КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра математичної інформатики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

БЛОКЧЕЙН ТЕХНОЛОГІЇ

Виконав студент 4 курсу
Шевченко Максим Олексійович _____
(підпис)

Науковий керівник:
Професор, доктор фіз.-мат. наук
Анісімов Анатолій Васильович _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри математичної інформатики

«____»_____202__р.

протокол №_____

Завідувач кафедри

Терещенко В.М _____
(підпис)

Київ – 2021

Реферат

Обсяг роботи 57 сторінок, 24 ілюстрації, 1 таблиця, 38 джерел посилань.

БЛОКЧЕЙН, БІТКОЇН, КРИПТОВАЛЮТА, РОЗПОДІЛЕНИЙ РЕЄСТР, СМАРТ-КОНТРАКТИ, ETHEREUM, LITECOIN, HASHGRAPH, HOLOCHAIN, DAG, PYTHON

Об’єктом роботи є дослідження технології блокчейн та інших технологій розподіленого реєстру, їх порівняння, а також дослідження структури та принципів роботи й порівняння криптовалют.

Метою роботи є розробка програмного застосунку «Клієнт мережі блокчейн» з метою подальшого впровадження в приватні мережі блокчейн та модифікації відповідно до потреб користувачів.

Методи розроблення: аналіз технологій розподіленого реєстру, принципів роботи мереж блокчейну, криптовалют та смарт-контрактів, порівняння їхніх сильних та слабких сторін, розробка загальної бібліотеки алгоритмів, необхідних для функціонування клієнта мережі блокчейн, розробка прикладів використання алгоритмів, розробка API [6] для застосування та демонстрації можливостей бібліотеки.

Інструменти розроблення. Для розробки реалізації були використані інтерактивні середовища розробки PyCharm [1] Community Edition та Visual Studio Code [2]. Було використано мову програмування Python 3 [4]. Створено чотири файли класів мережі блокчейн та один виконавчий файл, необхідних для функціонування мережі блокчейн та її користувачів. Кожен клас являє собою бібліотеку, яку можливо імпортувати в інші застосунки. Для створення API використано фреймворк Flask [5], а для демонстрації роботи програми – платформу Postman [3].

Результат роботи. Виконано детальний опис технології блокчейн, пояснення принципів її роботи, наведено приклади та сфери застосування. Також було досліджено три інші популярні технології розподіленого реєстру, дано їхній опис та порівняно з блокчейном. Проаналізовано структуру, принципи роботи трьох найпопулярніших криптовалют та їхніх мереж блокчейну, а також проведено порівняльний аналіз із біткоїном. Розроблено програмний застосунок клієнта мережі блокчейн, необхідний для її функціонування. Розроблений застосунок являє собою API та бібліотеки, легкі для розуміння та модифікації. Додаток було викладено в мережу [38] на правах загального доступу та зроблено доступним усім розробникам. Було продемонстровано роботу застосунку та надано інструкцію його використання.

Зміст

ВСТУП

Оцінка сучасного стану об’єкта дослідження. На сьогоднішній день технології блокчейн та криптовалюти стали дуже популярними. Зі зростанням ціни біткоїну почався бум майнінгу та зросла зацікавленість людей у криптовалютах та блокчейні загалом. Наразі найбільш популярним застосуванням блокчейну серед звичайних користувачів є криптовалюти. Особливого визнання серед них здобули Bitcoin, Ethereum, Litecoin та інші. Проте розвиток технології викликав появу нових її застосувань у найрізноманітніших сферах: від ланцюга поставок та фінансів до охорони здоров’я та реклами. Унаслідок люди, особливо IT-спеціалісти, почали задумуватися над створенням власних продуктів на основі блокчейну з метою зберігання незмінності та контролю важливих для них даних.

Актуальність роботи та підстави для її виконання. Застосування, розробка нових та модифікація існуючих технологій розподіленого реєстру має важливе значення зі збільшенням цифровізації інформації у сьогоднішній час. Задля збереження незмінності даних та попередження їх підробки, прискорення взаємодії між користувачами у фінансових сферах доцільно використати блокчейн. Готові рішення для побудови мереж блокчейну полегшують розробникам робочий процес та прискорюють впровадження цих технологій до загальності.

Мета й завдання роботи. Метою дипломної роботи є огляд технології блокчейн, дослідження принципів роботи блокчейн мереж, криптовалют та технологій розподіленого реєстру, їхнє порівняння. Технологіями для порівняння обрано Blockchain [10], DAG [16], Hashgraph [18] і Holochain [17], криптовалютами – Bitcoin [7], Litecoin [21] й Ethereum [20] як найбільш популярні з них. Також завданням є розробка програмного застосунку для користування власною мережею блокчейну. Для досягнення цієї поставлено такі завдання:

- Огляд технології блокчейн
- Огляд та порівняння популярних криптовалют
- Дослідження та порівняння технологій розподіленого реєстру
- Розробка програмного застосунку для взаємодії з блокчейном

Об'єкт, методи й засоби розроблення. Об’єктом порівняльного аналізу є популярні криптовалюти та технології розподіленого реєстру. Об’єктом розробки є програмна реалізація клієнта мережі блокчейн, яка може бути використана, модифікована та встановлена будь-яким розробником через імпортування необхідних модулів та використанням API.

Можливі сфери застосування. Дана реалізація доступна у вигляді розширюваної бібліотеки та API для вбудовування у будь-які додатки на розсуд розробника. Розроблені модулі являють кістяк мережі блокчейн із реалізованим

базовим клієнтським функціоналом та допомагають спростити розробку власних мереж блокчейну для подальшого локального застосування.

РОЗДІЛ 1 БЛОКЧЕЙН

1.1 Визначення блокчейну

Блокчейн, тобто ланцюг блоків (англ. Blockchain, Block chain від block – блок, chain – ланцюг) – це децентралізована база даних, зростаючий список записів, які називаються блоками, пов'язаних між собою за допомогою криптографії; спільний, незмінний реєстр, призначений для запису транзакцій, обліку активів і побудови довірчих відносин [9]. У кожному блоці зберігається криптографічний хеш попереднього блоку, мітка часу створення блоку та транзакції (в основному утворюють дерево Меркла [27]) (рис. 1.1). За задумом блокчейн стійкий до зміни даних, які в ньому зберігаються. Це зумовлене тим, що при зміні даних змінюється значення хешу блока, що викликає конфлікт у ланцюгу.

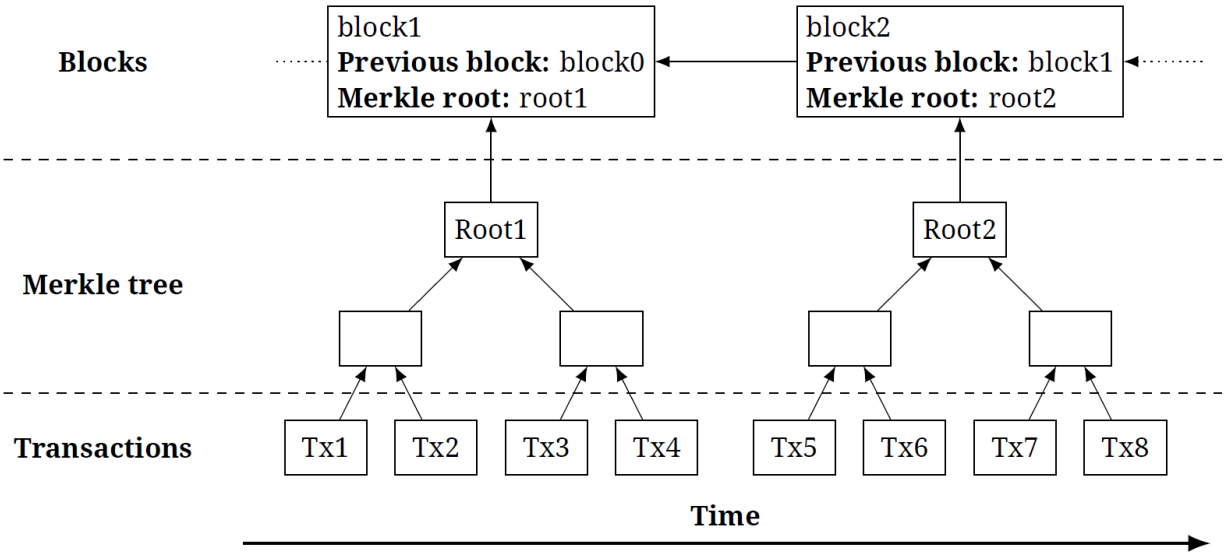

Рисунок 1.1 – Структура блокчейну

Можна стверджувати, що блокчейном є технологія децентралізованого незалежного паралельного виконання алгоритму з наступною верифікацією результатів виконання алгоритмом консенсусу і синхронізацією результатів між усіма елементами мережі блокчейну. Ідеальним з точки зору визначення наведеного вище є блокчейн біткоїнів, проте IT індустрія почала створювати альтернативні рішення, засновані на технології розподіленого реєстру (розділ 3).

Коли заходить мова про пошук сприятливих можливостей для впровадження блокчейн-технології з метою підвищити цінність організації, часто виникає питання "Де саме технологія блокчейну може виявитися корисною і чим вона відрізняється від вже існуючих технологій?"

Блокчейн-ланцюги являють собою особливий тип баз даних, тому їх можна використовувати у всіх випадках, в яких можуть використовуватися будь-які інші види баз даних. Однак може виявитися і так, що немає ніякого сенсу заново долати труднощі і нести додаткові витрати там, де звичайна база даних і так прекрасно справляється з поставленим завданням.

Але якщо вам доводиться ділитися важливою інформацією з іншими сторонам, яким ви не цілком довіряєте, або якщо ваші дані потребують аудиту, або існує ризик, що дані будуть порушені або підмінені зсередини або ззовні, ви дійсно зможете оцінити всі переваги використання блокчейну певного типу.

Завдяки своїй зрозумілості та простоті реалізації технологія блокчейну використовується в найрізноманітніших сферах життя [9]:

- ланцюжки поставок
- охорона здоров’я
- державний сектор
- роздрібна торгівля
- ЗМІ та реклама
- нафтогазова галузь
- телекомунікації
- страхування
- фінансові послуги
- подорожі та транспорт

1.2 Історія створення блокчейну

У 1982 р. побачила світ дисертація «Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups» криптографа Девіда Чауму, в якій він описав блокчейн-подібний протокол. Надалі над криптографічно захищеним ланцюжком працювали Стюарт Хабер та В. Скотт Сторнетт. Робота була пов’язана із створенням системи для збереження непідробності позначки часу в документах. У 1992 році Хабер, Сторнетт та Дейв Байєр додали до системи дерево Меркла, що дозволило збирати документи в один блок.

Перший блокчейн був впроваджений під псевдонімом Сатоші Накамото в 2008 році [7] для використання в криптовалюті біткоїн. Використання блокчейну вирішило проблему подвійних витрат, при цьому не використовуючи центральний довірений орган влади. У дизайні мережі Накамото використав метод, схожий на Hashcash та ввів динамічний параметр

складності генерації блоків. Повноцінна реалізація вийшла в наступному році основним компонентом мережі біткоїн, де блокчейн використовується як реєстр транзакцій.

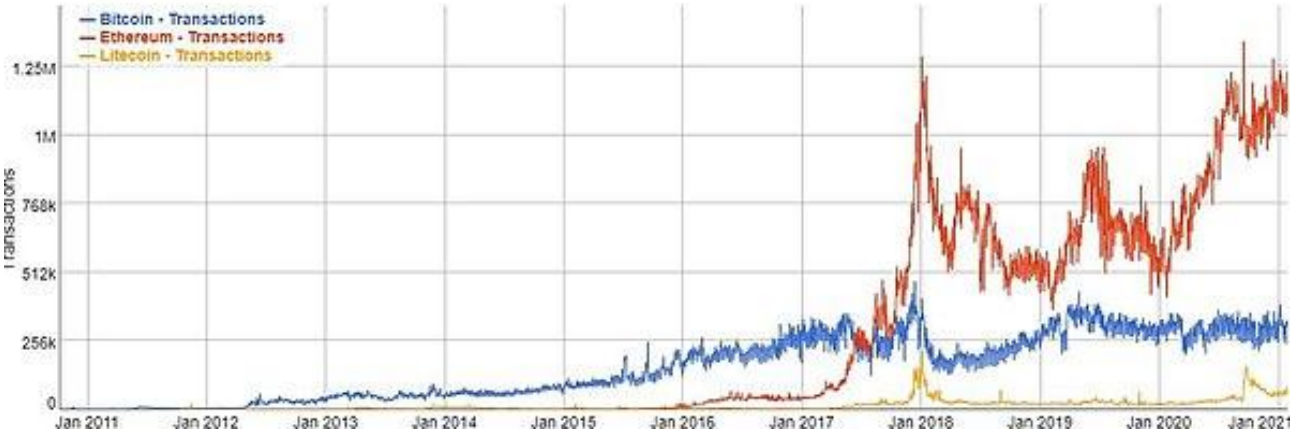

Рисунок 1.2 – Добова кількість транзакцій у мережах Bitcoin, Ethereum і Litecoin

1.3 Структура блокчейну

1.3.1 Загальний опис

Блокчейн – це децентралізований, розподілений та часто загальнодоступний цифровий реєстр, до якого вносяться транзакції, згруповані в блоки. Ланцюг блоків зберігається на комп'ютері кожного з клієнтів. Це дозволяє учасникам валідувати транзакції самостійно, без третьої сторони, не сплачуючи комісію третім особам. Розподіленого реєстр блокчейну використовується в одноранговій мережі, учасники якої дотримуються одного протоколу зв'язку та валідації блоків транзакцій і блоків. Хоча записи блокчейну не є незмінними, блокчейн вважається захищеними за задумом. Операції підтверджуються масовою співпрацею, що зумовлена колективними власними інтересами. Завдяки блокчейну неможливо нескінченно використовувати цифровий актив, адже транзакції підтверджуються тільки після додавання до блоку. Це гарантує, що кожна одиниця ресурсу була передана лише один раз, тобто вирішує проблему подвійних витрат. Блокчейн слугує протоколом обміну цінностями (певними ресурсами). Він підтримує права власності, бо транзакцію можна деталізувати до вигляду контракту.

1.3.2 Логічна структура та розподіл ролей

Логічна структура блокчейну можна утворюється з наступних шарів:

- апаратне забезпечення мережі
- власне мережа (вузли, якими поширюється інформація)
- алгоритми консенсусу
- інформація (транзакції), збережена в блоки

- програмний застосунок

Учасники мережі блокчейн відіграють певну роль у її функціонуванні. Ролі можна поділити на:

1. користувач блокчейну: учасник (як правило, діловий користувач) з дозволами на приєднання до мережі блокчейну та проведення транзакції з іншими учасниками мережі. Блокчейн технологія працює у фоновому режимі, тому блокчейн користувач не знає про це. Зазвичай буває кілька користувачів у будь-якій одній бізнес-мережі.
2. регулятор: користувач блокчейну зі спеціальними дозволами контролювати транзакції, що відбуваються в мережі. Регуляторам може бути заборонено проводити операції.
3. розробник блокчейну: програмісти, які створюють додатки та інтелектуальні контракти, що дозволяють блокчейн користувачам проводити транзакції в мережі блокчейну. Додатки служать каналом між користувачами та блокчейном.
4. оператор мережі блокчейн: фізичні особи, які мають спеціальні дозволи та повноваження для визначення, створення, управління, та відстежування мережу блокчейну. Кожен бізнес, який користується мережею блокчейну, має оператора мережі блокчейн.
5. традиційні платформи обробки: існуючі комп’ютерні системи, які блокчейн може використовувати для збільшення швидкості обробки. Цій системі також може знадобитися ініціювати запити у блокчейні.
6. традиційні джерела даних: існуючі системи даних, які можуть надавати дані для впливу на поведінку смарт-контрактів і допомагають визначити спосіб спілкування та передачі даних, який відбуватиметься між традиційними програмами / даними та блокчейном – через виклики API, за допомогою хмарних повідомлень у стилі MQ, або обидва.
7. орган сертифікації: особа, яка видає та управляє різними типами сертифікатів, необхідних для запуску дозволеного блокчейну. Наприклад, може знадобитися видавати сертифікати користувачам блокчейну або окремим транзакціям.

1.3.3 Блоки

Блоки складаються з набору підтверджених транзакцій, хеші яких утворюють дерево Меркла. У кожному блоці також зберігається криптографічний хеш попереднього блоку, що пов'язує їх між собою. Такі блоки поєднуються в ланцюг. Завдяки цьому можна перевірити цілісність ланцюга, пройшовши по ньому з кінця до початку.

