated)
0.00648625
0.00648625 ETH
Likely in < 30 seconds
Max fee: 0.00868525 ETH
Усього
0.00648625
0.00648625 ETH
Amount + gas fee
Max amount: 0.00868525 ETH
Відхилити
Підтвердити

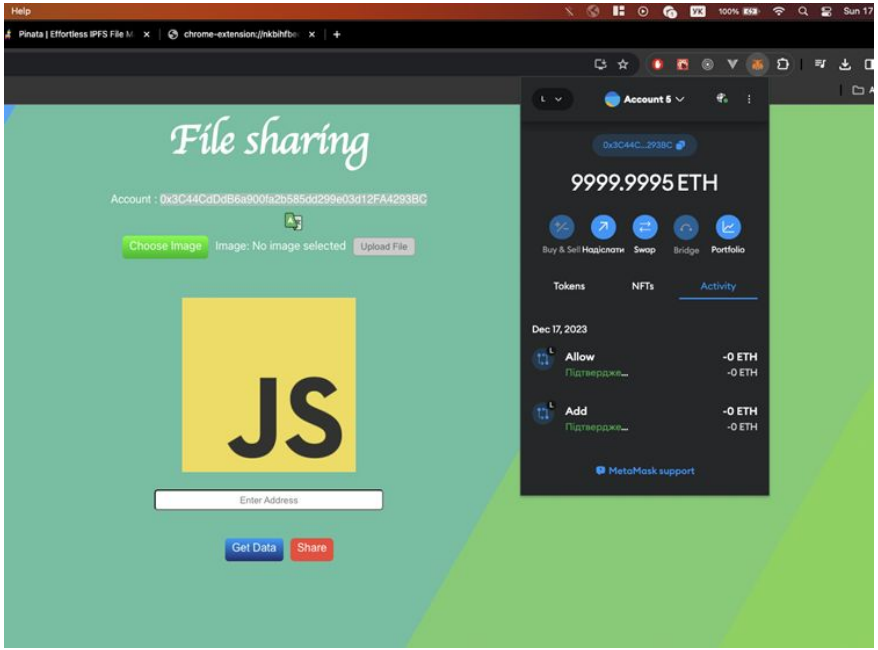
Тестування


Window Help
Pinata | Effortless IPFS File M
chrome-extension://nkbihfbe
File sharing
Account : 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Choose Image
Image: No image selected
Upload File
Browser
Internet
Web Server
Front-end
Ethereum Virtual machine
Ethereum Blockchain
Enter Address
Get Data
Share

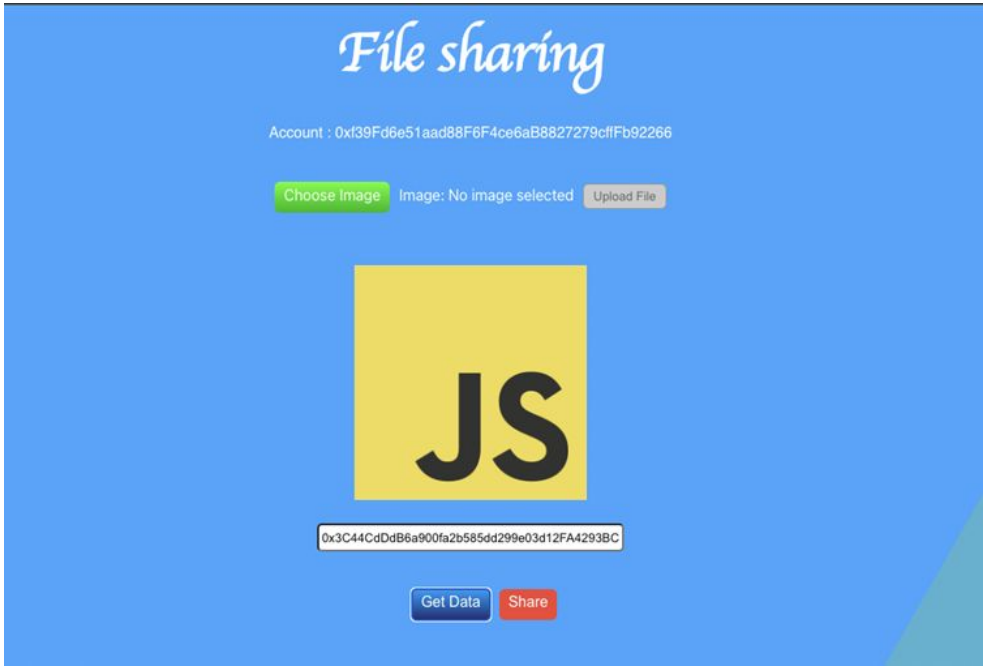

Tab Window Help
Pinata | Effortless IPFS File M
chrome-extension://nkbihfbe
File sharing
Account : 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Choose Image
Image: No image selected
Upload File
Share with
0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
People With Access
Cancel
Share
Ethereum Virtual machine
Ethereum Blockchain
Enter Address
Get Data

