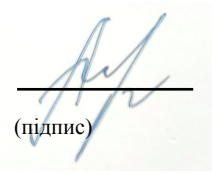
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота
на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

БЛОКЧЕЙН ТЕХНОЛОГІЯ ТА ЇЇ ЗАСТОСУВАННЯ

Виконав студент 4-го курсу
Наумович Артем Олександрович


(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
«_____»_____ 2022 р.
протокол №_____

Київ - 2022

РЕФЕРАТ

Обсяг роботи 36 сторінок, 11 рисунків, 16 джерел посилання.

БЛОКЧЕЙН, ХЕШ-ФУНКЦІЇ, ДЕРЕВО МЕРКЛЯ, РОЗПОДІЛЕНИЙ РЕЄСТР, КРИПТОВАЛЮТИ, БІТКОЇН МЕРЕЖА, NON-FUNGIBLE TOKEN, PYTHON.

Об’єктом роботи є дослідження технології блокчейну, її застосуванні в криптовалютах, системах з розподіленими реєстрами, виявлення переваг та недоліків у порівнянні з нерозподіленими варіантами.

Метою роботи є створити розподілену систему з використанням блокчейну, в якому зберігатимуться як класичні транзакції про переказ коштів так і про створення та купівлю невзаємозамінних токенів, а саме, коротких висловлювань.

Методи розробки: аналіз загальних принципів блокчейну та його застосування на прикладі біткоїн мережі, реалізація застосунку з базовим функціоналом майнингу та переказу коштів, розширення до версії зі створенням повідомлень від користувачів та їх купівлею.

Результати роботи: проведено аналіз технології блокчейну та прикладів її застосування, розроблено програмний застосунок засобами мови Python, що являє собою API, яке може бути запущене на різних вузлах та дозволяє переказувати кошти між рахунками, майнити блоки та отримувати за це винагороду, створювати висловлювання від користувачів, купувати їх та встановлювати нову ціну.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API - Application Programming Interface

HTTP - Hypertext Transfer Protocol

JSON - JavaScript Object Notation

NFT - Non-Fungible Token

ВСТУП

Оцінка сучасного стану об’єкта розробки. Не можна не погодитись, що з кожним роком роль інформаційних технологій у житті людей стає все більшою. Невпинно збільшуються об’єми інформації, частина з якої є вкрай чутливою. І навіть на перший погляд непомітна заміна хоча б одного символу в базі даних, може призвести до явної помилки в кращому випадку, а в гіршому - до непередбачуваних наслідків у вигляді припинення роботи всієї системи або зміни стану всіх її користувачів. І навіть не завжди тим, хто спричинить помилку буде стороння особа або шкідливе програмне забезпечення. В 2022 році довіряти не можна навіть собі. Добре що існує технологія блокчейну, яка допомагає вирішити певні проблеми, захистити данні від непотрібних змін та спроб їх сфальсифікувати. Технологія використовується в широкому спектрі програмних застосунків, так як є лише загальним принципом зберігання даних.

Актуальність роботи та підстави для її виконання. Блокчейн сам по собі не є новою технологією, але його використання є вкрай перспективним в купі з грамотною роботою. Тому що спектр галузей, де можно його застосувати вперше, доволі широкий та, ймовірно не обмежений. І навіть у більшості випадків, новий продукт може нав’язати конкуренцію лідеру ринку.

Мета й завдання роботи. Метою кваліфікаційної роботи є деталі вивчити технологію блокчейну. Дослідити внутрішні складові та принципи їх роботи, на рівні хеш-функцій та алгоритмів доказу роботи. Як результат, створити API - прототип розподіленої системи, яка у блокчейні зберігала б інформацію про різни види транзакцій:

- Майнинг генезис-блоку
- Переказ між рахунками коштів (взаємозамінні токени)
- Створення повідомлення від користувача (невзаємозамінні токени)
- Купівля повідомлень та встановлення покупцем нової ціни

Методи й засоби розробки. Розробці програмного засобу передувало поглиблення знань у вибраній темі. Було проведено аналіз існуючих алгоритмів та виявлено їх сильні та слабкі місця.

Можливі сфери застосування. Створене API є лише прототипом повноцінної системи, але після вдосконалення певних своїх частин, може стати конкурентоспроможним гравцем на безжальному ринку інноваційних програмних застосунків. Можливість торгувати невзаємозамінними токенами вже знаходить свій відгук у серцях колекціонерів та поціновувачів мистецтва.

1 ХЕШ-ФУНКЦІЇ

1.1 Загальний опис хеш-функцій

Тема блокчейну не є простою та потребує попередніх знань. Тож перед тим як перейти до опису концепції блокчейну, опишемо ряд основних принципів та сутностей, на які будемо спиратися надалі при розгляданні більше складних речей. І першим розглянемо поняття хеш-функції.

Хеш-функція - це функція, яка на вхід приймає одне число, яке будемо називати ключем, і повертає також деяке число. Зазвичай, число результат використовують у якості індексу для пошуку в базі даних. Проте в блокчейні результат роботи хеш-функції набуває дещо іншого значення. Тобто, іншими словами хеш-функція потрібна для того щоб відобразити одне значення в інше. У випадку, якщо для якоїсь хеш-функції різні значення ключей переходять в різні результати, то такі хеш-функції називають ідеальними. Побудувати їх так, щоб можна було використовувати для дійсно практичних проблем, може бути складно. Частіше хочеться уникати випадків співпадіння результатів хешування для різних ключів, які називаються колізіями, тому задача полягає в тому, аби віднайти таку хеш-функцію, для якої колізії будуть траплятися як можна менше.

Хеш-функція повинна мати деякі властивості[1]. Перш за все, вона повинна ефективно обчислюватися. З одного боку це залежить від потужностей комп’ютера, але з іншого слід з розумом підходити до структури функції. Так як хеш-функції викликаються неймовірно часто, то час на знаходження результату повинен бути нормальним на будь-якій машині, яка здатна задовольняти найпростішим потребам користувачів.

Іншою властивістю, яка частково вже було згадана раніше, є важливість рівномірного розподілу результатів хешування. Або ж іншими словами можна сказати так: відображення кожного можливого ключа в будь-який результат має бути рівномірним.