Часом виникає ситуація, що різні блоки створюються одночасно, що породжує тимчасове розгалудження в ланцюгу (рис. 1.3). У такому випадку дійсною вважається тільки найдовша послідовність блоків. Не включені до

головного ланцюга блоки називають блоками-сиротами. Користувачі блокчейну можуть мати різні версії ланцюга, адже зберігають лише відому їм комбінацію блоків. Кожного разу, коли користувач одержує довшу коректну версію ланцюга (як правило, стару версію, розширену новим блоком), вони розширюють або перезаписують ланцюг і передають вдосконалення іншим користувачам. Блокчейн надає перевагу розширенню новими блоками, а не перезапису старих. Звідси достовірність даних у блоці тим вища, чим більше нових блоків набудовано поверх нього.

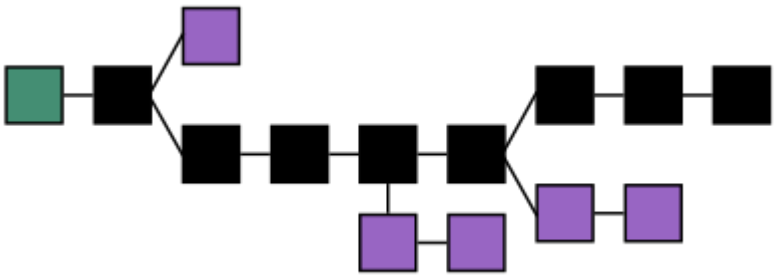

Рисунок 1.3 – Ланцюг блокчейну. Головний ланцюг (чорний) складається з найдовшої серії блоків, починаючи з генезисного (зелений). Блоки-сироти (фіолетові) не включені до основного ланцюга.

Під час майнінгу блоку включені в нього транзакції стають підтвердженими. У мережі блокчейн задається середній час створення нових блоків. У деяких мережах цей час складає декілька секунд, у деяких – хвилини. Час додавання блоків в Ethereum перебуває в межах від 14 до 15 секунд, тоді як у мережі біткоїн він становить в середньому 10 хвилин.

Жорсткі розгалудження, хард форк (англ. hard forks, дослівно жорсткі виделки) – це зміна правила перевірки блоків, при якому застаріле програмне забезпечення розпізнаватиме блоки, створені за новими правилами, як недійсні. Якщо оновлення правил перевірки виконають не всі вузли мережі, то виникне постійний розкол. Наприклад, Ethereum роздвоївся на мережі Ethereum та Ethereum Classic, щоб зберегти інвесторів після взлому через вразливість у коді. Натомість цій стратегії, біткоїн запобіг форку 2013 року відкатом прогамного забезпечення з оновлених вузлів на старе, але в 2017 році все одно відбулося розгалудження мережі на класичний біткоїн та Bitcoin Cash. Це розгалудження мережі було зроблене для реалізації різних алгоритмів підвищення швидкості транзакцій.

1.3.4 Децентралізація мережі

У блокчейні дані зберігаються в одноранговій (peer-to-peer) мережі, чим нівелюються ризики централізованого утриманням даних, адже відсутня

централізована точка вразливості, якою могли б скористатися хакери. Так само не існує певної точки відмови.

У блокчейні використовується криптографія з відкритими ключеми для підвищення рівня анонімності та безпеки. Відкритий ключ слугує адресою користувача. Він являє собою довгу, випадкову комбінацію символів. Приватний ключ схожий на пароль, який надає його власнику доступ до його цифрових активів та засобів взаємодії з реалізованими в конкретній мережі можливостями.

Копія ланцюга зберігається в кожному вузлі мережі. Достовірність даних підтримується масовою реплікацією баз даних та обчислювальною довірою. Офіційної головної версії ланцюга не існує, тому ланцюги кожного користувача піддаються перевірці. Транзакції транслюються в мережу за допомогою програмного забезпечення. Повідомлення доставляються максимально ефективно. При майнінгу блоку вузли перевіряють коректність транзакцій, підтверджують їх, додаючи до нового блоку, після чого надсилають змайнений блок сусіднім вузлам.

1.4 Типи мереж блокчейну

1.4.1 Публічний блокчейн

Публічні мережі зазвичай дуже великі і децентралізовані. Будь-хто може взяти участь в їх роботі і на будь-якому рівні – підтримувати роботу повного вузла, майнити криптовалюту, торгувати токенами або записувати нові дані. Ці мережі зазвичай краще забезпечують свій захист і незмінюваність, ніж приватні або ексклюзивні мережі. Однак працюють вони, як правило, повільніше та їх експлуатація обходиться дорожче. Вони маніпулюють захищеною криптовалютою і мають суворі обмеження на розмір даних, які зберігаються в записі. Для створення блоків у цих мережех використовується певна варіація алгоритмів на кшталт Proof of Stake або Proof of Work. Користувачу, який створив новий блок, надається винагорода. Яскравими представниками публічних блокчейнів є біткоїн й Ethereum.

Перевагою публічної мережі блокчейну є те, що в ній реалізований захист від зловмисників у самій структурі мережі, звідси не потрібно контролювати права доступу до неї. На сьогодні в криптовалютах використовується принцип підтвердження роботи для створення нових записів. Це надає мережі додаткового захисту. Наприклад, для підтвердження блоків у біткоїні використовується задача Hashcash.

Аналіз публічних мереж стає необхідним із розвитком біткоїнів, Ethereum, litecoin та криптовалют загалом. Проблема доступу до даних важлива не тільки для криптовалют, а й для криптобірж і банків. Криптовалюти часто

звинувачують у тому, що блокчейн сприяє незаконній торгівлі забороненими товарами та послугами на чорному ринку. Це викликано тим, що багато людей вважають криптовалюти приватними та такими, які неможливо відстежити. У публічному блокчейні доступ кожному зареєстрованому користувачеві надається повний доступ до всіх даних ланцюга. Отже користувачі за бажанням можуть проаналізувати та дослідити всі транзакції кожного гаманця. Крім того, можна скористатися послугами спеціалізованих IT-компаній, які займаються відстеженням блокчейнів. Все це гарантує прозорість криптовалютного фонду для правоохоронних органів та фінансових структур. Звісно, злочинці можуть почати користуватися однією з нових криптовалют, яких стає все більше, та яка не гарантує прозорості даних. Це питання є головним щодо повномасштабного впровадженні технологій блокчейну до фінансового сектору держав та світу загалом.

1.4.2 Приватний блокчейн

Приватні мережі спільно використовуються довіреними сторонами і можуть бути повністю невидимими для широкого загалу. Вони, як правило, надзвичайно швидкі при повній відсутності часу очікування виконання транзакції. Крім того, вартість їх експлуатації зазвичай невелика, а розгорнути їх можна за короткий термін. Більшість приватних мереж не використовує криптовалюту й не забезпечує тієї незмінності та захищеності, які характерні для децентралізованих мереж. Обсяг даних, які зберігаються в записі, може бути необмеженим.

У противагу публічним мережам, валідація приватних мереж блокчейнів відбувається власниками мережі. Для створення блоків та підтвердження операцій використовуються певні загальновідомі користувачам вузли мережі. При правильній реалізації мережі приватного блокчейну можуть бути децентралізованими та вимагати права доступу від користувачів, на противагу блокчейнам без дозволів.

Цікавою є поведінка приватних мереж перед атакою 51%. У цьому типі мережі контролюються всі ресурси майнінгу блоків. Тобто замість того, щоб намагатися перевершити обчислювальну потужність мережі, необхідно хакнути корпоративний сервер та пошкодити засоби створення блоків. Або ж можна взяти їх під контроль на та власну забаганку здійснювати контроль над усією мережею та всіма клієнтами. Питання захисту даних у приватному блокчейні дуже значне. Власники мережі, в основному, не бажають витрачати великі кошти та ресурси для підтримки малої кількості користувачів. Крім того, відсутність винагороди за майнінг не спонукає власників збільшувати обчислювальну потужність мережі, що призводить до її повільної роботи та неефективності.

1.4.3 Ексклюзивний блокчейн

Ексклюзивні мережі відкриті для широкої публіки, але участь в їхній роботі контролюється. Багато з них використовують криптовалюту, але при цьому за програми, що працюють з мережею, може стягуватися невелика плата. Цей підхід спрощує масштабування проекту й підвищення обсягу транзакцій. Ексклюзивні мережі можуть працювати дуже швидко при малому часі підтвердження транзакцій і зазвичай допускають більший обсяг збережених в одному записі даних в порівнянні з публічними мережами.

1.4.4 Гібридний блокчейн

Також можуть існувати гібридні варіанти між трьома основними типами блокчейн-мереж. Мета такого підходу – знайти необхідний баланс між захищеністю, контрольованістю, можливостями масштабування і збереження даних в записах для додатків, що функціонують поверх цих мереж. У гібридному блокчейні за задумом мають використовуватися найкращі переваги як централізованих, так і децентралізованих мереж. Конкретна структура мережі, алгоритмів роботи та обміну даними може відрізнятися в різних гібридних мережах у залежності від програмної реалізації застосунку.

1.4.5 Бічні ланцюги

Бічним ланцюгом (sidechain) називають відокремлений ланцюг блокчейну, який працює паралельно з головним ланцюгом. Бічний ланцюг може бути пов’язаним із записами головного блокчейну. Первинний ланцюг у свою чергу теж може мати доступ до бічного ланцюга. Завдяки цьому можлива робота обох версій блокчейну паралельно. Записи в таких мережах в переважній більшості являють цифрову валюту. У такому випадку незалежна робота ланцюгів виконується використанням різних алгоритмів консенсусу, зміною структури блоків або способів збереження записів у блоках.

1.4.6 Блокчейн-консорціум

Відповідальність за адміністрування блокчейну може лежати на кількох організаціях. Ці заздалегідь обрані організації встановлюють права доступу для виконання транзакцій або доступу до даних. Блокчейн-консорціум є ідеальним рішенням для компаній, коли всі учасники мають дозволи і несуть колективну відповідальність за блокчейн.

РОЗДІЛ 2 КРИПТОВАЛЮТИ

2.1 Bitcoin

Біткоїн – це криптовалюта, винайдена в 2008 році невідомою особою або групою людей, що використовують ім'я Сатоші Накамото. Валюта почала використовуватись у 2009 р., коли її впровадження було випущено як програмне забезпечення з відкритим кодом. Біткоїн – це децентралізована цифрова валюта без центрального банку або єдиного адміністратора, яка може передаватися від користувача до користувача в одноранговій мережі біткоїнів без необхідності посередників. Транзакції перевіряються мережевими вузлами за допомогою криптографії та реєструються у відкритому розподіленому реєстрі, який називається блокчейн.

Біткоїни створюються як винагорода за процес, відомий як майнінг. Їх можна обміняти на інші валюти, товари та послуги, але реальна вартість монет надзвичайно нестабільна. Дослідження, проведене Кембриджським університетом, підрахували, що в 2017 році було від 2,9 до 5,8 мільйона унікальних користувачів, які використовують гаманець криптовалют, більшість із них використовують біткоїн. Користувачі використовують цифрову валюту з ряду причин: такі ідеології, як прихильність анархізму, децентралізація та лібертаріанство, зручність, використання валюти як інвестиції та псевдонім транзакцій. Поширене використання призвело до прагнення урядів до регулювання з метою оподаткування, сприяння легальному використанню в торгівлі та з інших причин (наприклад, розслідування щодо відмивання грошей та маніпуляцій з цінами).

2.1.1 Структура блоку

Блок біткоїна – структурна одиниця ланцюга, як містить в собі перелік записів про виконані транзакції в криптовалютній мережі. Блок містить тільки ті дані, які не знаходяться в попередніх блоках. Нові блоки додаються в кінець ланцюга. У кожному блоці зберігається криптографічний хеш попереднього блоку, що робить неможливим внесення змін до даних блокчейну без порушення цілісності ланцюга.

У блоці знаходяться записи про транзакції та про минулий блок. Кожен блок містить спеціальну змінну – доказ роботи, яка підбирається таким чином, щоб заголовок хешу блоку був менший певного ліміту. Підбір цього значення відбувається випадковим чином та займає багато часу, у середньому 10 хвилин. Цим процесом займаються майнери, використовуючи відеокарти. За свою роботу вони отримують винагороду.

При знаходженні рішення майнер надсилає його всім іншим вузлам. Вони перевіряють знайдений доказ роботи та в разі його правильності додають

блок до свого ланцюга. Складність пошуку залежить від кількості можливих комбінацій заголовку. При збільшенні їхньої кількості зростає необхідна обчислювальна потужність всієї мережі, що призводить до підвищення часу пошуку рішення. Складність змінюється кожні 2016 блоків, що становить рівно два тижня при середньому часі майнінга в 10 хвили.Варто зазначити, що складність видобутку намагаються тримати на такому рівні, щоб забезпечити стабільний час видобутку блоків.

Так як майнери працюють паралельно, часом виникає ситуація, що одночасно знаходять два різних рішення. Унаслідок цього виникає розгалудження головного ланцюга блокчейну. Проте, продовжуючи роботу мережі, одне з розгалудження зростатиме швидше й стане продовженням головного ланцюга. Транзакції, які потрапили до коротких ланцюгів, стають не підтвердженими та потребують нового включення в блок.

Кожен блок біткоїну складається із заголовка та переліку транзакцій. Заголовок містить наступні параметри:

1. **Hash** – хеш блоку, обчислений хеш-функцією SHA-256. Як зазначалося вище, процес майнінгу полягає в підборі параметрів, щоб значення хешу задовольняло деякій умові. Як наслідок отримуємо стабільний час обрахунку хеша. Транзакції, які зберігаються в блоці, не впливають напряму на значення хешу, а отже він не залежить від їхньої кількості та змісту.
2. **Prev_block** – значення хешу попереднього блоку. Ця змінна потрібна для забезпечення цілісності ланцюга. Завдяки їй можна перевірити, чи вносилися зміни в якийсь блок. Щоб підробка не була виявлена, довелося б заново рахувати значення доказу роботи для всіх блоків, якій ідуть після зміненого.
3. **Version** – версія схеми блоку
4. **Time** – змінна, яка вказує період часу видобутку нового блоку. Сатоші Накамото розробив систему так, щоб останній блок видобувся в 2016 році.
5. **Mrkl_root** – корінь дерева Меркла, утвореного з криптографічних хешів транзакцій. Використовується для того, щоб зробити хеш блоку залежним від вмісту транзакцій для того, щоб неможливо було змінити дані в якійсь транзакції без зміни структури ланцюга.
6. **Bits** – скорочений варіант значення хешу. Використовується при додаванні нових блоків: хеш наступного блоку має бути меншим за цей параметр. Bits напряму впливає на складність видобутку блоків. Його оновлення відбувається кожні чотирнадцять днів. Для розрахунку цього параметру порівнюється кількість створених за період часу блоків із бажаним значення. Чим більше видобуто блоків, тим важче видобути

наступний. Це дає змогу системі автоматично підлаштовуватися під зміну потужностей майнерів.

7. **N_tx** – кількість транзакцій у блоці.
8. **Nonce** – параметр доказу роботи. Перебирається випадковим чином, допоки значення хешу перевищує bits. Значення хешу блока змінюється при зміні довільних його даних чи параметрів: минулий хеш змінюється при появі нового елемента в ланцюгу, корінь дерева Меркла оновлюється при включенні нової транзакції, час – щосекунди, bits – що два тижня. Параметр nonce застосовується, щоб хеш блоку постійно задовольняв умові коректності.
9. **Size** – розмір блоку в байтах.

2.1.2 Структура транзакцій

Наступним елементом блоку є транзакції. Вони зберігаються в блоці у вигляді списку. У кожній транзакції обов’язково вказується відправник та отримувач криптовалюти.

Отримувач вказується за його публічним ключем. Щоб довести володіння потрібною сумою переказу, відправник вказує перелік зарахувань коштів на його рахунок. Щоб довести істиність транзакції, відправник використовує цифровий підпис, який може перевірити отримувач. Структура транзакцій зображена на рис. 2.1.