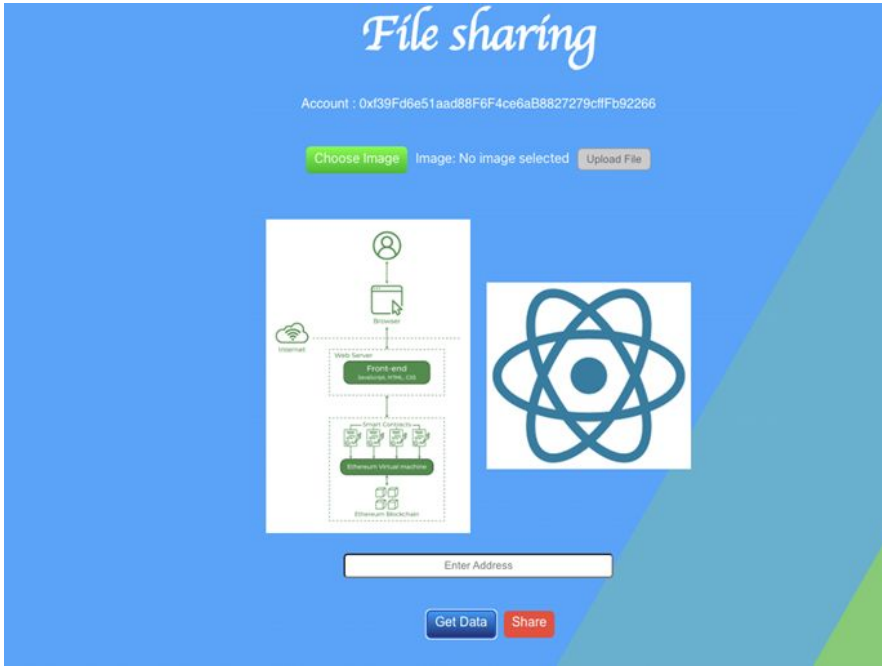
Тестування


File sharing
Account : 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
Choose Image
Image: No image selected
Upload File
0xf39Fd6e51aad88F6F4ce6aB8827279cfFb92266
Get Data
Share


File sharing
Account : 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
Choose Image
Image: No image selected
Upload File
JS
Enter Address
Get Data
Share
Account 5
9999.9995 ETH
Buy & Sell
Swap
Bridge
Portfolio
Tokens
NFTs
Activity
Dec 17, 2023
Allow
-0 ETH
Add
-0 ETH
MetaMask support

Тестування


File sharing
Account : 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Choose Image
Image: No image selected
Upload File
JS
0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC
Get Data
Share


File sharing
Account : 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Choose Image
Image: No image selected
Upload File
Enter Address
Get Data
Share

ВИСНОВКИ

Метою даної дипломної роботи була розробка методу децентралізованого обміну файлами з використанням технології Blockchain з метою підвищення безпеки, ефективності та надійності обміну інформацією між користувачами. Під час її виконання було досліджено та вивчено еволюцію технологій обміну файлами, проведено аналіз технології Blockchain та реалізовано відповідний метод децентралізованого обміну файлами.

У роботі проведено аналіз розвитку технологій для обміну файлами, визначено основні тенденції та характеристики. Розглянуто еволюцію цих технологій, враховано важливі аспекти їхнього використання в сучасному інформаційному середовищі.

Також було вивчено принципи та можливості технології Blockchain, досліджено її вплив на обмін файлами. Зазначено переваги та можливості використання Blockchain для підвищення безпеки та надійності обміну файловою інформацією.

Здійснено розробку та реалізацію методу децентралізованого обміну файлами на основі технології Blockchain. Проведено тестування функціональності, ефективності та безпеки запропонованого методу. Крім того, реалізований додаток, має потенціал для покращень та вдосконалень, наприклад додавання розширених функцій для керування доступом до файлів або додавання інших методів для завантаження файлів.

Отримані результати дозволяють зробити висновок, що розроблений метод може слугувати ефективним рішенням для децентралізованого обміну файлами в різних галузях, де важливо поєднати безпеку та надійність із сучасними технологічними вимогами.