У загальному випадку хешувати можна зовсім не тільки числа, а і будь-яку інформацію. Але як би там не було, будь-що буде, можливо, неявно, але спочатку приведене до числа - ключа. Якщо треба обчислити хеш для бінарного об’єкту або

файлу, алгоритм перетворення входу до числа-ключа, є тривіальним. Для якихось інших структур алгоритм може відрізнятися. Наприклад якщо ми працюємо з датами, як парами чисел, то можемо, скажімо, до числа, що позначає місяць, приписати справа день. Для дат 10 листопада та 24 серпня матимемо 1110 та 824, відповідно.

Наведемо приклад якоїсь простої хеш-функції. Отримуючи число на вхід, будемо повертати кілька, скажімо 3, символи ключів у будь-якій системі числення. В цьому випадку якщо ключами будуть, наприклад, заробітні плати, матимемо багато колізій через те, що не рідко заробітна плата складається з якоїсь цілої кількості тисяч грошових одиниць. Тоді часто результатом роботи такої хеш-функції буде 0.

Враховуючи досвід попереднього прикладу, побудуємо іншу хеш-функцію, яка тепер буде в якості результату повертати вже перші 3 цифри від ключа[2]. Але і такий варіант є зовсім не практичним, так як, наприклад, для ключів, які є номерами телефонів, значення хешу дуже часто буде 380 для українських номерів телефонів, 482 - для польських номерів.

Наявність великої кількості колізій не завжди є помилкої, а може слугавати, наприклад, як спосіб неявно розділити дані на блоки за певною ознакою. Але використовувати хешування в такому випадку треба вкрай обережно і зовсім не варто це робити в контексті роботи з блокчейном.

Можемо вдосконалити перший варіант іншим чином та отримати на порядок кращу ситуацію. Назвемо це методом ділення.

$$h(K) = K \bmod M$$

Якщо оберемо $M = 1000$, ніякої користі не отримаємо, навіть більше, отримаємо випадок з першого прикладу. Отже для методу ділення треба уважно та ретельно підходити до вибору значення числа M. Якщо взяти M парним, тоді для ключів, які теж є парними числами, результат також буде парним. Навіть таке зауваження може нашкодити тому, хто використовуватиме цю хеш функцію. Якщо обирати модуль на основі розрядності комп’ютера, то так як і для $M = 1000$, отримаємо хеш-функцію з самого першого прикладу тільки з тією різницею, що

тепер основа числення не десятична, а двійкова. На практиці, досить ефективним є варіант, у якому M - це просте число.

В якості вдосконалення попереднього випадку, наведемо наступну хеш-функцію [3]. Модулем можемо також обрати якесь просте числе, але і степінь двійки тепер підійде, а з боку ефективності обчислень, раціональним рішенням буде використовувати саме степінь двійки в якості M. У даному варіанті працюватимемо з K як з бінарними ключами.

$$K = k_{n-1}k_{n-2}\dots k_0$$

Тоді обчислимо значення поліному з коефіцієнтами, що відповідають ключу K, в деякій завжди фіксованій точці x.

$$h(K = k_{n-1}k_{n-2}\dots k_0) = h_{n-1}x^{n-1} + \dots + h_1x + h_0$$

Можна використовувати коефіцієнти і в зворотньому порядку без втрати ефективності.

$$h(K = k_{n-1}k_{n-2}\dots k_0) = h_0x^{n-1} + \dots + h_{n-2}x + h_{n-1}$$

1.2 Криптографічні хеш функції

Хоча й остання з розглянутих хеш-функцій є досить надійною та може бути використана для розв'язку задач олімпіадного програмування, але для роботи з блокчейном такої надійності не завжди достатньо. Блокчейн потребує більш досконалих хеш-функцій, а тому в даному розділі буде розглянутий клас криптографічних хеш-функцій.

Криптографічні хеш-функції [4] задовольняють ряду умов:

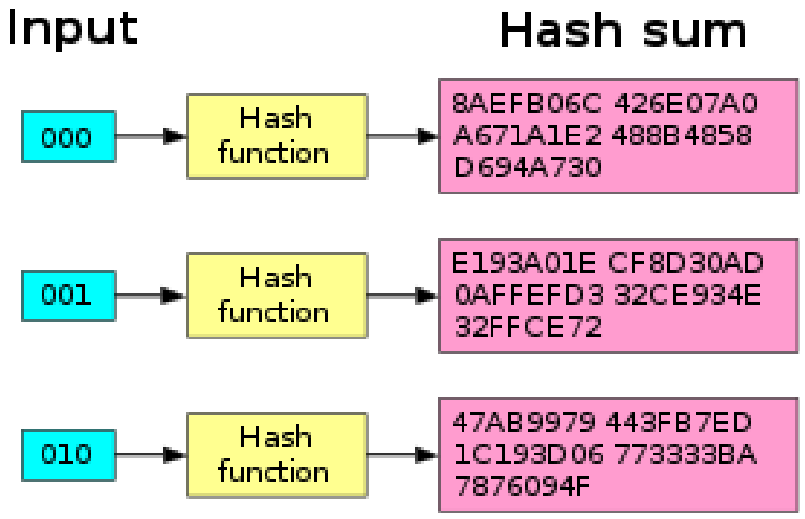
- незворотність: маючи значення хешу $h = h(K)$, повинно бути певною мірою важко знайти значення ключа K, щонайменше за сприйнятливий час
- стійкість до колізій першого роду: маючи ключ K_1, повинно бути складно знайти $K_2 \neq K_1$ таке, для якого виконується рівність $h(K_1) = h(K_2)$

- стійкість до колізій другого роду: має бути неможливо за нормальний час відшукати пару ключів K_1, K_2 для яких виконується рівність

  $h(K_1) = h(K_2)$

Крім зазначених правил, важливо відмітити, що криптографічні хеш-функції, як і інші, повинні бути детермінованими. Адже якщо це не так, переваги, які хешування надає, просто нівелюються.

Також слід згадати лавиновий ефект [5], який повинна мати гарна криптографічна хеш-функція. Це означає, що при навіть маленькій незначній зміні бітів ключа, значення хешу зміниться досить суттєво(див. рис. 1.1). Даний термін ввів у використання Хорст Фейстель.