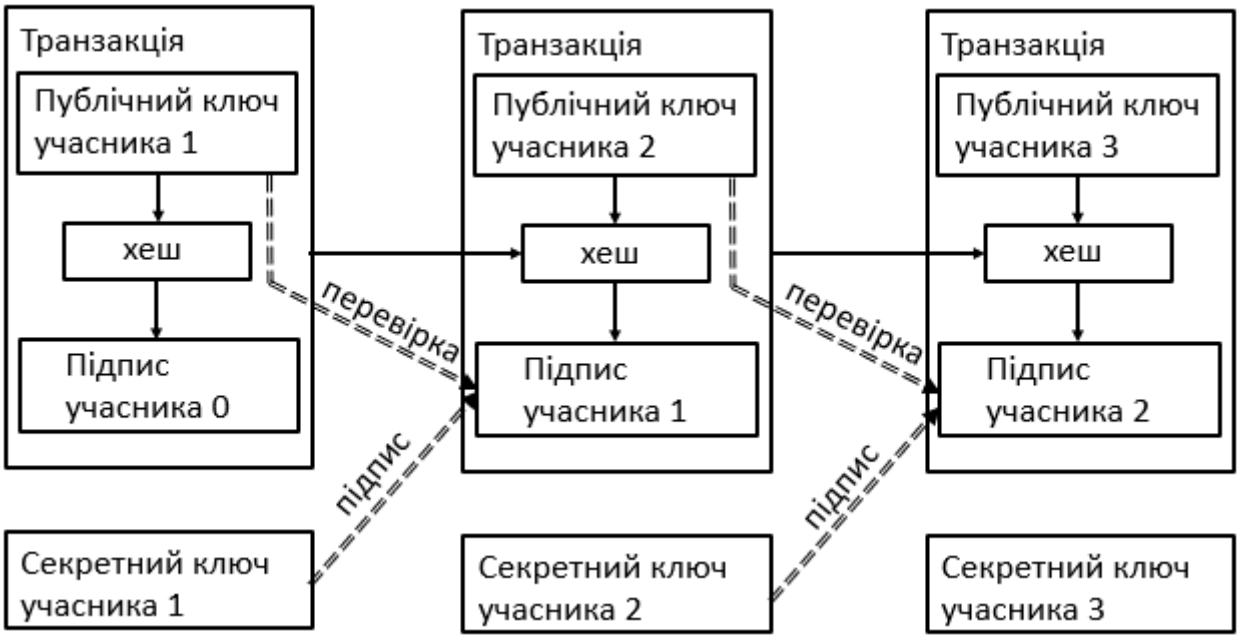

Рисунок 2.1 – Структура транзакцій у мережі біткоїн

Параметри списку транзакцій такі:

1. **Hash** – криптографічний хеш транзакції. Він також використовується в дереві Меркла та хеші блока, в якому зберігається дана транзакція. Між транзакціями існує зв'язок, подібний до самого ланцюга блоків – у кожній наступній транзакції зберігається хеш попередньої. Тобто при зміні даних операції відбудеться руйнування цілісності списку транзакцій. Крім того, порушиться структура дерева Меркла, що дасть змогу виявити підробку.
2. **Vout_sz** – кількість адрес, на які кошти будуть надіслані.
3. **Vin_sz** – кількість операцій зарахування коштів відправникам.
4. **Ver** – версія структури.
5. **Size** – розмір транзакції в байтах.
6. **Loch_time** – поки що не реалізований параметр відстрочки підтвердження транзакції. Вказує кількість блоків, через які операція буде включена до блокчейну та підтверджена. Параметр було введено для можливості скасування операцій чи додаткової її обробки. Поки його значення нульове.
7. **In** – вказує транзакції зарахування коштів відправникам. Використовується для підтвердження права володіння криптовалютою. Включає перелік транзакцій зарахування коштів відправнику, вказуючи їхній хеш, індекс та підпис.
8. **Out** – список витрат відправників. Вказує кількість витрачених коштів та публічний та скрипт перевірки валідності. Витрати не можуть перевищувати дохід відправника. Кількість грошей зазначається в елементарних частинах біткоїну, щоб не користуватися дробовими числами. Скрипт являє собою хеш публічного ключа та функцію, яка використовується для підтвердження коректності транзакції.

2.2 Ethereum

Ethereum (від англ. Ether – «етер»), Етеріум, (часто просто «етер» або «ефір») – це децентралізований блокчейн з відкритим кодом із функціоналом смарт-контракту. Ефір (ETH) – це криптовалюта платформи. Це друга за величиною криптовалюта за ринковою капіталізацією після Біткоїна . Ефіріум є найбільш активно використовуваним блокчейном.

Ethereum був запропонований в 2013 році програмістом Віталіком Бутеріном . Розвиток фінансувався за рахунок краудфандингу в 2014 році, а мережа розпочала роботу 30 липня 2015 року, початковий запас склав 72 мільйони монет. Віртуальна машина Ethereum (EVM) може виконувати сценарії та запускати децентралізовані програми. Ефіріум використовується для децентралізованого фінансування, створення та обміну NFT і використовується для обміну монет.

У 2016 році хакер скористався вадою стороннього проекту The DAO та викрав 50 мільйонів доларів ефіру. В результаті спільнота Ethereum проголосувала за форк блокчейну, щоб скасувати крадіжку, і Ethereum Classic (ETC) продовжив свою роботу як оригінальний ланцюжок.

Ethereum розпочав реалізацію серії модернізацій під назвою Ethereum 2.0, яка включає перехід до підтвердження частки і спрямована на збільшення пропускної здатності транзакцій.

Етеріум дозволяє запускати програмні застосунки в розподіленій мережі, стійкі до зовнішніх впливів. Додатки включаються в блокчейн ефіру, піддають криптографіному захисту та стають стійкими до редагування. Відповідно до засад блокчейну, всі застосунки, їх код доступний всім і кожний може перевірити його цілісність.

Усі після запуску застосунку його неможливо зупинити програмними методами шахрайства. Користувачі мережі можуть бути впевненими в надійності, безпеці користування безвідмовності мережі. Оскільки мережа Етеріуму користується криптовалютою, то програмісти можуть застосовувати у своїх програмних застосунках певні правила обміну цінністю. Також можливо розробити спеціальні додатки – смарт-контракти, які будуть виконувати певну дію за деяких наперед заданих умов, наприклад надходженню грошей на рахунок користувача.

2.2.1 Порівняння з Біткоїном

У біткоїні головним чином використовується блокчейн та майнінг як причина користувачам для утримання мережі. Як перша масова криптовалюта біткоїн він викликав революцію у фінансовій сфері, дозволивши користувачам обмінюватися валютою без перевірчого органу контролю. Кожен користувач має доступ до всіх операцій у мережі та може перевіряти дані в розподіленій мережі.

Структура бітокоїна відносно складна та не гнучка до змін. Хоча це робить мережу стійкою до відмов та полегшує вив махінацій, проте робить еможливим повноцінне використання смарт-контрактів на кшталт Ефіріуму. Біткоїн відносять до блокчейнів першого рівня. Вони підходять для застосування тільки як платіжна система.

Натомість Етеріум є другим поколінням блокчейну та надає можливість програмувати застосунки. Користувачі можуть експериментувати з мережею, створювати різноманітні додатки з довільними умовами обміну грошима. Ці додатки будуть розподіленими по всій мережі, тому носять назву децентралізованих додатків (від англійського Decentralized Applications, скорочено dApps).

Етеріум є першим блокчейном другого покоління. Він відразу здобув значну популярність завдяки своїй гнучкості. Його можна застосовувати не тільки в якості криптографічної валюти, а й довільних інших сферах, застосунки до яких можна відобразити в програмному коді. Між Ефіріумом та Біткоїном є багато як спільного, так і відмінностей. Кожен із них здобув популярність і продовжує нею користуватися.

2.2.2 Принцип роботи

Етеріум працює як система станів, постійно переходячи від одного стану до іншого. Стан враховує кошти користувачів, статус смарт-контрактів. Статуси постійно змінюються, що викликає зміну стану мережі та перехід в новий (рис. 2.2).

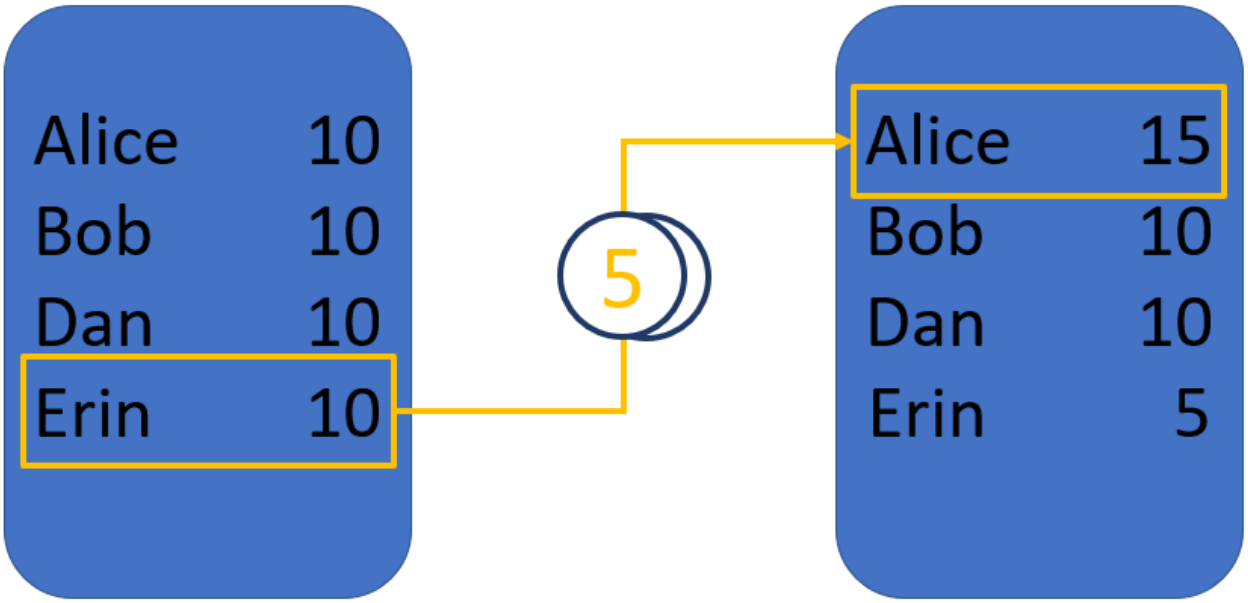

Рисунок 2.2 – Перехід ефіріуму в інший стан

Смарт-контракти являють собою програмний код. Їх називають розумними, бо вони можуть забезпечувати додержання угод між користувачами, перевіряючи виконання умов створення контракту. Це відбувається без третіх особі. Код смарт-контрактів Етеріуму починає виконання при поступанні грошей на контракт. Код запускається відразу на всіх вузлах мережі, зберігаючи результат роботи. Для цього використовується віртуальна машина Ефіріума (від англ. Ethereum Virtual Machine, скор. EVM). Вона розшифровує смарт-контракт, компілює його код та робить зрозумілим для виконання. Як і в біткоїні, для оновлення статусу мережі використовуються алгоритми доказу роботи Proof of Work.

Програміст вкладає в розумні контракти програмний код, компілюємий у віртуальну машину Ефіріуму. Для завантаження в мережу потрібно надіслати

контракт на зарезервований вузол, який зареєструє його та створить унікальну адресу контракту. Після цього для активації контракту необхідно надіслати вказану суму ефіру на його новостворену адресу. Смарт-контракт почне своє виконання.

Код контракту може виконувати довільні дії, які зможе описати програміст. Крім того, можна розробити розподілені додатки з декількома різними контрактами.

2.2.3 Структура транзакцій

Головною складовою обміну даними в Ефіріумі є транзакції. Ними є певні інструкції, підписані криптографічними методами. Транзакції використовуються для надсилання повідомлень та створення смарт-контрактів у мережі.

Транзакції містять наступні параметри:

1. **nonce** – кількість транзакцій, які були зроблені відправником.
2. **gasPrice** – сума Wei (найменша неділима частина ефіру), яку відправник може сплатити за одиницю газу, який використовується для підтвердження угоди.
3. **gasLimit** – максимальний об'єм газу , який відправник може витратити на виконання транзакції.
4. **to** – адреса одержувача. Поле залишається порожнім при створенні контраку.
5. **value** – сума Wei, які будуть передані від відправника до одержувача. При створенні контракту в цьому параметрі визначається початковий баланс гаманця контракту.
6. **v, r, s** – параметри, які використовуються для створення підпису, підтверджуючи особу відправника.
7. **init** – вказується тільки в контрактах. Фрагмент EVM-коду, який використовується для ініціалізації нового контракту. Як і конструктори класів, init виконується тільки один раз і надалі не використовується. При ініціалізації контракту повертається код його тіла.
8. **data** – вхідні дані транзакції чи контракту, використовуються для введення даних ззовні.

2.2.4 Структура блоків

Для початку введемо нове вихначення. Оммер (від англ. «ommer») – це блок, батьком якого є батьківський елемент поточного блоку. Їх наявність, в першу чергу, пов'язана з тим, що час блокування в Ефіріумі набагато нижче (приблизно 15 секунд), ніж для інших блокчейнів, наприклад, для біткоїну

(приблизно 10 хвилин). Завдяки такій особливості швидкість проведення транзакцій збільшується. З іншого боку, однією з негативних сторін більш короткого часу блокування є те, що боротьба майнерів за чергове блочне рішення тільки посилюється. Такі конкуруючі блоки ще називають «блоки без батька» (тобто такі блоки не входять до основного ланцюжка блоків).

Оммери були створені для того, щоб майнери могли отримати заслужену нагороду за включення блоків без батьків в основний ланцюжок. Оммери, включені майнером в основний ланцюжок, повинні бути «дійсними»: вони, оммери, повинні бути нащадками в шостому або більш ранньому поколінні поточного блоку. Наприклад, після шостого покоління такі нащадки не можуть бути включені в основний ланцюг в якості блоків без батька: більш пізні транзакції можуть негативно впливати на роботу системи в цілому.

Кожен блок має заголовок (рис. 2.3) – це частина блоку, яка складається з:

1. **parentHash** – хеш заголовку попереднього блоку
2. **ommersHash** – хеш списку оммерів поточного блоку
3. **beneficiary** – адреса користувача для отримання винагороди за створений блок
4. **stateRoot** – корінний хеш вузла префіксного дерева для збереження його стану
5. **transactionsRoot** – корінний хеш дерева транзакцій
6. **receiptsRoot** – корінний хеш вузла префіксного дерева для збереження інформації про оплату
7. **logsBloom** – Фільтр Блума інформації, збереженої в блоці
8. **difficulty** – складність блоку
9. **number** – порядковий індекс
10. **gasLimit** – теперішній ліміт газу
11. **gasUsed** – об’єм газу, який треба для виконання транзакції, збереженої в блоці
12. **timestamp** – час створення блоку
13. **extraData** – довільні інші дані, що мають певне значення для конкретного блоку
14. **mixHash** – хеш підтвердження роботи, використовується разом з nonce
15. **nonce** – змінна підтвердження роботи

Варто зауважити, що кожен заголовок блоку містить три структури префіксного дерева Меркла для:

1. стану (stateRoot)
2. проведення транзакції (transactionsRoot)
3. отримання інформації про оплату (receiptsRoot)

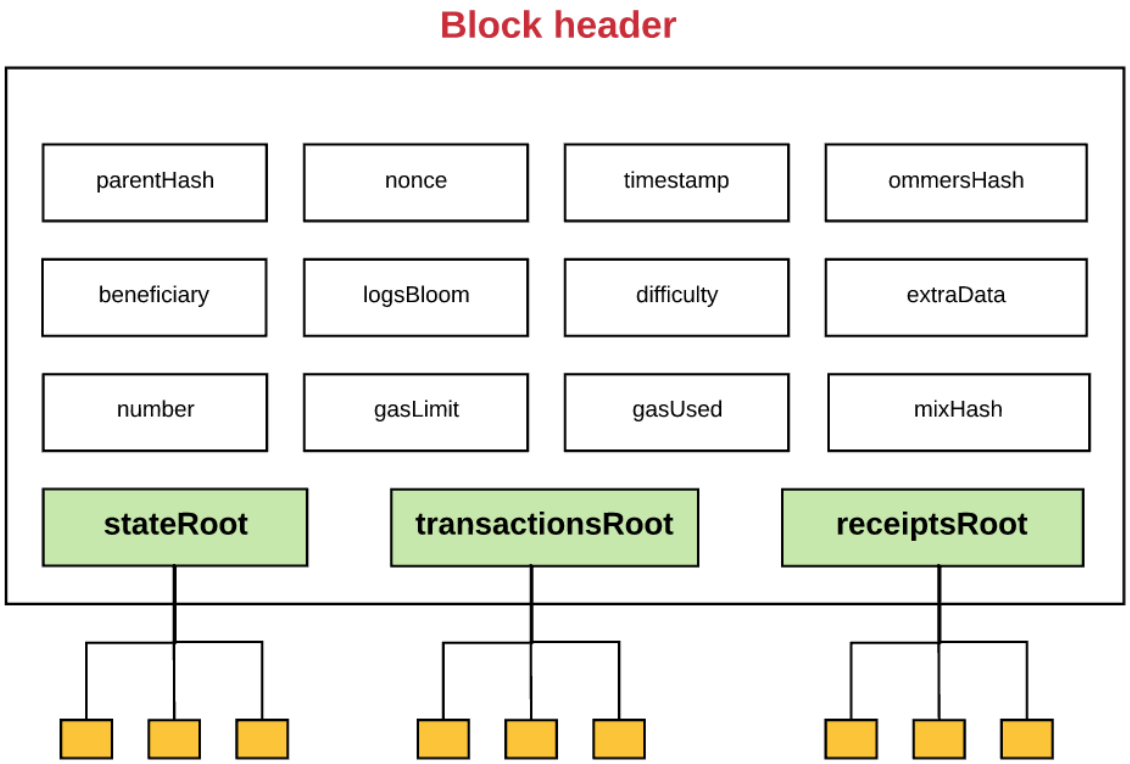

Рисунок 2.3 – Структура заголовку блока

2.3 Litecoin

Litecoin (від англ. Lite – «легкий», англ. Coin – «монета»), лайткоїн – це однорангова криптовалюта, яка спрямована на забезпечення миттєвих, майже нульових платежів, які можуть бути здійснені між людьми або установами в будь-якій точці світу. Лайткоїн, мабуть, найвідоміший за використання математичного коду, подібного до біткоїну, але із в чотири рази більшою пропозицією та в чотири рази більшою швидкістю обробки.