“Рисунок 1.1 - приклад хеш функції з лавиним ефектом”

Можна сформулювати наступні умови лавиновості:

- сувора умова лавинововості: при зміні одного біту ключа, кожен біт значення хеша зміниться з ймовірністю 50 відсотків
- умова незалежності бітів: $\forall\ i,\ j,\ k$: при зміні біта i, біти j та k мають змінюватися незалежно один від одного

Наведемо список деяких добре відомих та часто використовуваних криптографічних хеш-функцій:

- MD4 [6]. Хеш-функція була розроблена в 1990 році Рональдом Рівестом. Довжина результату обчислення є 128 біт. І хоч даний

варіант є гарним прикладом з точки зору швидкості знаходження хешу, але в 1995 році була вперше опублікована атака на MD4

- MD5 [7]. Є найбільш поширеним прикладом зі свого класу. Частково функція схожа на MD4, але має не 3, а 4 цикли по 16 кроків кожен. MD4 вже є більше надійною хеш-функцію, але така перевага над молодшим братом коштувала третиною часу на обчислення. У 2006 році була опублікована атака на MD5, яка дозволяє знаходити колізії менш як за 1 хвилину
- SHA-1(Secure Hash Algorithm) [8]. На відміну від MD5, який створює 128-бітний хеш, SHA-1 - повертає в якості результату 160-бітне число. Хоча й функція створена була з ціллю використовувати її для роботи з криптовалютами, з часом були знайдені недоліки, які знецінили плюси SHA-1.
- SHA-2 [9]. Має значні відмінності від попередньої версії SHA-1. Класс складається з шести хеш-функцій: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256. Вперше SHA-2 був опублікований Національним інститутом стандартів і технологій як федеральний стандарт США. Більш того, класс алгоритмів SHA-2 запатентований США.

1.3 SHA-256

Алгоритм SHA-256 [10] є одним з представників класу хеш-функцій SHA-2. Він особливий тим, що на даний момент не був оприлюднений ключ до нього. Своє місце алгоритм зайняв у багатьох напрямках. І перш за все такі алгоритми як SHA-2 є частиною процесу генерації криптографічних елементів для спеціальних файлів-сертифікатів, які гарантують безпечний зв'язок веб-сервісів та веб-сайтів.

Раніше для генерації сертифікатів використовувався інший алгоритм - SHA-1. Але з часом він ставав менш надійним та більше не міг гарантувати безпеку на такому рівні, як того потребували користувачі. Тому в 2016 в індустрії сертифікатів стався перехід на новий стандарт - SHA-2. Було перегенеровано

тисячі сертифікатів, а в наступні роки переважна більшість сертифікатів SHA-1 просто зникла.

Перший варіант алгоритму SHA-256 був створений Агентством національної безпеки США навесні 2002 року. Кількома місяцями пізніше Національний метрологічний університет опублікував нещодавно анонсований протокол шифрування в стандарті безпечної обробки даних, який був поповнений другою версією алгоритму вже взимку 2004 року.

Цей протокол працює з інформацією, яка розбита на частини по 512 біт (64 байт), та як результат повертає 256-бітний хеш-код. SHA-256 включає відносно простий раунд(див.рис.1.2), який повторюється 64 рази.

SHA-256 має досить непогані технічні параметри:

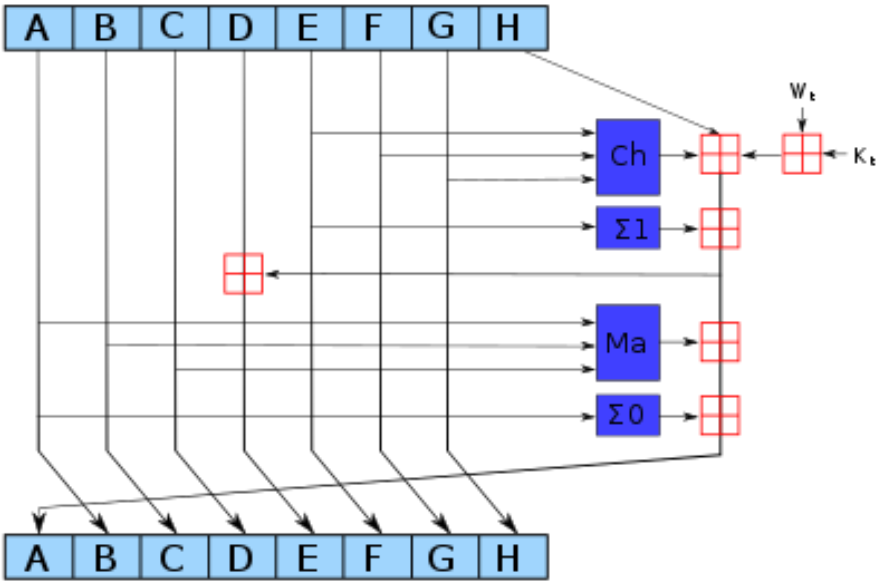
- індикатор розміру блоку: 64 байт
- максимально можлива довжина повідомлення: 33 байт
- характеристика розміру підпису повідомлення: 32 байт
- стандартний розмір слова: 4 байт
- внутрішній параметр довжини позиції: 32 байт
- кількість ітерацій в одному циклі: 64
- швидкість досягнута протоколом: близько 140 Міб/с

Алгоритм SHA-256 заснований на методі побудови Меркла-Дамгарда, згідно з яким початковий індекс відразу після внесення зміни розбивається на блоки, а ті, на 16 слів.

Багато криптовалют створені з використанням алгоритму SHA-256. Далі наведено список деяких з них:

- Bitcoin (BTC)
- Bitcoin SV (BSV)
- Bitcoin Cash (BCH)
- DigiByte (DGB)
- Syscoin (SYS)
- Namecoin (NMC)
- Peercoin (PPC)

- Litecoin Cash (LCC)
- Bitcoin Classic (BXC)
- Elastos (ELA)
- Auroracoin (AUR)
- Bitcoin Vault (BTCV)
- Super Bitcoin (SBTC)
- Pyrk (PYRK)
- EmerCoin (EMC)
- HTMLCOIN (HTML)
- Terracoin (TRC)
- Myriad (XMY)



“Рисунок 1.2 - Схема однієї ітерації обробки блоку даних алгоритмом SHA-256”

2 ДЕРЕВО МЕРКЛЯ

Наразі можна було б перейти безпосередньо до опису концепції блокчейну, проте це було б неповноцінно без згадки про поняття дерева Меркля.

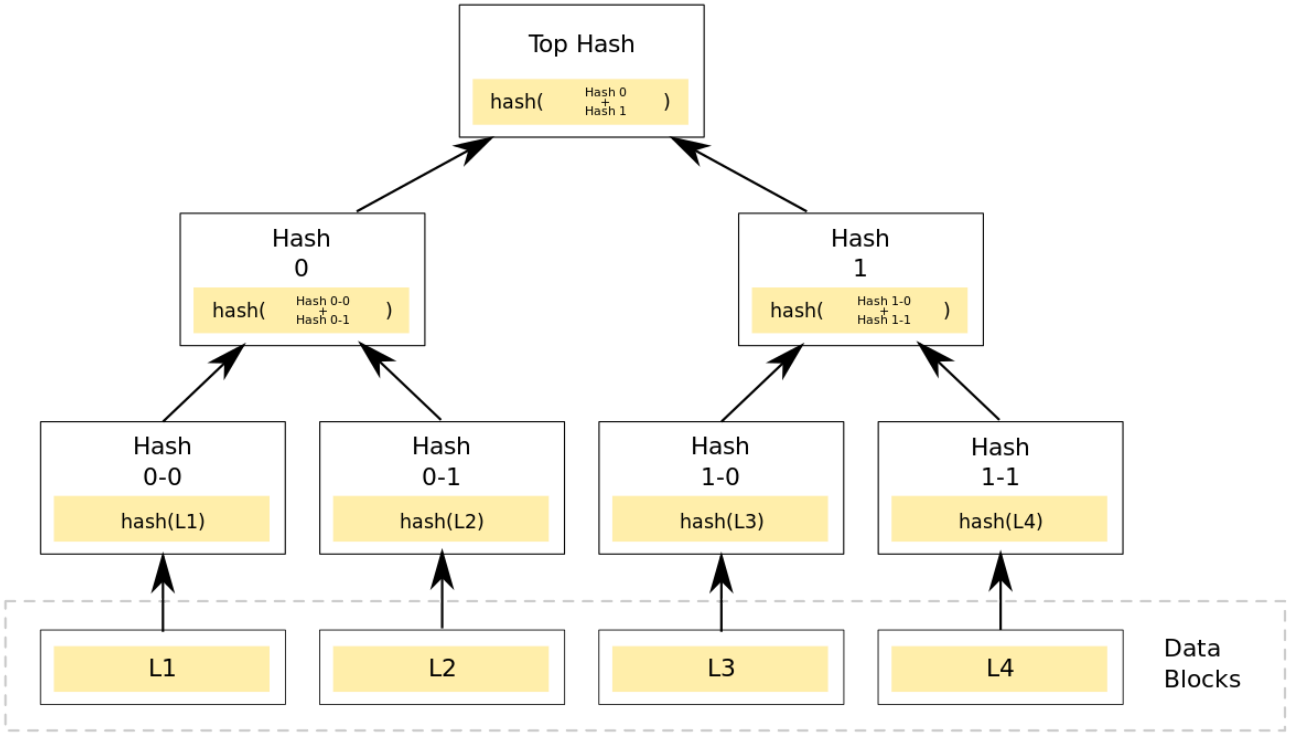
Концепція дерева Меркля [11] була запропонована Ральфом Мерклем на початку 1980-х років. Ральф Меркл був вченим-інформатиком, якого більшість знають за його роботами в галузі криптографії з відкритим ключем.

Дерево Меркля - це структура, яка використовується для верифікації даних в наборі. В основі лежать хеш-функції, яким був присвячений попередній розділ роботи.

Уявімо ситуацію, що нам необхідно завантажити деякий великий файл. Можна перевірити чи співпадає хеш файлу, який ми отримали, з хешем, який опублікований власниками файлу. І якщо хеші однакові, наші два файли однакові. А якщо ж різні, то маємо два варіанти. Або ж ми завантажили шкідливе програмне забезпечення, яке було замасковане під звичайний файл, або ж при завантаженні виникли якісь помилки та ми отримали файл, не аналогічний оригіналу. Як би там не було, але зрозуміло що файл треба буде завантажувати ще раз. І добре якщо це займає до кількох хвилин часу. А якщо ми занатажуємо великий архів даних та вимушені чекати кілька годин. Звісно не хотілося б знову витрачати так багато часу в очікуванні. Саме тут і можуть бути корисними дерева Меркля. Головна ідея полягає у тому, що великий файл можна на початку розбити на багато блоків, а далі спробувати відшукати пошкоджений фрагмент та провести завантаження тільки цього елемента.

Перейдемо до опису самої структури. Дерево Меркля являє собою бінарне дерево хешів [12], в якому в листах зберігається інформація про блоку файлу, або цілі файли якогось набору. Кожен інший вузол, крім листків, є хешем обох своїх нащадків. $Hash_0$ є хешем від конкатенації хешів елементів L_1 та L_2, а $Top\ Hash$ є хешем від конкатенації хешів $Hash_0$ та $Hash_1$.

Частіше хеш-дерева реалізуються за допомогою двійкових дерев(див. рис. 2.1), але це не обов’язково та можуть буть дерева з вузлами, які використовують багато інших дочірніх вузлів. Ще одним не обов’язковим правилом є використання такої криптографічної хеш-функції як SHA-2.



“Рисунок 2.1 - структура дерева Меркля для набору з 4 елементів”

У верхній частині хеш-дерева міститься кореневий хеш. Перед завантаженням файлу до публічного сховища даних повинен бути обчислений верхній хеш, отриманий з надійного джерела. Після цього будь який вузол мережі або ж будь-який користувач може обчислити все дерево Меркля, маючи послідовність блоків, та перевірить тільки верхній блок для верифікації цілісності інформації. Це базується на тому, що будь-яка зміна блоків даних спричиняє зміни хешів цілої гілки дерева, включно самого кореня.

Структура дерева Меркля дозволяє на зберігати всі хеші постійно, а тільки верхній хеш. Будь хто зможе в будь-який момент часу побудувати все дерево. Наприклад цілісність блоку L_1 можно перевірити без зайвих обчислень.