Децентралізація банківської системи є метою лайткоїна. Багато хто з нас, хто коли-небудь намагався відправити або отримати гроші через фінансові установи або в іншу країну, ймовірно, стикалися з незручностями. Чесно кажучи, не так просто зрозуміти, як гроші відправляються між банками, які витрати і чому транзакції займають так багато часу. Для любителів технологій (і установ, що переміщують великі суми грошей) ці незручності ще більш виражені в цифрову епоху.

Чарлі Лі, Колишній співробітник Google й інженерний директор Coinbase, випустив Litecoin в 2011 році . Для порівняння, біткоїн був створений в 2009 році невідомою особою або групою осіб на ім'я Сатоші Накамото.

Лі розробив лайткоїни як комплімент оригінальній криптовалюті, а не як заміну або навіть конкурента. Ось чому його іноді називають молодшим братом біткоїна.

Оскільки Litecoin може створювати блоки за 2,5 хвилини (на відміну від 10 для біткоїнів), час транзакції також швидше. Таким чином, Лайткоїн часто вважається "легшою", більш швидкою версією біткоїна.

Litecoin являє криптовалюту для платежів, засновану на технології блокчейн. Його основна мета полягає в тому, щоб виступати в якості засобу для здійснення платежів без банку або іншого стороннього посередника.

Litecoin використовує дуже схожу на біткоїн технологію, але з можливістю проводити транзакції швидше, ніж біткоїн.

За даними Litecoin, для обробки блоку потрібно дві з половиною хвилини в порівнянні з десятьма, що робить валюту в чотири рази швидшою, ніж біткоїн. Компроміс полягає в тому, що транзакція, створена в лайткоїнах, може бути не такою безпечною, як транзакція в біткоїні.

Але, звичайно, багато людей цікавляться лайткоїном як потенційним довгостроковим холдингом, а не тільки як засобом обробки транзакцій. Як і при покупці будь-якого типу валюти, є надія, що нова валюта збільшиться у вартості по відношенню до базової валюти.

Тому багато спекулянтів, які вивчають криптовалюту, таку як litecoin, зазвичай припускають, що ця валюта з часом створить відносне багатство. Проте завжди існують ризики, пов'язані з спекулятивними іграми, такими як валюта.

2.3.1 Порівняння з Біткоїном

Лайткоїн має багато подібного з біткоїном, адже це криптовалюти, засновані на блокчейні. Крім того лійткоїн є нащадком біткоїну, тому він перейняв певні рішення в структурі блоків, транзакцій та мережі загалом. Проте також відбулися певні значні зміни, які перенесли лайткоїн на щабель вище братської криптовалюти.

Обидві валюти використовують алгоритми доказу роботи. Валюти можна видобути шляхом майнінгу при створенні блоків або ж купити за реальні гроші. Обидві валюти використовують криптографічний захист, такий як цифрові підписи та хеш-функції. Варто зазначити, що вартість монети обох криптовалют дуже нестійка.

Перша відмінність між лайткоїном та біткоїном полягає в ринковій капіталізації валют. Так вона в біткоїна в 70 раз більша за лайтоїн. Натомість

лайткоїн зможе використовувати більшу кількість монет. Місткість його мережі складає 84 мільйони монет порівняно з 21 мільйонами біткоїнів.

Ще однією перевагою лайткоїну є швидкість його транзакцій. Вона в чотири рази краща за конкурента. Це є наслідком того, що лайткоїн розроблявся спеціально для швидких транзакцій, щоб пришвидшити обмін ресурсами в мережі.

Головною технічною відмінністю між криптовалютами є використання різних криптографічних алгоритмів. Лайткоїн побудований на новому крипто-алгоритмі Scrypt, тоді я біткоїн застосовує застарілу хеш-функцію SHA-256. Хоча обидва алгоритми виконують поставлені свої функції, проте SHA-256 є не оптимальним в даний час для використання в мережах блокчейну. Це викликане тим, що незважаючи на складність обчислень хеш біткоїнів легко піддається паралельному обчислення, що призводить до значної витрати електроенергії на спецобчислювачі майнерів, які намагаються видобути блоки в конкурентній боротьбі. У свою чергу Scrypt не потребує великих спеціальних потужностей та може обраховуватися на звичайних користувацьких комп'ютерах. Це дає змогу всім учасникам мережі виступати в ролі майнерів, щоб підтверджувати транзакції швидше.

У вересні 2017 року завершилася робота над протоколом Lightning Network (LN) для мережі Лайткоїн. Протокол був розроблений для пришвидшення до миттєвого рівня транзакцій та відсутності комісії для них. Сенс протоколу полягає в тому, що транзакції на отримання коштів відбуваються миттєво та зберігаються в буфері, аж допоки хтось із користувачів не захоче вивести кошти з мережі. Це дозволило зробити мережу ще швидшою та привабливішою для нових користувачів.

2.3.2 Принцип роботи

Щоб зрозуміти, як працює litecoin, корисно спочатку отримати базові знання про базову технології блокчейна. За допомогою блокчейна інформація кодується і зберігається в блоці, і кожен блок, пов'язаний разом, створює ланцюжок. Ланцюжок інформації діє як книга транзакцій litecoin.

Блокчейн – це відкрита розподілена бухгалтерська книга. Сама книга також може бути запрограмована на автоматичне ініціювання транзакцій. Інформація, яка використовується в системі блокчейн, зберігається в безпеці за допомогою методів шифрування. Транзакції з використанням технології блокчейн зазвичай вважаються анонімними (хоча насправді вони є псевдонімами – оскільки у кожного користувача є публічна адреса, хтось може виконати велику роботу, щоб відстежити його до фактичної IP-адреси і, отже, фактичної людини).

Як і в багатьох криптовалютах, лайткоїни видобуваються користувачами, сподіваюся, в обмін на валюту. Майнер перевіряють транзакції і створюють нові блоки, вирішуючи складні математичні рівняння, що робить Лайткоїн частиною когорти валют, заснованої на математиці.

За допомогою litecoin майнери отримують 25 нових лайткоїнів за кожен видобутий блок, сума зменшується вдвічі приблизно кожні чотири роки (кожні 840 000 блоків).

Загальна капіталізація Litecoin становить 84 мільйони лайткоїнів – в чотири рази більше одиниць, ніж біткоїн. Подібно біткоїну, Лі розробив Лайткоїн, щоб більшість монет було видобуто в перші два десятиліття.

Будь-хто, хто відчуває себе готовим зробити рішучий крок і придбати Лайткоїн, може захотіти дізнатися про унікальний спосіб покупки і зберігання криптовалюта. Неможливо купити лайткоїн або інші криптовалюти через багатьох традиційних брокерів. Замість цього лайткоїн повинен бути придбаний в цифровому гаманці, через одну з криптовалютних бірж або через онлайн-брокерську фірму, яка пропонує криптоторгівлю. Подумайте про збори, безпеку та доступність, перш ніж приймати рішення про те, де купувати і зберігати криптовалюту.

Незалежно від того, де купуються лайткоїни, вони повинні зберігатися в криптовалютному гаманці. Гаманець, який є програмним забезпеченням або реальним обладнанням, дозволяє надсилати та отримувати цифрові валюти та стежити за Вашою спекулятивною грою.

Можливо, було б корисно зрозуміти, що криптовалюта не "зберігається" в традиційному сенсі, як в гривнях, які зберігаються на банківському рахунку. Замість цього записи транзакцій зберігаються в блокчейні.

Більшість варіантів гаманців розглядаються або в автономному режимі, або в Інтернеті, причому онлайн-гаманці, як правило, вважаються більш ризикованими і більш відкритими для злому, ніж автономні гаманці. Автономний гаманець, наприклад гаманець, що зберігається на жорсткому диску, може бути більш безпечним, але покупці ризикують втратити жорсткий диск.

Як і більшість найпопулярніших криптовалют, Litecoin зазнав значну волатильність за свою коротку історію на цій планеті. Ця волатильність відбувалася як з позитивного, так і з негативного боку. У своїй найвищій точці, в грудні 2017 року, Лайткоїн торгувався на рівні понад 375 доларів, що стало його самим стрімким зростанням на сьогоднішній день. У грудні 2018 року ціна була всього 24 долари, до чергового сплеску на початку 2019 року.

2.3.3 Ризики використання Litecoin (та інших криптовалют)

Ризики лайткоїна аналогічні ризикам більшості криптовалют. По-перше, потенційним покупцям розумно пам'ятати про шахраїв. На жаль, завжди є люди, які готові скористатися перевагами інших. Крім того, криптовалютні гаманці і біржі не захищені від злому. Злом може стати серйозною проблемою для будь-кого, хто приймає рішення про те, де купувати і зберігати свої гроші. Якщо криптовалюта зламана й вкрадена, цілком можливо, що не існуватиме способу її відновити.

Ось ще одна перспектива: для уявлення про позицію комісії з цінних паперів і бірж щодо криптовалют, агентство недавно знову відмовило в крипто-ETF, заявивши, що "його несхвалення не залежить від того, чи має лайткоїн або блокчейн в цілому корисність або цінність в якості інновації або інвестиції", а тому, що він не відповідає вимозі "запобігати шахрайським і маніпулятивним діям або практикам" – стандартній вимозі SEC.

Коли мова заходить про криптовалюту, існує безліч "що, якщо". Наприклад, яка доля криптовалют, якщо вони не будуть широко прийняті? Криптовалюта, поряд з блокчейном, – це технології, які все ще знаходяться в зародковому стані. Як і у випадку з винаходом Інтернету і подальшими компаніями, пов'язаними з Інтернетом, надзвичайно важко передбачити, які криптовалюти можуть бути успішними або чи будуть які-небудь з них успішними.

Важко сказати, як в кінцевому підсумку буде використовуватися технологія блокчейна. Ось одна думка про прогнозоване впровадження технології блокчейн з Harvard Business Review: "хоча вплив буде величезним, будуть потрібні десятиліття, щоб блокчейн просочився в нашу економічну і соціальну інфраструктуру". "Процес прийняття буде поступовим і стійким, а не раптовим, оскільки хвилі технологічних та інституційних змін набирають обертів."

Автори продовжують говорити: "Наш досвід вивчення технологічних інновацій говорить нам, що якщо відбудеться революція блокчейна, багато бар'єрів технологічні, управлінські, організаційні і навіть соціальні – повинні будуть впасти. Блокчейн і криптовалюта в даний час стикаються з серйозними проблемами регулювання з боку тих самих систем, які вони прагнуть зруйнувати – і можуть продовжувати це робити.

Ризик є природною частиною покупки волатильних активів. Але важливо знати, що криптографія через свою волатильність несе в собі більш високий ступінь ризику. При покупці криптовалюти покупець повинен знати, скільки він може дозволити собі втратити. При покупці криптовалюти може бути корисно економити й інвестувати в інші більш усталені способи. І завжди є можливість придбати невелику кількість крипти, що дещо знижує ризик.

РОЗДІЛ 3 ПОРІВНЯННЯ ТЕХНОЛОГІЙ РОЗПОДІЛЕНОГО РЕЄСТРУ

Технологія розподіленого реєстру (DLT, distributed ledger technology) сама по собі не нова. Блокчейн – це найперша DLT, запущена в публічний простір. І це незважаючи на те, що перший опис блокчейна з'явився майже три десятиліття тому. Блокчейн вперше став відомий світові відразу після появи біткоїнів, найпершої цифрової валюти. Впродовж останніх років виникло багато цікавих розробок, в яких технологія розподіленого реєстру або, іншими словами, блокчейн знайшла безліч варіантів використання крім роботи з криптовалютою.

Однак системна неефективність і проблеми масштабування привели до того, що розробники стали шукати рішення поза блокчейном. Технологія пройшла такий довгий шлях, що вже є модифікації й альтернативи. Таким чином, існують нові та геніальні розробки, такі як Holochain, Directed Acyclic Graph і Hashgraph. По суті, їхнє завдання полягає в тому, щоб зберегти початкову мету блокчейну перед обличчям нових і непередбачених труднощів.

Поява нових рішень, які багато в чому відрізняються від блокчейну структурою даних, викликало інші фундаментальні, але все ж важливі дискусії про те, яка технологія є найкращою. Створюючи основу для інноваційного нового способу зберігання, спільного використання та виконання багатьох інших дій з даними, блокчейн займає лідируюче місце в гонці мереж технології розподіленого реєстру. Таким чином, в наступних розділах буде порівнюватися блокчейн з іншими мережами DLT.

3.1 Hashgraph

Хешграф – це тип технології розподіленого реєстру, в основі якої лежить досягнення консенсусу. Зокрема, DLT покладається на узгоджену часову мітку, щоб гарантувати, що транзакції в мережі узгоджені з кожним вузлом на платформі. Алгоритм консенсусу надає надійність і перевагу мережі технології розподіленого реєстру.

На відміну від традиційної мережі розподіленого реєстру, цей тип побудови DLT забезпечує успіх транзакції виключно за допомогою консенсусу. Це означає, що вузлам не потрібно перевіряти транзакції, які відбуваються в мережі. Таким чином, користувачам не потрібно пред'являти докази роботи (PoW).

Цей аспект усуває необхідність в двох речах. По-перше, традиційні ланцюжки блоків, які покладаються на доказ роботи, вимагають безлічі обчислень для досягнення успіху транзакції. В результаті цей фактор робить транзакції громіздкими, що забезпечує дуже низьку кількість транзакцій в секунду.

У свою чергу Hashgraph навпаки вимагає, щоб вузли в мережі тільки досягли консенсусу за допомогою техніки пліток (Gossip about Gossip) і техніки віртуального голосування. Цікаво, що ці методи не вимагають підтвердження роботи для перевірки транзакцій. В результаті між ініціюванням та завершенням транзакції проходить мало часу.

Отже, відсутність необхідності в підтвердженні роботи в мережі DLT означає, що можуть виконуватися тисячі транзакцій за секунду (TPS, transactions per second). Цікаво, що команда, що стоїть за Hashgraph, стверджує, що мережа може досягати більше 250 000 TPS.

Завдяки віртуальному голосуванню і техніці пліток вузли Hashgraph DLT можуть досягти консенсусу. Зокрема, часові мітки консенсусу дозволяють уникнути проблем з ланцюжком блоків, таких як скасування транзакцій або їх розміщення в майбутніх блоках.

3.1.1 Порівняння Blockchain та Hashgraph

Очевидно, що блокчейн і Hashgraph мають багато спільного, оскільки служать подібній меті. По суті, вони представляють собою тип технології розподіленого реєстру, який прагне запровадити нові грошові системи. Зокрема, вони використовують однорангову мережу, тому транзакції не вимагають центрального органу для їх регулювання.

Крім того, ці DLT працюють на основі консенсусної системи, в якій транзакції повинні задовольняти учасників всередині мережі. Ця потреба в консенсусі є причиною того, чому транзакції в цих мережах прозорі та гнучкі. Крім того, високий ступінь захисту від криптографічної природи мереж забезпечує високу безпеку даних.

Проте, блокчейн відрізняється від Hashgraph більш фундаментально. Зокрема, Hashgraph є відповіддю на основні обмеження блокчейна, такі як масштабованість і швидкість транзакцій. Ці конкретні обмеження відповідальні за обмежене застосування DLT.

Хешграф в основному відрізняється від блокчейна, коли мова йде про механізм консенсусу. З одного боку, блокчейн в основному покладається на діяльність майнерів у мережі, щоб полегшити процес транзакцій. Це означає, що майнер може істотно вплинути на успіх транзакції самотужки.

Зокрема, блокчейн покладається на доказ роботи, що майнери можуть використовувати для перевірки справжності транзакції. Таким чином, швидкість транзакцій низька і дорога. Крім того, доказ роботи має на увазі, що мережа піддається величезному тягарю громіздких обчислень, які призводять до ускладнення мережі.

З іншого боку, Hashgraph використовує механізм консенсусу, який не дає майнерам занадто багато повноважень. Навпаки, DLT використовує

алгоритм консенсусу Gossip about Gossip і віртуальним голосуванням, щоб вирішити, яка транзакція буде успішною. Таким чином, більшість має право голосу при виборі транзакції. Тому Hashgraph має набагато більше справедливості в порівнянні з блокчейном.

3.2 Directed Acyclic Graph

Hashgraph не єдина спроба виправити обмеження блокчейну. Як згадувалося раніше, розробники зосереджуються на структурі даних мереж розподіленого реєстру, які впливають на їх ефективність. Таким же чином орієнтовані ациклічні графи (DAG, Directed Acyclic Graphs) використовують іншу структуру даних, яка забезпечує більший консенсус.