Нехай був пошкоджений тільки блок L_4. Тоді кореневий хеш точно не співпадатиме. Тож спробуємо спуститися рекурсивно вниз по дереву до дочірніх вузлів. Піддерево $Hash_0$ пошкоджень не має, а тоді і хеші співпадуть. Тобто продовжувати рекурсивний спуск сенсу на має. Натомість $Hash_1$ буде відрізнятися. Проробивши аналогічний рекурсивний спуск виявимо пошкоджний блок L_4. Таким чином можемо за $O(logN)$ віднайти невірний фрагмент даних.

3 БІТКОЇН МЕРЕЖА

3.1 Загальні відомості про блокчейн

Блокчейн - це інноваційна технологія баз даних [13], яка є фундаментом для майже всіх криптовалют. Ідентична база даних розповсюджується по всім вузлам мережі, через що неймовірно ускладнюється обман або злам системи. Хоча й індустрія криптовалют тісно пов'язана з поняттям блокчейну, не слід плутати ці два терміни. Технологія блокчейну пропонує потенціал для обслуговування дуже широкого кола додатків.

За своєю суттю блокчейн є розподіленим цифровим реєстром, який зберігає дані будь-якого роду. В блокчейні може записуватися інформація про транзакції з криптовалютами, володіння NFT або розумні контракти DeFi.

На перший погляд будь-яка звичайна база даних може зберігати інформацію подібного роду, проте блокчейн має деякі переваги. Він унікальний тим, що він повністю децентралізований. Замість того, щоб тримати базу в одному централізованому сховищі - наприклад, електронній таблиці Excel, - багато ідентичних копій бази даних блокчейну зберігаються на кількох комп'ютерах, розкиданих по мережі. Такі окремі машини називаються вузлами мережі.

Термін блокчейн (blockchain) не є випадковим: цифровий реєстр часто називають “ланцюжок", тобто “chain”, який в свою чергу складається з окремих частин - “блоків”, тобто “block”. З додаванням нової інформації до мережі створюється нові блоки, які додаються до останнього вже існуючого блоку ланцюжка. І тепер вже новий блок має статус останнього. Це передбачає, що всі вузли оновлюють свою версію блокчейну, щоб він був у всіх ідентичним.

Саме те як створюються нові блоки, є ключем до того, чому блокчейн вважається даже надійним та безпечним. Більшість вузлів повинні перевірити та підтвердити валідність нових даних, перш ніж новий блок можна буде подати в ланцюжок. Для криптовалют важливо передбачити щоб нові транзакції в блоці не були шахрайськими або щоб монети не були витрачені більше одного разу. Це відрізняється від окремої бази даних або електронної таблиці, в яких один користувач може вносити зміни без нагляду.

Транзакції зазвичай захищаються за допомогою криптографічних алгоритмів. Це означає, що вузли повинні вирішувати складні математичні задачі для обробки транзакції, правильність яких повинно бути перевірено набагато простіше. Ті хто вирішують ці задачі, називаються майнерами. В нагороду за свою роботу вони отримують, як правило, криптовалюту.

Блокчейни поділяють за приватністю:

- публічні: будь-хто може брати участь. Тобто будь-який користувач може зчитувати, записувати та перевіряти на валідність дані в блокчейні. В публічному блокчейні дуже важко змінити транзакції, оскільки вузли не контролюються якимось одним головним вузлом
- приватні: контролюються групою або організацією. Саме той, хто контролює блокчейн, може вирішувати кого запрошувати до системи а також має право повертатися назад та змінювати блокчейн. Такий приватний процес більше подібний до внутрішньої системи зберігання даних

Переваги, які надає блокчейн:

- висока точність транзакцій: оскільки транзакція блокчейну повинна бути перевірена кількома вузлами, це може зменшити кількість помилок. Якщо один вузол має помилку в базі даних, інші це не приймуть оскільки більшість має іншу інформацію. А ось в традиційних централізованих сховищах якщо хтось робить помилку, висока ймовірність того, що вона не буде помічена відразу. Крім того, кожен актив окремо ідентифікується та відстежується в реєстрі блокчейну, тому немає шансів подвійного його витрачання
- відсутність посередників: маючи справу з блокчейном дві сторони в транзакції можуть підтвердити та завершити її без звертання до третьої сторони з метою врегулювати весь процес. Це економить час, а також витрати на оплату посереднику, як це буває з банком
- додаткова безпека: децентралізована мережа, як і блокчейн, теоретично, робить практично неможливим для когось здійснити

шахрайську транзакцію. Щоб це вдалося зловмиснику треба було б зламати більшу частину вузлів мережі та змінити їх ланцюжки блоків. Хоча це і не обов’язково, багато систем основаних на блокчейні використовують методи підтвердження участі та доказу роботи, що ускладнює життя шахраям

- більш ефективні перекази коштів: оскільки блокчейни працюють цілодобово, без вихідних, користувачі можуть здійснювати перекази активів та фінансів в будь-який час, особливо це важливо на міжнародному рівні. Більше не потрібно чекати днями, поки банк чи державна установа вручну все перевірять та підтвердять

Можна багато обговорювати користь від використання блокчейну та все ж слід сказати і про слабкі місця:

- обмежена кількість транзакцій в секунду: враховуючи що блокчейн залежить від великої мережі, для схвалення транзакцій існує обмеження на те, як швидко вони можуть оброблятися. Наприклад, в мережі біткоїну може оброблятися лише 4,6 транзакцій в секунду, в той час коли для Visa це значення сягає 1700
- високі витрати на електроенергію: щоб усі вузли працювали для перевірки транзакцій, потрібно значно більше електроенергії, ніж того потребує одна електронна таблиця або база даних. Це не тільки робить транзакції на основі блокчейну дорожчими, але й спричиняє велике навантаження на навколишнє середовище. В якийсь момент людство це зрозуміло і деякі лідери галузі почали відмовлятися від певних шкідливих технологій блокчейну. Наприклад, відомий Ілон Маск, турбуючись про навколишнє середовище, прийняв рішення, що Tesla частково припинить приймати біткоїн
- ризик втрати активів: щоб захистити цифрові активи використовують криптографічні ключі, які треба тримати в секреті. Доступ до ключа не можна надавати нікому другому. Навіть більше того, якщо втратити

ключ, немає способів його відновити, тож доступ до активів, як він захищав, втрачений назавжди

- можливість незаконної діяльності: децентралізація блокчейну надає приватності та анонімності, що, на жаль, робить його привабливим для злочинців. В блокчейні відстежувати незаконні транзакції набагато важче, ніж через банківські, які прив'язані до імен

3.2 Біткоїн

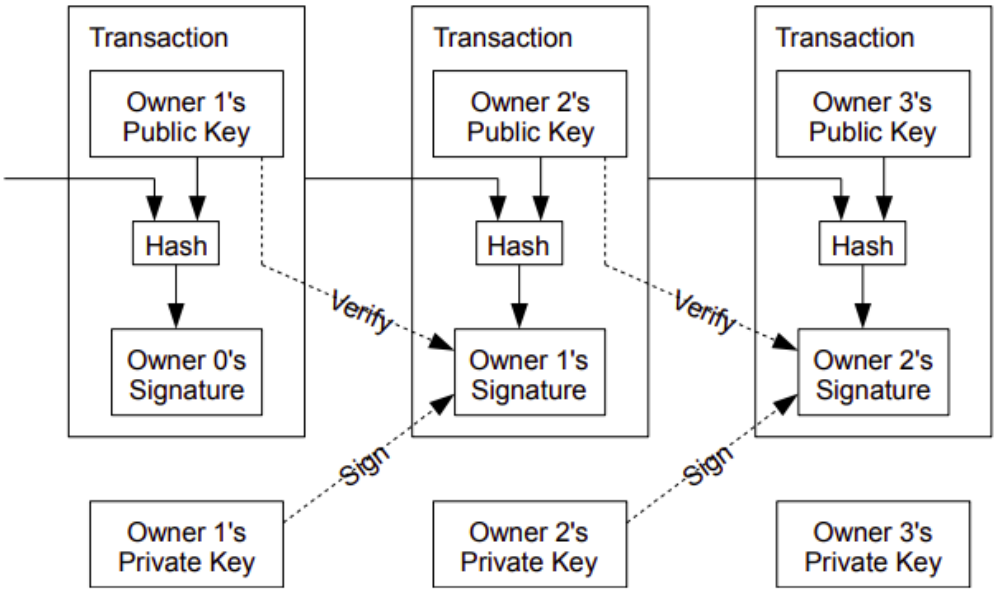
Біткоїн мережа [14] є одноранговою платіжною мережею. Її особливістю, як і мереж інших криптовалют, є те, що вона працює з криптографічними протоколами. Процес взаємодії користувачів полягає в тому, що вони надсилають транзакції разом з їх цифровими підписами, які досить складно зламати за нормальний час, шляхом трансляції повідомлень до мережі, використовуючи для цього програмне забезпечення крипто гаманців. Всі транзакції записуються в блокчейні на різних вузлах системи. Сам проект біткоїну був представлений в світ в 2009 році у вигляді програмного забезпечення з відкритим кодом.

Для обміну транзакціями мережа не потребує складних структур. Вузли за власним бажанням можуть залишати систему та під’єднуватися до мережі знову. Після оновлення роботи вузол підвантажує нові блоки для того, щоб отримати локально найактуальнішу версію блокчейну.

Сам по собі, біткоїн задається послідовністю транзакцій з цифровими підписами. Кожен користувач мережі може передавати біткоїни шляхом цифрового підпису (див. рис. 3.1) за допомогою транзакції, подібно до того, як в банку виписують чеки для переказу. Одержувач в свою чергу може перевірити на коректність кожну попередню транзакцію. На відміну від традиційних підтверджень чеків в банку, транзакції з біткоїнами не можна відмінити. Це гарантовано усуває ризик шахрайства зі зворотними платежами.

Проте існує проблема, яка полягає в тому, що одержувач не має змоги перевірити, чи хтось з користувачів мережі не намагався витрачав одні й ті ж самі монети двічі. Поширеним рішенням даної проблеми є створення надійного центрального органу, або монетного двору, який перевіряв би кожну транзакцію

на подвійні витрати. Після кожної транзакції монету необхідно повертати на монетний двір випустити нову монету, і лише монети, випущені безпосередньо з монетного двору, не підлягають подвійному витрачанню. Проблема цього рішення полягає в тому, що доля всієї грошової системи залежить від компанія, яка керує монетним двором, і кожна транзакція повинна проходити через них, як банк. Нам потрібен спосіб, щоб одержувач знав, що попередні власники не підписували раніше транзакцій. Для наших цілей важлива сама рання транзакція, тому нам байдуже про подальші спроби подвійних витрат. Єдиний спосіб підтвердити відсутність транзакції – це бути ознайомлений з усією історією транзакцій. Монетний двір був обізнаний про всі транзакції та вирішував, яка з них прийшла першою. Щоб здійснити це без втручання третьої сторони, транзакції повинні бути публічно оголошені і нам потрібна система, щоб учасники домовилися про єдину історію порядку, в якому вони були отримані. Одержувачу потрібен доказ того, що під час кожної транзакції більшість вузлів погодились, що деяка фіксована транзакція була першою.



“Рисунок 3.1 - схема підписання транзакцій в мережі біткоїну”

Проблему можна вирішити використанням міток часу. Кожен блок повинен містити мітку часу, тоді в наступні блоки будуть потрапляти транзакції, ініційовані в момент часу не менше мітки часу з попереднього блоку. Кожен наступний блок посилює коректність попередніх міток часу, як і іншої інформації блокчейну.

Щодо переказу не цілої кількості монет: можна було б обробляти монети окремо, тоді було б громіздко здійснювати окрему транзакцію для кожного цента при переказі. Щоб значення можна було розділити та об’єднати, транзакції містять кілька входів і виходів. Зазвичай буде або один вхід від більшої попередньої транзакції, або кілька введених даних, що поєднують менші суми, і щонайбільше два виходи: один для платежу, а другий повертає решту, якщо така є, назад відправнику.