Зокрема, DAG – це тип технології розподіленого реєстру, заснований на консенсусних алгоритмах. Алгоритми консенсусу працюють таким чином, що домінуючі транзакції просто вимагають підтримки більшості в мережі. У такій мережі набагато більше співпраці, роботи в команді та ще й вузли мають рівні права.

На відміну від традиційних технологій блокчейн, де доказ роботи є ключовим, DAG гарантує справедливість. Така справедливість створює враження, що мережа дотримується первісної мети – технології розподіленого реєстру. Зокрема, головною метою DLT було демократизація інтернет-економіки.

Наприклад, приватна мережа блокчейнів покладається на централізоване керівництво, яке усуває демократію в DLT. Навпаки, цей тип технології розподіленого реєстру надає рівну значимість кожному вузлу, який існує в мережі. Отже, це означає, що кожен вузол не повинен посилатися на інший вузол.

Уже є такі проекти, як ByteBall, які використовують структуру DAG для створення мереж нового покоління, вільних від обмежень традиційного блокчейну. Однією з найбільш помітних мереж «нового покоління», які використовують структуру даних DAG, є Tangle IOTA.

Тут майнери та вузли можуть виконувати подвійні обов'язки, коли вузли в блокчейні працюють окремо. Це означає, що майнер на Tangle може одночасно виконувати транзакцію і підтверджувати транзакцію.

3.2.1 Порівняння Blockchain та DAG

Як і блокчейн, DAG спрощує транзакції і робить неможливим «повернути транзакцію» до більш раннього етапу. Зокрема, слово ациклічний в орієнтованому ациклічному графі означає, що операції строго однонаправлені. Точно так же незмінність – це один з аспектів, який робить блокчейн популярним в порівнянні з раніше існуючими засобами зберігання даних.

Крім того, обидві платформи працюють через систему на основі консенсусу, де вузли вирішують, що відбувається. Таким чином, існує щось на кшталт демократії в порівнянні з платформами, які проходять через центральне управління. На жаль, на цьому схожість закінчується.

Як і Hashgraph, DAG принципово відрізняється від блокчейна, коли справа стосується структури даних. Як обговорювалося раніше, блокчейн організовує транзакції в блоки, так що кожен обсяг інформації, що відноситься до конкретної транзакції, складає єдиний блок. Отже, успішні транзакції призводять до нових блоків.

У свою чергу DAG повністю працює з блоками. В DAG попередня транзакція має більш сильну зв'язок з подальшою транзакцією. Наприклад, якщо у вас було три транзакції, X, Y і Z, вам знадобиться транзакція X для Y, щоб пройти. Точно так же транзакція Y підтверджує транзакцію Z.

Щоб транзакція пройшла успішно в мережі DAG, вона повинна підтвердити тільки дві з попередніх транзакцій. Це означає, що транзакція повинна гарантувати, що дві попередні транзакції не містять суперечливої інформації. Цікаво, що це сильно відрізняється від блокчейна, де транзакція повинна перевіряти безліч транзакцій, перш ніж стати валідною.

Це означає, що для розрахунку транзакції потрібно більше часу. Крім того, у міру збільшення кількості блоків у блокчейні стає все важче з точки зору обчислень отримати нові блоки. Таким чином, майнінг стає більш енергоємним, а значить, і дорогим. З іншого боку, транзакції в мережі DAG збільшують пропускну здатність у міру того, як відбувається більше перевірок.

3.3 Holochain

Цей тип технології розподіленого реєстру може похвалитися тим, що він з'явився після технології блокчейн. Незважаючи на високу дозу риторики в їх сміливих заявах, команда Holochain дійсно має кілька переконливих пропозицій, які можуть конкурувати з іншими платформами.

Одне з цікавих видінь Holochain – змінити нинішню структуру Інтернету. Інтернет сьогодні структурований за принципом сервер-клієнт. Це означає, що децентралізація не оптимальна. Крім того, у використанні ресурсів мало демократії і свободи.

У світлі цього Holochain хоче створити розподілену мережу, яка також може стати основою «Інтернету наступного покоління». Згідно з офіційним документом платформи, Holochain є об'єднанням блокчейна, BitTorrent і Github. Це означає, що це DLT, який розподіляється між вузлами, щоб уникнути будь-якого централізованого управління потоком даних.

Розподілена платформа має на увазі, що кожен вузол буде працювати у власному ланцюжку. Це означає, що вузли або майнери можуть працювати автономно. У тому, що команда Holochain називає розподіленою хеш-таблицею (DHT, distributed hash table), користувачі можуть зберігати дані, використовуючи певні ключі. Однак ці дані залишаються в реальних місцях, «розподілених» в різних точках по всьому світу.

Цікавою частиною цієї структури даних є те, що мережі не відчуває тягаря перевантаження, характерного для традиційного блокчейну. Ця «розподіленість» місць, де зберігаються дані, розряджає мережу і дає їй більше місця для масштабованості. Таким чином, транзакції в цій мережі можуть легко досягати мільйонів TPS.

Масштабованість – величезна проблема, яка переслідує як публічні, так і приватні платформи блокчейнів. Наприклад, розробники зазвичай стикаються з величезними перешкодами при створенні децентралізованих додатків на традиційному блокчейні. Це тому, що вони потребують перевірки величезної спільноти учасників платформи.

Розробнику на платформі Holochain, навпаки, потрібно підтвердження тільки від одного ланцюжка, що становить всю мережу DLT. Таким чином, в цьому типі технології розподіленого реєстру час очікування між запитом і підтвердженням стає незначним.

3.3.1 Порівняння Blockchain та Holochain

У порівнянні DLT розгляд фундаментальної структури Holochain і блокчейна виявляє різні відмінності. Зокрема, вони багато в чому розрізняються за своєю структурою, хоча цілі мають деяку схожість. Цікаво, що Holochain – це свого роду революційна технологія, яка прагне перевернути все з ніг на голову.

Як і блокчейн, Holochain прагне забезпечити безпечні і прозорі транзакції між учасниками в мережі. Інформація на обох платформах криптографічно безпечна, і її не можна змінити. Крім того, обидві платформи дозволяють користувачам взаємодіяти в одноранговій мережі. Таким чином, вони можуть взаємодіяти безпосередньо і без необхідності в центральній владі.

Тим не менш, Holochain – це дещо наступний рівень у порівнянні з блокчейном. По суті, Holochain прагне ввести новий функціонал, який сильно відрізняється від фундаментальної мети блокчейна. Блокчейн прагне децентралізувати транзакції, щоб люди могли взаємодіяти безпосередньо без посередників. У свою чергу, Holochain хоче, щоб взаємодії були розподіленими.

Holochain створює мережу, що складається з різних мереж технології розподіленого реєстру. Отже, DLT – це одна з основних мереж, яка безмежна з

точки зору масштабованості і кількості транзакцій, які користувачі можуть виконати за секунду.

У мережі блокчейн вузли покладаються на єдину мережу для ініціювання та перевірки транзакцій. Таким чином, чим більше блоків приєднується до ланцюжка, тим більше обчислювальне навантаження збільшується, а також збільшуються комісії, пов'язані з транзакціями. Навпаки, вузли в Holochain працюють у своїх власних ланцюжках. Отже, залишається більше місця для обчислень.

Той факт, що кожен вузол працює у своєму власному ланцюжку в Holochain, означає, що майнери не потрібні. Таким чином, комісії за транзакції практично відсутні. Крім того, це означає, що на платформі немає токенізації, а скоріше розумні контракти управляють нею.

Вузли, що працюють у своїх власних ланцюжках, можуть обробляти реєстри, які належать виключно їм. Таким чином, відносини між різними вузлами в мережі повністю довірені. Крім того, у dApps є нескінченний простір для роботи. Таким чином, можна очікувати, що dApps працюватимуть в оптимальному режимі в усіх випадках.

3.4 Підсумки порівняння

Порівняння DLT в Blockchain, Hashgraph, DAG і Holochain виявляє цікаві аспекти платформ. Незважаючи на очевидну подібність між DLT, помітні й відмінності. Цікаво, що блокчейн існував ще до офіційного документа про біткоїн, розробленого псевдонімом Сатоші Накамото.

Блокчейн вперше був згаданий в документі, в якому була зроблена спроба знайти спосіб захисту інтелектуальної власності за допомогою міток часу в документах. Однак DLT набув популярності після популярності біткоїнів. З популярністю прийшли різні непередбачені проблеми, такі як масштабованість і TPS. Незважаючи на те, що блокчейн намагається виправити обмеження, інші проєкти повністю створюють новий тип технології розподіленого реєстру.

З вищевикладеного очевидно, що всі DLT поділяють спільні аспекти прозорості, консенсусу, орієнтації на транзакції, розподіленості, одноранговості та гнучкості. Однак виникають величезні відмінності в механізмі консенсусу і структурі даних в кожному DLT.

Результати порівняння вищеописаних технологій розподіленого реєстру наведено в наступній таблиці:

	Blockchain	Hashgraph	DAG	Holochain
Дата запуску	2008 рік	24 серпня 2018 року	NXT – 9 листопада	Альфа-1 – 26 травня 2018

			2015 року	року
Майнінг	Учасники можуть майнити нові токени за допомогою різних алгоритмів консенсусу	Вузли створюють консенсус завдяки віртуальному голосуванню	Попередні транзакції валідують досягнення консенсусу	Вузли працюють на індивідуальних ланцюгах, тому немає потреби в майнерах для підтвердження транзакції
Швидкість транзакцій, TPS	Сильно обмежена в термінах масштабованості та TPS	Унікальний механізм консенсусу скорочує обчислювальне навантаження, тому масштабованість та TPS високі	Унікальна структура даних на спрямованих ациклічних графах створює високу масштабованість та TPS	Кожен вузол обробляє власний ланцюг блоків, тому масштабованість та TPS необмежені
Структура даних	Дані структуровані в блоки у порядку транзакцій, які були підтверджені майнерами	Віртуальне голосування та унікальний механізм консенсусу гарантують валідацію транзакцій за пріоритетністю	Структура даних наслідує направлений ациклічний граф, де кожна транзакція незалежна	Дані розподілені серед різних вузлів платформи, звідси відсутність проблем із перевантаженням мережі
Валідація транзакцій	Майнери мають владу відкласти транзакцію або скасувати її повністю	Валідація транзакцій відповідно до консенсусу	Успіх теперішньої транзакції покладається на її здатність перевірити дві попередні	Вузли обробляють їхній власний ланцюг, тому відсутня потреба в майнерах

Застосування	Bitcoin, Ethereum	Swirlds, NOIA	NXT, Tangle, ByteBallï	Holochain

РОЗДІЛ 4 ПРОГРАМНИЙ ЗАСТОСУНОК «КЛІЄНТ МЕРЕЖІ БЛОКЧЕЙН»

4.1 Постановка задачі

Як було описано в попередніх розділах, мережі блокчейн створюються для різноманітних задач та кожна з них має свою реалізацію. Для найбільш популярних застосувань блокчейну вже існують реалізації від великих корпорацій. Зрозуміло, що силою однієї людини розробити конкурентне рішення неможливо. Проте можна розробити певний кістяк, шаблон, на якому бажаючі зможуть створити нову унікальну мережу для власних цілей та локального застосування. Це і є метою створення програмного застосунку «Клієнт мережі блокчейн».

Для клієнтів мережі можна виділити спільні риси, хоча й вони можуть відрізнятися залежно від ролі клієнта в мережі. Виділимо загальні функції, притаманні користувачам:

1. користувачі з’єднані між собою та можуть обмінюватися даними
2. учасники мережі здійснюють транзакції
3. підтвердження транзакцій відбувається шляхом занесення їх у блок і майнінгу
4. кожен клієнт зберігає копію ланцюга блоків
5. наявна можливість перевірки коректності ланцюга
6. розгалудження основного ланцюга блоків вирішується обиранням найдовшої серії блоків, починаючи з генезисного

Реалізація описаних вище функцій є постановкою задачі.

4.2 Розробка структури програмного застосунку

Для початку визначимося із технологіями розробки (мова програмування, фреймворки), які будемо застосовувати. Метою створення додатку є його подальше застосування із внесенням змін до програми, необхідних конкретному розробнику. Тому мову програмування необхідно обирати популярну, просту в написанні, зрозумілості коду та його модифікації. Зрозуміло, що будуть модифікуватися частини коду, тому необхідно створювати проєкт на основі об’єктно орієнтованого програмування (ООП) [23]. Взаємодія між користувачами найчастіше відбувається мережею Інтернет, тому й у додатку використовуватимуться http-запити. З переліченого робимо висновок, що вимогам найкраще відповідає мова Python. Для розробки веб-додатку застосуємо фреймворк Flask як один із найбільш популярних.

Створення структури проєкту полягає в розбитті програми на класи та виділення функцій кожного з них. Кожен клас являє собою певний об’єкт,

структурну одиницю функціонування мережі. У блокчейні такими структурними одиницями можна виділити транзакції, блоки, саму мережу, її вузли та клієнтів. Нижче буде наведено реалізовану структуру класів. Вона розроблена із загальних міркувань та може бути змінена відповідно до потреб майбутніх користувачів мережі.

Так як планується передача даних між комп'ютерами мережею Інтернет, було вирішено передавати дані у форматі JSON [22]. Необхідно мати змогу представити дані класів у цьому форматі. Для цього було вирішено спадкувати класи від базового класу словника (dict [25]). Це ж вирішує проблему додавання нових атрибутів об'єктам мінімальною зміною коду, не ламаючи структуру програми.

Кожна транзакція в мережі має зберігати інформацію про її відправника та отримувача. Інші атрибути можуть відрізнятися. Було вирішено додати атрибут кількості переданих грошових одиниць. Очевидно, що в якості платіжної системи додаток не буде застосовуватися, адже існують великомасштабні криптовалютні мережі блокчейну. Проте даний атрибут чудово підходить для нарахування винагороди за майнінг блоку, якщо така буде існувати. Також важлива реалізація методу отримання хешу транзакції для подальшого його застосування для перевірки цілісності даних (для цього реалізоване дерево Меркла). Реалізована структура класу транзакцій зображена на рис. 4.1.

```
class Transaction(dict):
    """
    Клас для збереження транзакцій у мережі
    """
    def __init__(self, sender: str, recipient: str, quantity: float):···

    def get_transaction_hash(self) -> str:···
```

Рисунок 4.1 – Структура класу транзакцій

У блоках мережі блокчейн зберігаються транзакції. Із хешів транзакцій утворюється дерево Меркла. Воно необхідне для перевірки цілісності даних, які зберігаються у блоці. Кожен блок мережі містить хеш попереднього блоку та свій, тому необхідний метод для його обрахунку. Хеші блоку та кореня дерева Меркла обраховуються в конструкторі класу відповідними методами класу. Для доказу роботи над створенням блоку було вирішено використати алгоритм proof-of-work, як один із найзрозуміліших та простих у реалізації. Для майнігу блоку необхідно знайти таке значення (proof), щоб хеш блоку починався з певної послідовності (NONCE). Відповідний метод було додано до структури. Також блок має часову мітку, яка також є часом підтвердження транзакцій цього блоку. Додатково було вирішено зберігати порядковий номер блоку для

легшого спостереження на ланцюгом мережі. Реалізована структура класу блоку зображена на рис. 4.2.

```python
class Block(dict):
    """
    Клас для збереження блоків у блокчейні
    """
    NONCE = "0"  # Необхідний заголовок хешу блоку для PoW

    def __init__(self, index: int, time: float, transactions: list, proof: int, prev_hash: str):···

    def get_block_hash(self) -> str:···

    def merkle(self, transaction: list) -> str:···

    def validate_proof(self) -> bool:···
```

Рисунок 4.2 – Структура класу блоку мережі блокчейн

Клієнтський ланцюг блоків зберігатиметься в класі Blockchain. Крім нього в цьому класі зберігатимуться транзакції, які мають бути додані до наступних блоків, а також посилання на інші вузли мережі. Для збереження принципу інкапсуляції в класі мають бути реалізовані наступні методи: додавання нових вузлів мережі, пошук найдовшого ланцюга в мережі та перевірка його валідності, додавання нових блоків та транзакцій. Так як клієнт взаємодіятиме з іншими класами через клас Blockchain, то було додано метод перевірки знайденого значення доказу роботи, який викликатиме відповідний метод класу блоку. Реалізована структура класу мережі блокчейн зображена на рис. 4.3.

```python
class Blockchain:
    """
    Клас для збереження ланцюга блокчейну
    """
    def __init__(self):
        """
        Конструктор класу
        """
        self.current_transactions = []  # Транзакції, які мають бути додані до наступного блоку
        self.chain = []  # Ланцюг блоків
        self.nodes = []  # Список адрес інших користувачів мережі

        # Створення початкового (генезисного) блоку
        self.new_block(Block(index=0, time=time(), transactions=[], proof=0, prev_hash=Block.NONCE))

    def add_node(self, link: str):···

    def validate_chain(self, chain: List[Block]) -> bool:···

    def find_main_chain(self) -> bool:···

    def new_block(self, block: Block) -> Block:···

    def new_transaction(self, sender: str, recipient: str, quantity: float):···

    def proof_of_work(self, proof: int):···
```