Коли майнер вирішує певну задачу для додавання блоку в кінець ланцюжка, кожен може перевірити це на коректність. А задачі, які майнери намагаються вирішити називають проблемою консенсусу. Існують 2 найбільш відомих алгоритмів консенсусу [15]:

- Proof of Work (PoW). В біткоїні PoW був використаний як засіб досягнення консенсусу. При цьом Сатоші Накамото взяв ідею проекта Hashcash та додав до неї механізм змінної складності - збільшення або зменшення необхідної кількості нулів в залежності від загальної потужності учасників мережі. Pow забезпечує можливість вузлам перевірити, що майнер все ж таки виконав вірні обчислення. Даний процес включає в себе спробу знайти хеш блоку, який буде за своїм значенням відповідати рівню складності
- Proof of Stake (PoS). PoS є альтернативним механізмом консенсусу, вперше реалізований в 2012 році в криптовалюті PPCoin. Ідея полягає в використанні частки в якості ресурсу, який визначає, який саме вузол отримає право бути здобувачем наступного блоку. В даному підході вузли намагаються хешувати дані в пошуку результату менше фіксованого значення, але складність в цьому варіанті розподіляється пропорційно до балансів користувачів. Тож, більше шансів видобути блок у того, хто має більший баланс

Біткоїн використовує SHA-256 в якості алгоритма майтингу Proof of Work з моменту його запуску в 2009 році [16]. На початку алгоритм майнився тільки з використанням ресурсів CPU та GPU. Проте з часом індустрія розвивалися та

вдосконалювалися інструменти для майнинга. Як результат CPU і GPU стали менш конкурентоспроможними для майнінгу біткоїну. На сьогоднішній день в мережі біткоїну переважають тільки ASIC (спеціалізовані інтегральні схеми).

Існує велика кількість ASIC, які використовують алгоритм SHA-256, проте більшість з них є застарілими та, як наслідок, не здатні приносити прибуток. Нові покоління ASIC здатні надавати більш високу хеш-потужність.

На даний момент існує не багато відомих майнерів в мережі біткоїну, які приносять прибуток:

- Bitman Antiminer S19 Pro
- S19
- S17 +
- S17 Pro
- Bitman T19
- MicroBT Whatsminer M30S ++
- Canaan AvalonMiner 1246
- Innosilicon T3 + 52T

Приблизний огляд процесу добування біткоїнів включає наступні етапи:

- Нові транзакції транслюються на всі вузли
- Кожен вузол майнера збирає нові транзакції в блок
- Кожен вузол майнера працює над пошуком коду підтвердження роботи для свого блоку
- Коли вузол знаходить підтвердження роботи, він передає блок усім вузлам
- Вузли-отримувачі перевіряють транзакції, які вони утримують, і приймають, лише якщо всі дійсні
- Вузли висловлюють своє прийняття, переходячи до роботи над наступним блоком, включаючи хеш прийнятого блоку

Традиційна банківська модель забезпечує певний рівень конфіденційності, обмежуючи доступ до інформації залученими сторонами та довіреною третьою стороною. Необхідність публічно оголошувати всі транзакції цей метод виключає,

але конфіденційність все ще можна підтримувати, перериваючи потік інформації в іншому місці: зберігаючи анонімні публічні ключі. Громадськість може бачити, що хтось надсилає суму комусь іншому, але без інформації, яка пов'язує транзакцію з кимось. Це схоже на рівень інформації, яку оприлюднюють фондові біржі, де час і розмір окремих торгів, «стрічка», оприлюднюються, але без уточнення, хто були сторонами. Як додатковий брандмауер для кожної транзакції слід використовувати нову пару ключів, щоб вони не були пов'язані зі загальним власником. Деякі зв'язки все ще неминучі з транзакціями з кількома вводами, які обов'язково показують, що їхні вхідні дані належали одному власнику. Ризик полягає в тому, що якщо буде виявлено власника ключа, за блокчейном може виявити інші транзакції, які належали тому самому власнику.

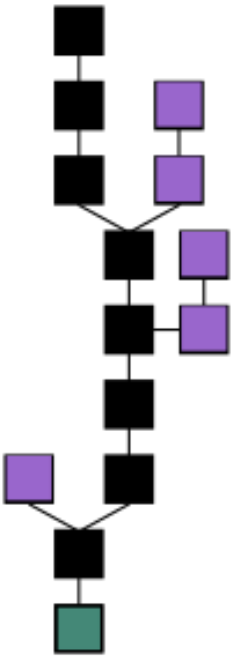
Біткойн не має підтримки державних органів, а також не має системи банків-посередників для поширення його використання. Децентралізована мережа, що складається з незалежних вузлів (див. рис. 3.2), відповідає за схвалення транзакцій на основі консенсусу в мережі біткоїн. Немає повноважень у формі уряду чи іншого грошово-кредитного органу, який би виступав у ролі контрагента для ризику та, так би мовити, повноцінних кредиторів, якщо транзакція піде не так.

Однак криптовалюта має деякі атрибути системи фіатної валюти. Його мало, і його неможливо підробити. Єдиний спосіб створити фальшивий біткоїн – це зробити подвійні витрати.

Однією з найбільших проблем є статус біткоїна як джерела цінності. Корисність біткоїна як засобу збереження вартості залежить від того, наскільки добре він працює як засіб обміну. Якби біткоїн не досяг успіху як засіб обміну, він не був би корисним як засіб збереження вартості.

Протягом більшої частини його історії спекулятивний інтерес був основним двигуном вартості біткоїна. Біткоїн продемонстрував характеристики бульбашки з різким зростанням цін і шаленою увагою засобів масової інформації (див. рис. 3.3). Ймовірно, цей показник зменшиться, оскільки біткоїн продовжує отримувати все більше поширення, але майбутнє невизначене.

Труднощі, пов’язані зі зберіганням та обміном криптовалют, також ставлять під сумнів корисність і можливість передачі біткоїн. В останні роки злом, крадіжки та шахрайство переслідували цифрову валюту.



“Рисунок 3.2 - чорний ланцюг є головним, а сині локальними невірними блоками”

Коли остання транзація опинилася всередині відносно старого блоку, усі попередні транзакції можуть бути видаленими з метою звільнення дискового простору. Щоб хеш блоку залишався незмінним, усі транзакції зберігаються у вигляді дерева Меркля, і лише хеш корінь зберігається в блоці. Розмір попередніх блоків може бути зменшений шляхом видалення непотрібних гілок дерева.

Заголовок пустого блоку займає близько 80 байт. Якщо допустити, що блоки генеруються кожні 10 хвилин отримуємо що в рік треба буде витрачати 80*6*24*365=4.2 Мб. Враховуючи закон Мура, передбачаючий збільшення витрат пам’яті на 1.2 Гб щороку, для середньостатистичного комп’ютера станом на 2008 рік з 2 Гб оперативної пам’яті зберігання даних не буде проблемою, навіть якщо

всі заголовки блоків будуть знаходитися в пам’яті.