Рисунок 4.3 – Структура класу мережі блокчейн

Для відображення вузлів мережі був створений клас Node. У даній реалізації міжкористувацька взаємодія відбуватиметься http-запитом із вказання ір-адреси та порту іншого користувача, тому в класі необхідний тільки один атрибут для збереження цього посилання. У подальшому впровадженні проєкту можливе ускладнення класу. Реалізована структура класу вузла мережі зображена на рис. 4.4.

```
class Node(dict):
    """
    Клас для збереження адрес інших користувачів
    """
    def __init__(self, link: str):···
```

Рисунок 4.4 – Структура класу вузла мережі блокчейн

Клієнт мережі блокчейн являє собою виконавчий файл, який не потребує створення окремого класу. Клієнт отримує запити та взаємодіє з класом мережі блокчейн. Необхідним мінімальним функціоналом клієнта є майнінг блоків, створення транзакцій та оновлення ланцюга блоків. Решта функціоналу покладається на клас Blockchain. У розробленому додатку кожен клієнт містить додатковий ідентифікатор клієнта, окрім ір-адреси та порту (передаються при запуску файлу), на випадок динамічної зміни останніх. Реалізована структура виконавчого файлу клієнта зображена на рис. 4.5.

```
user_id = str(uuid4()).replace('-', '')  # Генерує ідетифікатор клієнта

@app.route('/mine', methods=['GET'])
def mine():···

@app.route('/transactions/new', methods=['POST'])
def new_transaction():···

@app.route('/chain', methods=['GET'])
def chain():···

@app.route('/nodes/add', methods=['POST'])
def add_nodes():···

@app.route('/chain/update', methods=['GET'])
def update_chain():···
```

Рисунок 4.5 – Структура виконавчого файлу клієнта

Загальну структуру проєкту на основі вищевказаних класів зображено на наступному рисунку:

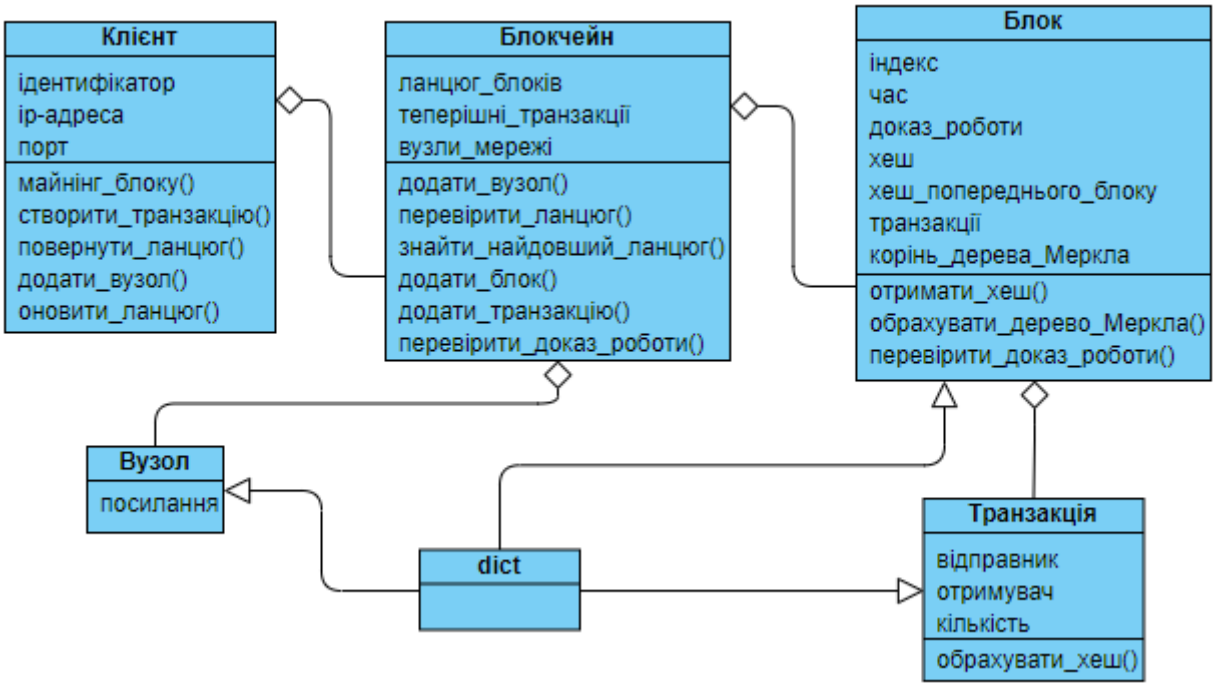

Рисунок 4.6 – Діаграма класів програмного застосунку

4.3 Програмна реалізація застосунку

У цьому підрозділі буде описана реалізація методів класів відповідно до зазначеної в попередньому підрозділі структурі.

У класі Node зберігається посилання на вузол мережі. Значення цього посилання передається в конструктор класу. Так як нам не відомий тип цього посилання, скористаємося методом urlparse бібліотеки urllib.parse [24]. Після виклику цього методу повертається об’єкт типу ParseResult, в атрибуті netloc якого зберігається Network location part, а в атрибуті path – Hierarchical path. Якщо обидва ці атрибути порожні, то посилання спарсити не вийшло, а отже воно вказано некоректно. У цьому випадку виникає помилка ValueError. Інакше викликається конструктор класу dict, у який передається осилання після парсингу. Код класу наведено на наступному рисунку.

```python
from urllib.parse import urlparse


class Node(dict):
    """
    Клас для збереження адрес інших користувачів
    """
    def __init__(self, link: str):
        """
        Парсить отриману адресу та зберігає в класі
        Наприклад http://127.0.0.1:5000
        :param link: адреса користувача
        """
        parsed_url = urlparse(link)
        if parsed_url.netloc:
            dict.__init__(self, link=parsed_url.netloc)
        elif parsed_url.path:
            dict.__init__(self, link=parsed_url.path)
        else:
            raise ValueError('Invalid URL')
```

Рисунок 4.7 – Код класу Node

У конструктор класу Transaction передається інформація про відправника, отримувача та кількість умовних грошових одиниць. Додаткової обробки ці дані не потребують, тому просто викликається конструктор класу dict з вищевказаними параметрами. Для обрахунку хеша транзакції використаємо функцію sha256 бібліотеки hashlib [26]. Аргументом цієї функції є рядок, тому нам необхідно змінити представлення транзакції. Як зазначалося, базові класи являють собою словник для представлення їх у форматі JSON. Тому було використано бібліотеку json та її методи dumps та encode для отримання рядкового представлення транзакції, яке буде аргументом функції sha256. Код класу наведено на наступному рисунку.

```python
from hashlib import sha256
import json


class Transaction(dict):
    """
    Клас для збереження транзакцій у мережі
    """
    def __init__(self, sender: str, recipient: str, quantity: float):
        """
        Конструктор класу
        :param sender: відправник
        :param recipient: отримувач
        :param quantity: кількість умовних одиниць
        """
        dict.__init__(self, sender=sender,
                      recipient=recipient, quantity=quantity)

    def get_transaction_hash(self) -> str:
        """
        Обраховує та повертає хеш транзакції, використовуючи хеш-функцію sha256
        :return: хеш даної транзакції
        """
        transaction_string = json.dumps(self, sort_keys=True).encode()
        return sha256(transaction_string).hexdigest()
```

Рисунок 4.8 – Код класу Transaction

У конструктор класу Block передаються всі аргументи блоку, окрім його хешу та кореня дерева Меркла транзакцій, адже вони потребують обрахування всередині класу блоку за принципом інкапсуляції. Корінь дерева Меркла знаходиться відповідно до його визначення: обраховується масив хешів транзакцій, а потім рекурсивно масиви суми хешів сусідніх елементів, допоки не залишиться один елемент масиву. Реалізацію наведено на наступному рисунку.

```python
def merkle(self, transaction: List[Transaction]) -> str:
    """
    Обраховує та повертає дерево Меркла транзакцій блоку
    :param transaction: транзакції, які зберігаються в блоці
    :return: значення хешу кореня дерева Меркла
    """
    if not transaction:
        return ''

    def tree(hashes: List[str]) -> str:
        """
        Рекурсивна функція обрахунку хешу кореня дерева Меркла
        :param hashes: масив хешів транзакцій
        :return: хеш кореня дерева Меркла
        """
        count = len(hashes)
        if count == 1:
            return hashes[0]
        return sha256(tree(hashes[:count // 2]).encode() + tree(hashes[count // 2:]).encode()).hexdigest()

    transaction_hashes = [tr.get_transaction_hash() for tr in transaction]
    return tree(transaction_hashes)
```

Рисунок 4.9 – Рекурсивна функція обрахунку дерева Меркла

Хеш блоку обраховується аналогічним методом, як і хеш транзакції. Перевірка правильності знайденого значення доказу роботи (proof) відбувається шляхом знаходження хешу блока та перевірки, щоб цей хеш починався з певної комбінації символів (NONCE). Код класу Block наведено в Додатку А.

У конструкторі класу Blockchain створюються змінні для збереження непідтверджених транзакцій, ланцюга блоків та вузлів мережі. Ці змінні є списками. Для додавання елементів до них створюється об’єкт відповідного класу, який і буде додано в список. У конструкторі також відбувається генерація генезисного блоку, який відразу додається до користувацького ланцюга. Передавати додаткові аргументи в конструктор не потрібно.

Перевірка коректності ланцюга блоків відбувається проходження в циклі всіх блоків ланцюга з перевіркою наступних критеріїв для кожного блоку:

- правильно вказаний хеш попереднього блоку

- хеш даного блоку обрахований вірно

- доказ роботи знайдено правильно

Якщо хоч в одному блоці якийсь із критеріїв не задовольняється, то й увесь ланцюг вважається не валідним.

Пошук головного ланцюга в мережі відбувається шляхом отримання ланцюгів блоків від кожного вузла мережі, перевірки їх коректності та знаходженням довжини. Якщо знайдено валідний ланцюг, більший за клієнтський, відбувається його заміна на знайдений.

У методі proof_of_work відбувається перевірка знайденого користувачем доказу роботи (передається аргументом при виклику методу). Для цього створюється об’єкт класу Block із вказанням усіх необхідних агументів та виконується викли його методу перевірки валідності доказу роботи. Якщо доказ валідний, клієнту повертається змайнений блок, інакше об’єкт типу None. Код класу Blockchain наведено в Додатку Б.

При запуску виконавчого файлу клієнта відбувається генерація його унікального ідентифікатора за допомогою методу uuid4 бібліотеки uuid та створюється об’єкт класу Blockchain. Ip-адресою за замовчуванням є 127.0.0.1 (локальний хост), а портом – 5000. Значення порту можна змінити при запуску програми вказанням відповідного аргументу в консолі.

Для майнінгу блоку клієнт спочатку додає до списку транзакцій транзакцію з винагородою для себе за створений блок. Ця транзакція не буде підтверджена, якщо якщо клієнт блок не видобуде. Це робиться для того, щоб у блоці було вказане надходження винагороди майнеру, а транзакція була врахована в дереві Меркла та хеші блоку. Після цього клієнт генерує випадкове значення доказу роботи та перевіряє його на валідність викликом відповідного

методу класу Blockchain. Процедура повторюється доти, доки клієнт не знайде потрібне значення. Після цього змайнений блок додається до ланцюга.

Для оновлення ланцюга (заміною на довший), додавання транзакцій та вузлів мережі реалізовані клієнтські методи, які викликають відповідні методи класу Blockchain. Також реалізований метод, який повертає клієнтський ланцюг блоків та його довжину. Це потрібно для обміну ланцюгами між вузлами мережі. Код класу наведено в додатку В.

4.4 Приклад роботи застосунку

Для наглядності роботи застосунку використовується програма Postman. Завдяки їй можна зручно створювати запити з даними у форматі JSON, а також переглядати відповіді на кожен надісланий запит.

Запустимо виконавчий файл клієнта в локальній мережі (рис. 4.10). Назвемо його Клієнт 1.

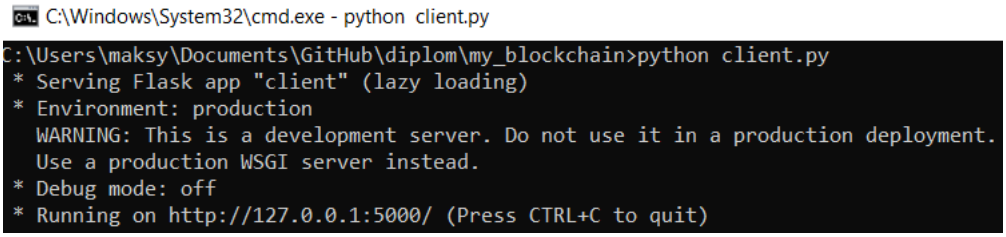

Рисунок 4.10 – Запуск виконавчого файлу клієнта

Переглянемо ланцюг блоків створеного користувача. Для цього надішлемо відповідний GET-запит (рис. 4.11). Як бачимо, його ланцюг складається з одного блоку – блоку генезу.

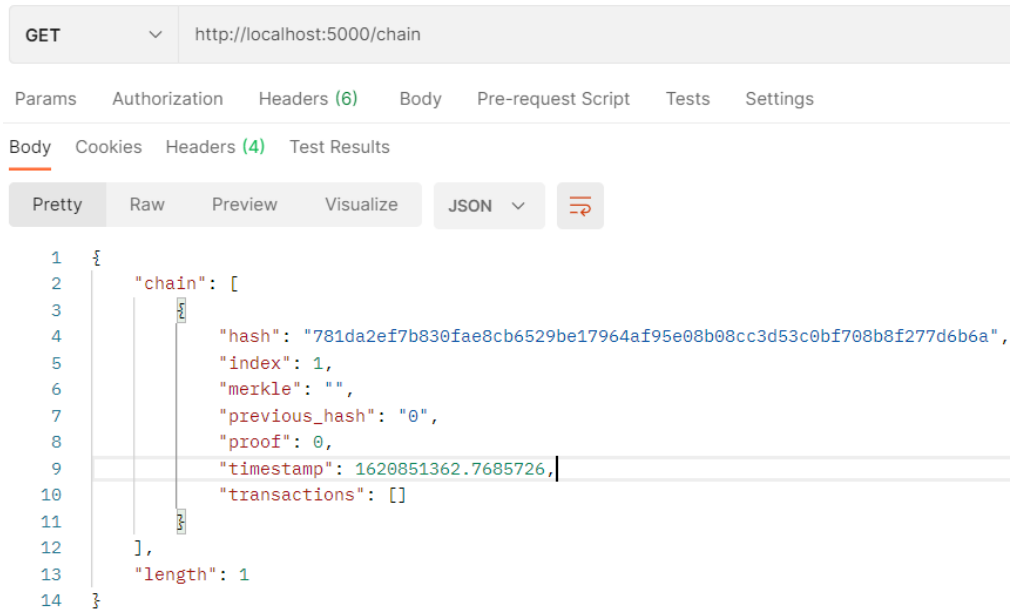

Рисунок 4.11 – GET-запит отримання ланцюга блоків користувача

Створимо новий блок майнінгом (рис. 4.12) та ознайомимося з його структурою. Як бачимо, попередній хеш відповідає хешу генезисного блоку. Хеш створеного блоку починається з «0», що є вказаним для прикладу параметром NONCE, тобто доказ роботи знайдено вірно. У списку транзакцій є лише одна, яка відповідає винагороді користувачу за створений блок. В полі отримувача транзакції знаходиться унікальний ідентифікатор клієнта. Відправник транзакції вказаний як «0», щоб помітити цю транзакцію винагородою за майнінг.

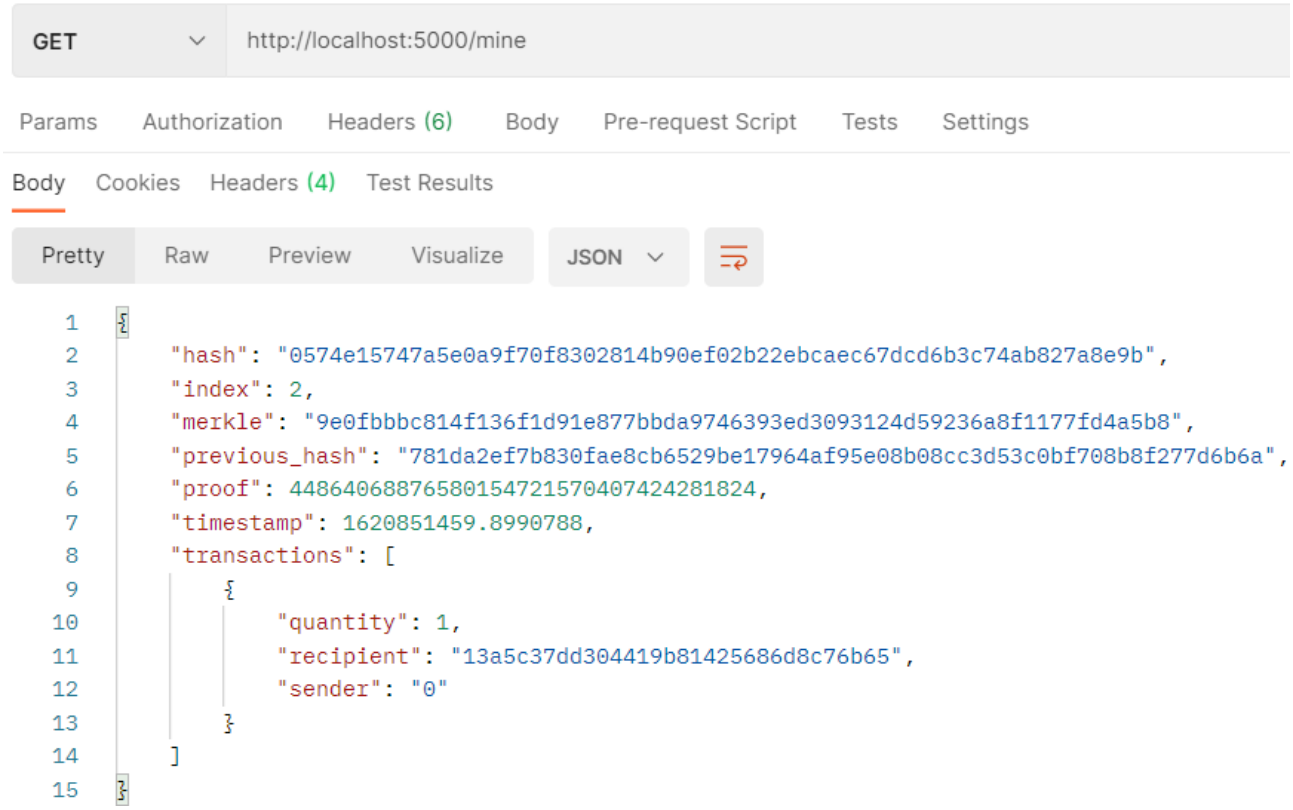

Рисунок 4.12 – Запит на створення блоку та його відповідь

Додамо нового користувача на порті 5555 (рис. 4.13). Це буде Користувач 2.

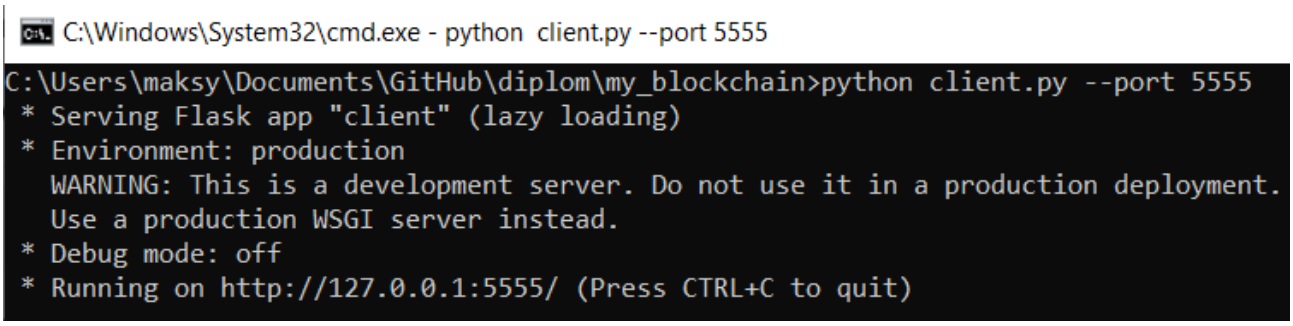

Рисунок 4.13 – Запуск виконавчого файлу нового користувача

Виконаємо POST-запит (рис. 4.14) із тілом у форматі JSON для вказання всіх інших вузлів мережі (наразі це лише Користувач 1), щоб оновити список адрес Користувача 2.

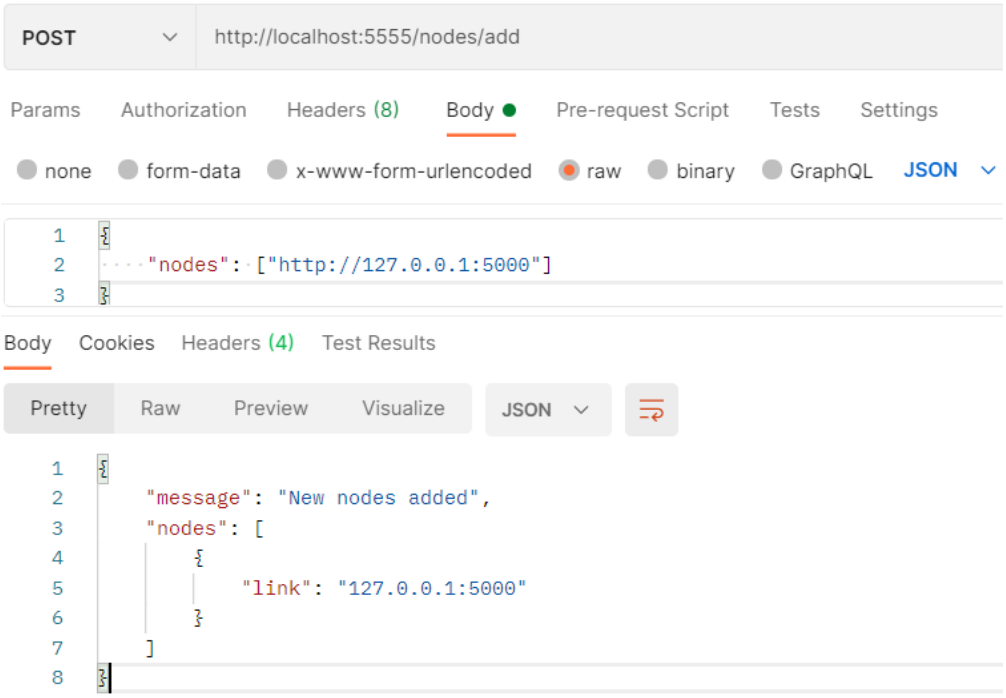

Рисунок 4.14 – Оновлення списку вузлів

Створимо транзакцію, в якій перешлемо частину коштів Користувача 1 Користувачу 2. Для цього виконаємо POST-запит, тілом якого є бажана транзакція (рис. 4.15). Як у відповіді на запит, транзакція буде додана до блоку з індексом 3, коли він буде змайнений.

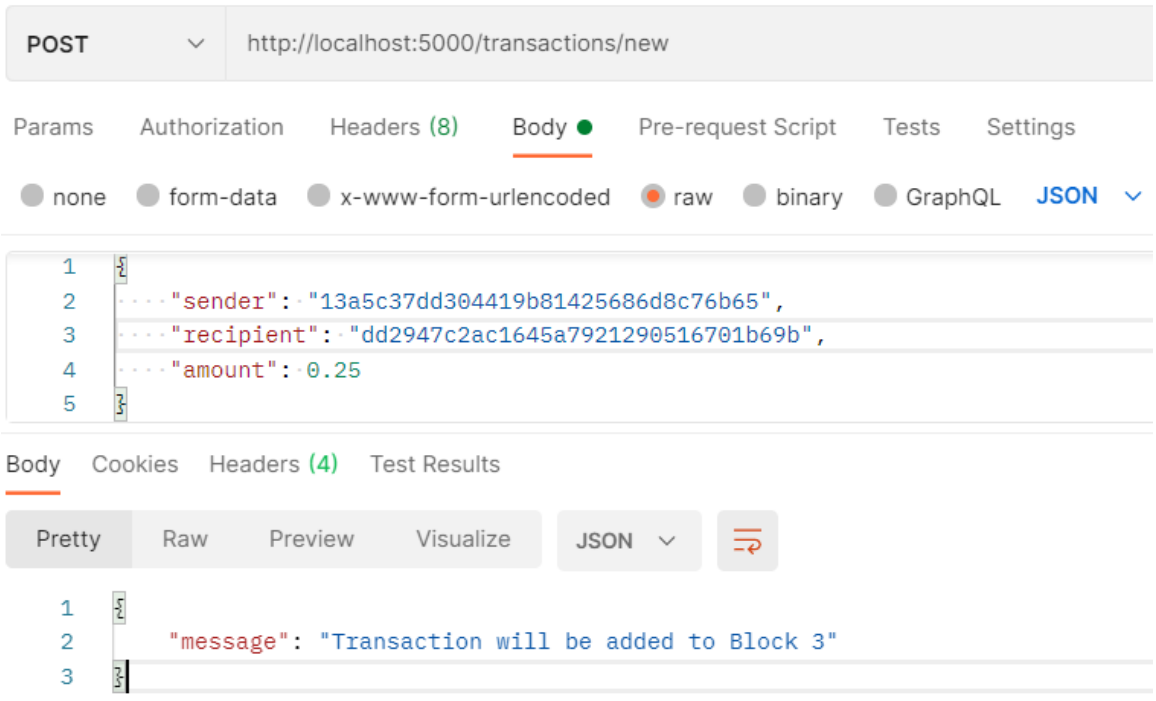

Рисунок 4.15 – Створення транзакції

Підтвердимо транзакцію шляхом майнінгу нового блоку. Для цього виконаємо відповідний запит (рис. 4.16). Як бачимо, у списку транзакцій наявна попередньо створена транзакція, а також винагорода Користувачу 1 за видобутий блок.

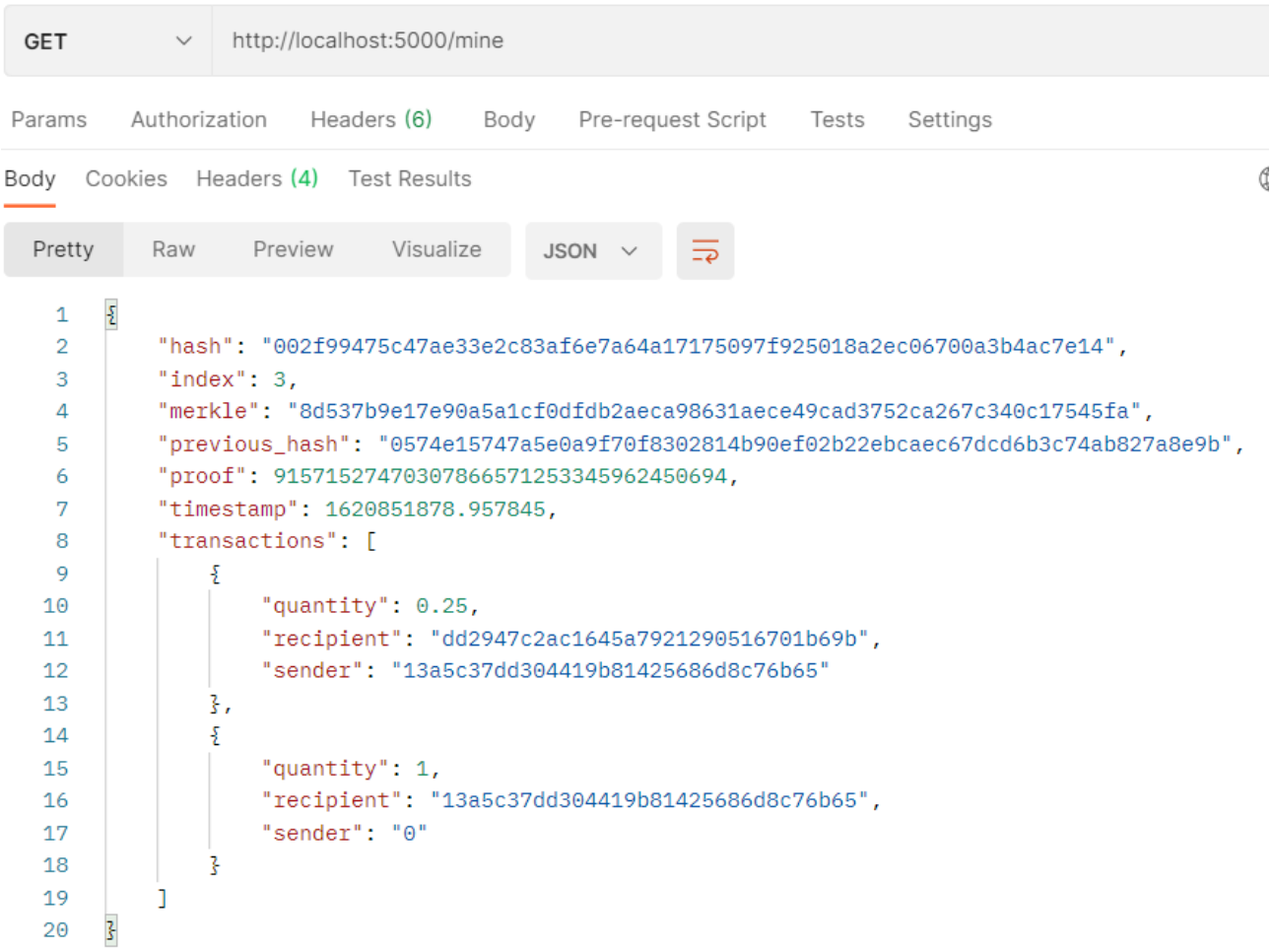

Рисунок 4.16 – Створення нового блоку для підтвердження транзакції

Щоб Користувач 2 побачив зміни в блокчейні, виконаємо запит на оновлення його ланцюга блоків (рис. 4.17). Наразі всі користувачі мережі мають актуальну версію ланцюга блоків.

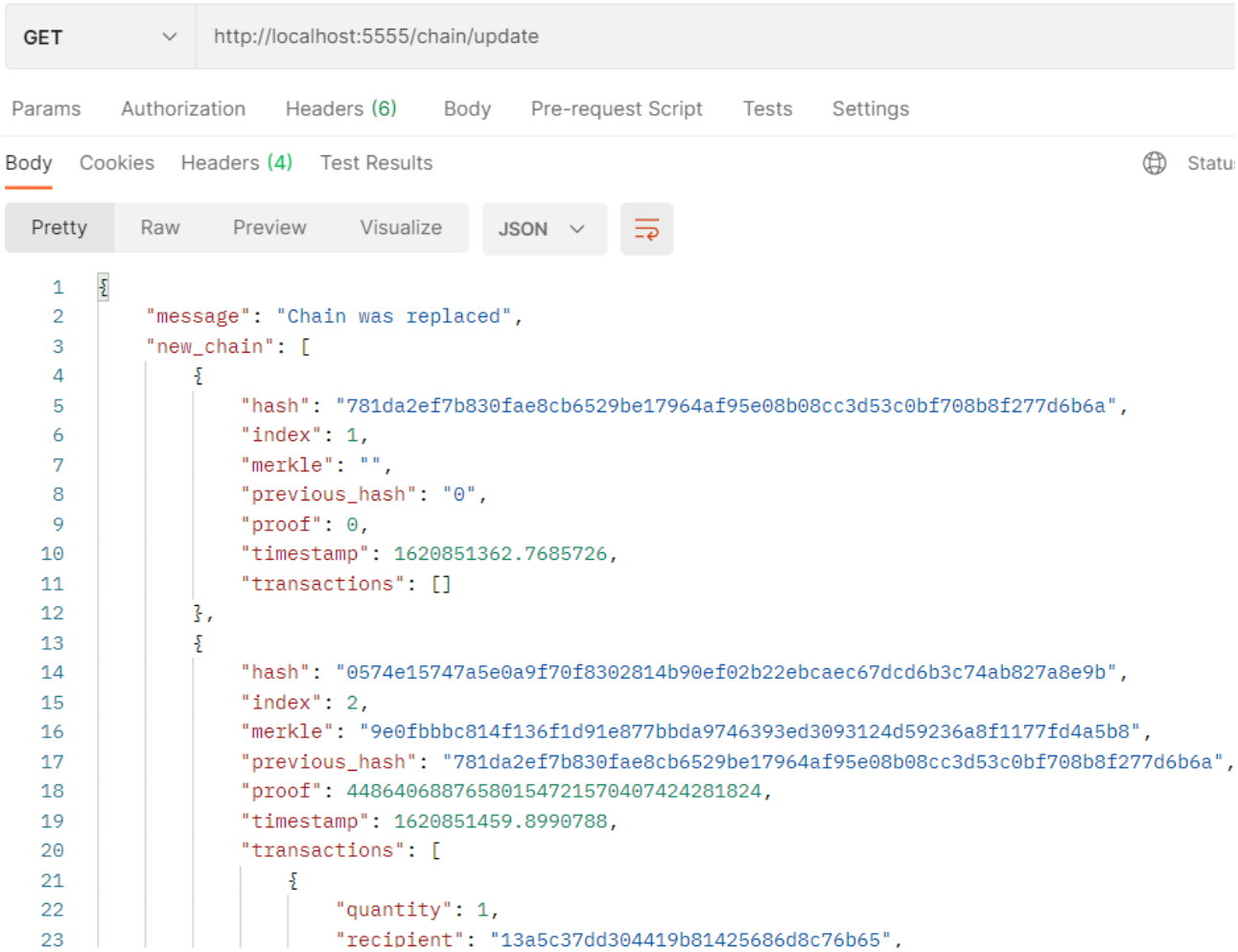

Рисунок 4.17 – Оновлення ланцюга блоків клієнта

Варто зазначити, що створювані нами запити відображаються в консолі, в якій запускали клієнтський виконавчий файл (рис. 4.18).

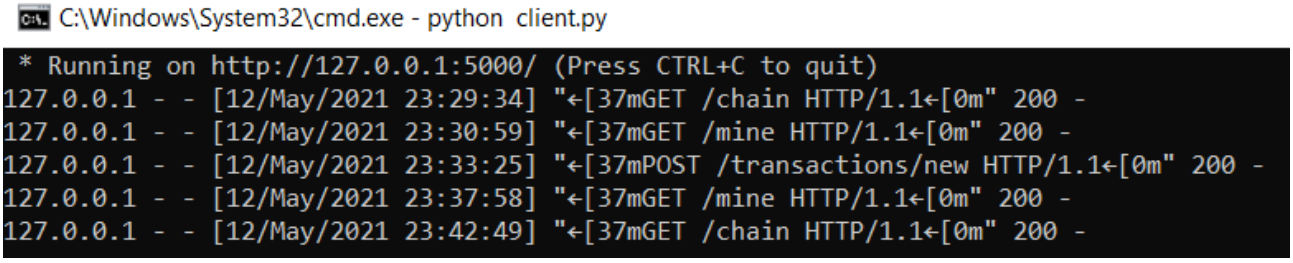

Рисунок 4.18 – Вивід консолі під час роботи застосунку

ВИСНОВКИ

Під час виконання роботи було досліджено історію та причини створення технологій розподіленого реєстру. Оглянуто сфери їхнього застосування, висвітлено доцільність користування ними. Було оглянуто, описано та проаналізовано наступні найбільш популярні технології розподіленого реєстру: Blockchain, DAG, Hashgraph і Holochain, а також криптовалюти – Bitcoin, Litecoin й Ethereum.

Було проведено порівняння структури та принципу роботи мереж криптовалют Litecoin й Ethereum із біткоїном. Також було проведено всестороннє порівняння технологій розподіленого реєстру з блокчейном, відмінності між якими оформлено у вигляді таблиці.

Завдяки попередньому дослідженню технології блокчейн було розроблено програмний застосунок «Клієнт мережі блокчейн» у вигляді імпортованих бібліотек та API, який можна застосовувати для побудови власної мережі блокчейн. Отриманий застосунок має модульну структуру, а саме окремі компоненти для збереження вузлів мережі, транзакцій, блоків та клієнтського ланцюга, а також виконавчий файл клієнта. Це дає можливість використовувати кожен компонент окремо та модифікувати його під власні потреби.

Отже, використання технологій розподіленого реєстру та блокчейну є виправданим, актуальним та популярним у різних сферах. Блокчейн і надалі продовжить розвиватися та здобуде ще більшу кількість варіантів застосування в повсякденному житті. Розроблений застосунок дає можливість будувати власні мережі блокчейну для довільної сфери застосування на розсуд розробника, є простим у розумінні та конкурентним серед подібних програмних рішень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PyCharm [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/pycharm/
2. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/
3. Postman [Електронний ресурс] – Режим доступу до ресурсу: https://www.postman.com/
4. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/
5. Welcome to Flask [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.0.x/
6. API [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/API
7. Satoshi Nakamoto – Bitcoin: A Peer-to-Peer Electronic Cash System, 2008
8. Manav Gupta – Blockchain, 3rd IBM Limited Edition.
9. Что такое технология блокчейна? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/ru-ru/topics/what-is-blockchain
10. Что такое блокчейн и критерии сравнения блокчейнов [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/sandbox/134048/
11. Types Of DLTs [Електронний ресурс] – Режим доступу до ресурсу: https://101blockchains.com/blockchain-vs-hashgraph-vs-dag-vs-holochain/
12. What is blockchain security? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/topics/blockchain-security
13. Транзакции Litecoin: как они работают? [Електронний ресурс] – Режим доступу до ресурсу: https://ecrypto.ru/kriptovalyuta/tranzaktsii-litecoin-kak-oni-rabotayut.html
14. Как работает Блокчейн (Blockchain)? [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/freecodecamp-russia-%D1%80%D1%83%D1%81%D1%81%D0%BA%D0%BE%D1%8F%D0%B7%D1%8B%D1%87%D0%BD%D1%8B%D0%B9/%D0%BA%D0%B0%D0%BA-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%B5%D1%82-%D0%B1%D0%BB%D0%BE%D0%BA%D1%87%D0%B5%D0%B9%D0%BD-blockchain-5262a2cd4842
15. Bitcoin vs. Litecoin: What's the Difference? [Електронний ресурс] – Режим доступу до ресурсу: https://www.investopedia.com/articles/investing/042015/bitcoin-vs-litecoin-whats-difference.asp
16. Directed acyclic graph [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Directed_acyclic_graph

17. Holochain: An end-to-end open source P2P app framework [Електронний ресурс] – Режим доступу до ресурсу: https://holochain.org/
18. Hedera Hashgraph: Hello future [Електронний ресурс] – Режим доступу до ресурсу: https://hedera.com/
19. Bitcoin [Електронний ресурс] – Режим доступу до ресурсу: https://bitcoin.org/en/
20. ЧТО ТАКОЕ ETHEREUM? [Електронний ресурс] – Режим доступу до ресурсу: https://ethereum.org/ru/what-is-ethereum/
21. Litecoin [Електронний ресурс] – Режим доступу до ресурсу: https://litecoin.org/
22. JSON [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JSON
23. What is Object Oriented Programming? OOP Explained in Depth [Електронний ресурс] – Режим доступу до ресурсу: https://www.educative.io/blog/object-oriented-programming
24. urllib.parse – Parse URLs into components [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/urllib.parse.html
25. Data Structures [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/tutorial/datastructures.html
26. hashlib — Secure hashes and message digests [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/library/hashlib.html
27. Merkle tree [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Merkle_tree
28. Що таке ефіріум (Ethereum)? [Електронний ресурс] – Режим доступу до ресурсу: https://mohito.com.ua/sho-take-ethereum/
29. Що таке Ethereum? [Електронний ресурс] – Режим доступу до ресурсу: https://academy.binance.com/uk/articles/what-is-ethereum
30. Litecoins vs bitcoin [Електронний ресурс] – Режим доступу до ресурсу: https://uk.classicfoxvalley.com/key/litecoins-vs-bitcoins/
31. Як і де перевірити транзакцію Лайткоін? [Електронний ресурс] – Режим доступу до ресурсу: http://gt-trader.com.ua/jak-i-de-pereviriti-tranzakciju-lajtkoin
32. The Economist – Blockchains: The great chain of being sure about thing, 31 жовтня 2015
33. Thulasiraman, K.; Swamy, M. N. S. (1992) – "5.7 Acyclic Directed Graphs", Graphs: Theory and Algorithms, John Wiley and Son, с. 118
34. The Hindu – Can hashgraph succeed blockchain as the technology of choice for cryptocurrencies?, 25 Березня 2018
35. Блок біткоіни: що це, структура, характеристики, приклад [Електронний ресурс] – Режим доступу до ресурсу: http://gt-trader.com.ua/blok-bitkoini-shho-ce-struktura-harakteristiki

36. Strydom, Moses; Buckley, Sheryl – AI and Big Data's Potential for Disruptive Innovation. IGI Global, липень 2019

37. Tapscott, Don; Tapscott, Alex – The Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World, березень 2016

38. Шевченко Максим – програмний застосунок «Клієнт мережі блокчейн» [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Gurdel/blockchain_network_client

ДОДАТКИ

ДОДАТОК А

Код класу Block

```python
from hashlib import sha256
import json
from transaction import Transaction

from typing import List


class Block(dict):
    """
    Клас для збереження блоків у блокчейні
    """
    NONCE = "0"  # Необхідний заголовок хешу блоку для PoW

    def __init__(self, index: int, time: float, transactions: List[Transaction], proof: int, prev_hash: str):
        """
        Конструктор класу
        :param index: порядковий номер блоку
        :param time: час створення блоку
        :param transactions: транзакції, які зберігаються в блоці
        :param proof: доказ роботи алгоритмом PoW
        :param prev_hash: хеш попереднього блоку
        """
        if transactions is None:
            transactions = []
        block = {
            'index': index,
            'timestamp': time,
            'proof': proof,
            'transactions': transactions,
            'merkle': self.merkle(transactions),  # хеш транзакцій, отриманий обрахунком дерева Меркла
            'previous_hash': prev_hash,
        }
        block['hash'] = self.get_block_hash()  # хеш даного блоку
        dict.__init__(self, **block)
    def get_block_hash(self) -> str:
        """
        Обраховує та повертає хеш блоку, використовуючи хеш-функцію sha256
        :return: значення хешу блоку
        """
        self['hash'] = ''
        block_string = json.dumps(self, sort_keys=True).encode()
        self['hash'] = sha256(block_string).hexdigest()
        return self['hash']
    def validate_proof(self) -> bool:
        """
        Перевірка коректності знайденого доказу роботи методом PoW
        :return: True, якщо доказ роботи знайдений правильно (хеш блоку починається з NONCE), інакше False
        """
        block_hash = Block.get_block_hash(self)
        return block_hash.startswith(Block.NONCE)
```

```python
    def merkle(self, transaction: List[Transaction]) -> str:
        """
        Обраховує та повертає дерево Меркла транзакцій блоку
        :param transaction: транзакції, які зберігаються в блоці
        :return: значення хешу кореня дерева Меркла
        """
        if not transaction:
            return ''

        def tree(hashes: List[str]) -> str:
            """
            Рекурсивна функція обрахунку хешу кореня дерева Меркла
            :param hashes: масив хешів транзакцій
            :return: хеш кореня дерева Меркла
            """
            count = len(hashes)
            if count == 1:
                return hashes[0]
            return sha256(tree(hashes[:count // 2]).encode() + tree(hashes[count // 2:]).encode()).hexdigest()

        transaction_hashes = [tr.get_transaction_hash() for tr in transaction]
        return tree(transaction_hashes)
```

ДОДАТОК Б

Код класу Blockchain

```python
from time import time
import requests
from typing import List

from block import Block
from transaction import Transaction
from node import Node


class Blockchain:
    """
    Клас для збереження ланцюга блокчейну
    """
    def __init__(self):
        """
        Конструктор класу
        """
        self.current_transactions = []  # Транзакції, які мають бути додані до наступного блоку
        self.chain = []  # Ланцюг блоків
        self.nodes = []  # Список адрес інших користувачів мережі

        # Створення початкового (генезисного) блоку
        self.new_block(Block(index=1, time=time(), transactions=[], proof=0, prev_hash=Block.NONCE))
```

```python
    def add_node(self, link: str):
        """
        Додає нову адресу користувача до списку
        :param link: адреса користувача
        """
        self.nodes.append(Node(link))
```

```python
def validate_chain(self, chain: List[Block]) -> bool:
    """
    Перевіряє коректність ланцюга блоків. Критерії:
    - правильно вказаний хеш попереднього блоку
    - хеш даного блоку обрахований вірно
    - доказ роботи знайдено правильно
    :param chain: ланцюг блоків для перевірки
    :return: True, якщо ланюг коректний, інакше False
    """
    for i in range(1, len(chain)):
        print(f'{chain[i - 1]}')
        print(f'{chain[i]}')
        print("\n-----------\n")

        if chain[i]['previous_hash'] != chain[i - 1]['hash'] or \
                chain[i]['hash'] != Block.get_block_hash(chain[i]) or \
                not Block.validate_proof(chain[i]):
            return False

    return True
```

```python
def find_main_chain(self) -> bool:
    """
    Пошук найдовшого коректного ланцюга блоків у мережі
    :return: True, якщо знайдено ланцюг, довший за наший, інакше False
    """
    new_chain = None
    max_chain_length = len(self.chain)

    # Перевіряємо ланцюги блоків інших користувачів
    for node in self.nodes:
        response = requests.get(f'http://{node["link"]}/chain')  # Запит на отримання ланцюга іншого користувача
        length = response.json()['length']
        chain = response.json()['chain']

        # Якщо ланцюг валідний та має більшу довжину, замінюємо наший ланцюг
        if length > max_chain_length and self.validate_chain(chain):
            max_chain_length = length
            new_chain = chain

    # Зміна нашого ланцюга на довший
    if new_chain:
        self.chain = new_chain
        return True

    return False
```

```python
def new_block(self, block: Block) -> Block:
    """
    Додає новий блок до нашого ланцюга
    :param block: змайнений блок
    :return: цей же блок
    """
    # Транзакції підтверджені, очищаємо список транзакцій
    self.current_transactions = []

    self.chain.append(block)
    return block
```

```python
    def new_transaction(self, sender: str, recipient: str, quantity: float):
        """
        Додає нову транзакцію
        :param sender: відправник
        :param recipient: отримувач
        :param quantity: кількість умовних одиниць
        :return: індекс блоку, до якого буде додана транзакція
        """
        self.current_transactions.append(
            Transaction(
                sender=sender,
                recipient=recipient,
                quantity=quantity
            ))

        return self.chain[-1]['index'] + 1

    def proof_of_work(self, proof: int):
        """
        Перевірка знайденого доказу роботи алгоритмом PoW
        :param proof: доказ роботи користувача, який необхідно перевірити
        :return: підтверджений блок, якщо доказ роботи знайдено вірно, інакше None
        """
        template = {
            'index': len(self.chain) + 1,
            'time': time(),
            'transactions': self.current_transactions,
            'proof': proof,
            'prev_hash': self.chain[-1]['hash']
        }
        block = Block(**template)
        if block.validate_proof():
            return block
        return None
```

ДОДАТОК В

Код клієнтського файлу

```python
"""
Виконавчий файл клієнта мережі блокчейн
"""
from argparse import ArgumentParser
from uuid import uuid4
from flask import Flask, jsonify, request
import random

from blockchain import Blockchain

app = Flask(__name__)
user_id = str(uuid4()).replace('-', '')  # Генерує ідетифікатор клієнта
blockchain = Blockchain()
```

```python
@app.route('/mine', methods=['GET'])
def mine():
    """
    Майнінг нового блоку
    :return: змайнений блок
    """

    # Додає транзакцію з винагородою для користувача за змайнений блок
    blockchain.new_transaction(
        sender="0",  # Вказує, що це винагорода за майнінг
        recipient=user_id,
        quantity=1,
    )

    # Пошук значення доказу роботи для алгоритма PoW
    proofed_block = None
    while proofed_block is None:
        proof = random.randint(0, 99999999999999999999999999999999)
        proofed_block = blockchain.proof_of_work(proof)

    blockchain.new_block(proofed_block)

    return jsonify(proofed_block)


@app.route('/transactions/new', methods=['POST'])
def new_transaction():
    """
    Створює нову транзакцію
    :return: Повідомлення з номером блоку, в який буде додано транзакцію
    """
    transaction = request.get_json()
    index = blockchain.new_transaction(transaction['sender'], transaction['recipient'], transaction['amount'])
    response = {'message': f'Transaction will be added to Block {index}'}
    return jsonify(response)
```

```python
@app.route('/chain', methods=['GET'])
def chain():
    """
    Метод для отримання ланцюга блоків користувача
    :return: ланцюг блоків та його довжина
    """
    response = {
        'chain': blockchain.chain,
        'length': len(blockchain.chain),
    }
    return jsonify(response)
```

```python
@app.route('/nodes/add', methods=['POST'])
def add_nodes():
    """
    Додає адреси інших користувачів до списку адрес користувача
    :return: повідомлення зі списком усіх адрес інших користувачів
    """
    nodes = request.get_json()

    nodes = nodes.get('nodes')
    if nodes is None:
        return "Error: Please supply a valid list of nodes"

    for node in nodes:
        blockchain.add_node(node)

    response = {
        'message': 'New nodes added',
        'nodes': list(blockchain.nodes),
    }
    return jsonify(response)
```

```python
@app.route('/chain/update', methods=['GET'])
def update_chain():
    """
    Знаходить головний ланцюг (валідний та найдовший) у мережі
    :return: ланцюг блоків після оновлення
    """
    replaced = blockchain.find_main_chain()

    if replaced:
        response = {
            'message': 'Chain was replaced',
            'new_chain': blockchain.chain
        }
    else:
        response = {
            'message': 'Chain is main',
            'chain': blockchain.chain
        }

    return jsonify(response)


if __name__ == '__main__':
    parser = ArgumentParser()
    parser.add_argument('--port', default=5000, type=int, help='port to listen on')
    args = parser.parse_args()

    port = args.port
    app.run(host='127.0.0.1', port=port)
```