“Рисунок 3.3 - графік вартості біткоїну”

4 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАСТОСУНКУ

Реалізувати програмний застосунок було прийнято в формі API, взаємодія з яким відбувалася б шляхом HTTP запитів. Зрозуміло що для високоякісної системи було б краще використовувати зв'язок на основі сокетів, проте варіант з HTTP запитами швидше в реалізації та є більш наглядним для створення прототипу кінцевого продукту.

Отже API було створено засобами мови Python, а саме за допомогою мікрофрейворку Flask. Flask у порівнянні з іншими фреймворками є більш легким інструментом для створення веб-додатків. Розпочати роботу з даним засобом можна просто та швидко, зберігаючи можливість масштабувати програму до складний комплексних проектів. До того ж, Flask є одним з найпопулярніших веб-фреймворків мови Python.

Весь проект складається з наступних файлів:

- block.py - опис класу Block
- blockchain.py - опис класу Blockchain
- exceptions.py - опис користувацьких класів помилок
- main.py - файл з точкою входу в програму та описом усіх маршрутів API
- merkle_tree.py - опис єдиної функції find_merkle_hash
- transaction.py - опис функцій та домоміжних класів для роботи з транзаціями

Спочатку розглянемо всі допоміжні елементи, а потім як відбувається керування ними шляхом HTTP запитів.

Почнемо з файлу block.py. Файл містить тільки клас Block, назва якого каже сама за себе. Об'єкт цього класу зберігає всю інформацію звичну для блоків блокчейну:

- transactions - список всіх транзакцій які повинні бути валідними при додаванні їх до блокчейну
- time_stamp - мітка часу, коли був ініційований процес майнингу блоку
- prev_hash - хеш попереднього блоку ланцюжка

- nonce - значення, яке майнер повинен віднайти
- miner - ім’я користувача, який знайшов потрібне значення nonce
- merkle_hash - хеш, який був отриманий після побудови дерева Меркля для списку транзакцій transactions
- hash - хеш поточного блоку

Окрім конструктора клас Block має наступні методи:

- mine_block - метод має єдиний аргумент difficulty, який позначає скільки нулів повинно бути на початку хешу блоку. В методі відбувається пошук значення nonce, допоки умова майнингу не буде виконана
- is_mined_block - метод приймає аналогічний аргумент difficulty, та перевіряє чи задовольняє блок з поточним значенням nonce умові
- calculate_hash - метод повертає хеш блоку, обчислений за допомогою SHA-256
- to_dict - перетворює об’єкт блоку до словника з корисними полями об’єкта
- from_dict - приймає словник data, який містить поля потрібного блоку та повертає саме об’єкт класу Block

Файл merkle_tree.py містить в собі тільки функцію find_merkle_hash. Вона на вхід приймає список data, який повинен складатися з транзакцій та булевого значення leaves, яке позначає чи є транзакції data листками дерева чи ні. Функція є рекурсивною і щоразу отримуючи список data, будує список pair_hashes, який є вдвічі коротшим і викликає find_merkle_hash для цього ж списку та хибного значення для флагу leaves.

Файл transaction.py містить наступні сутності:

- клас TransactionType, який успадкований від Enum з модуля enum. Містить поля MiningGenesisBlock (означає транзакцію нагороди за майнинг генезис-блоку), CoinsTransaction (означає транзакцію переказу коштів між користувачами), CreatingPhrase (позначає

створення фрази користувачем), PhraseTransaction (означає транзакцію купівлі фрази)

- клас Rewards, який зберігає у собі деякі константи, такі як MINING_REWARD = 5 (нагорода за майнинг блоків, окрім генезис-блоку), MINING_GENESIS_BLOCK_REWARD = 1000 (нагорода за майнинг генезис-блоку)
- get_transaction - метод, який приймає об’єкт класу TransactionType, а також словник з іменованими параметрами kwargs. На основі того якого типу транзакція потрібна, будується відповідний об’єкт, який являє собою словник

Файл exceptions.py призначений для збереження користувацьких помилок, які можуть виникнути в системі. Наразі присутній тільки клас ZeroLengthValidTransactionsException, який викликається при спробі змайнити блок з порожнім списком коректних транзакцій.

Файл blockchain.py містить опис найбільшого класу проекту - Blockchain. Він зберігає в собі наступні поля:

- user - ім’я користувача який створив блокчейн
- difficulty - число, що позначає скільки нулів повинен містити хеш блоків на початку
- current_transactions - список всіх транзакцій, які надходили після останнього майнингу блоку
- chain - список з блоків, який за свою суттю і є блокчейном
- other_nodes - множина сусідніх вузлів (їх адрес), які будуть використовуватися для пошуку найдовшого ланцюжка блокчейну

Список методів класу Blockchain:

- generate_genesis_block - майнить та повертає генезис-блок
- add_transaction - приймає тип транзакції та словник іменованих аргументів data. Метод викликає функцію get_transaction з файлу transaction.py та отриманий блок додає до списку chain.

- is_chain_valid - метод є статичним і приймає ланцюжок chain та значення складності difficulty. Метод перевіряє ланцюжок на коректність, перевіряючи чи співпадають записи про хеші попередніх блоків з їх реальними значеннями та значення хеша дерева Меркля
- get_last_block - метод повертає останній блок ланцюжка
- mine_new_block - метод приймає мітку часу time_stamp. В тілі методу відбувається майнинг нового блоку для поточного списку транзакцій, додавання блоку до ланцюжка chain. Метод повертає змайнений блок та список невалідних транзакцій, які були виявлені
- filter_transactions - метод фільтрує транзакції та повертає списки з валідними та невалідними транзакціями окремо
- add_nodes - метод приймає список nodes, з новими адресами сусідніх вузлів та оновлює ними множину other_nodes
- update - метод за допомогою HTTP запитів намагається отримати блокчейни сусідніх вузлів та визначити найдовший з них, щоб встановити основним для поточного вузла
- get_all_info - метод повертає поточний стан блокчейну у вигляді словника з ключами про інформацію про рахунки користувачів та інформацією про фрази
- apply_transaction - метод приймає транзакцію t, та два словники - phrases, wallets, які містять інформацію про фрази та стани рахунків, відповідно. Метод повертає булеве значення True якщо виконання транзакції було успішним, та False - інакше.
- to_dict - перетворює об’єкт блокчейну до словника з корисними полями цього об’єкту

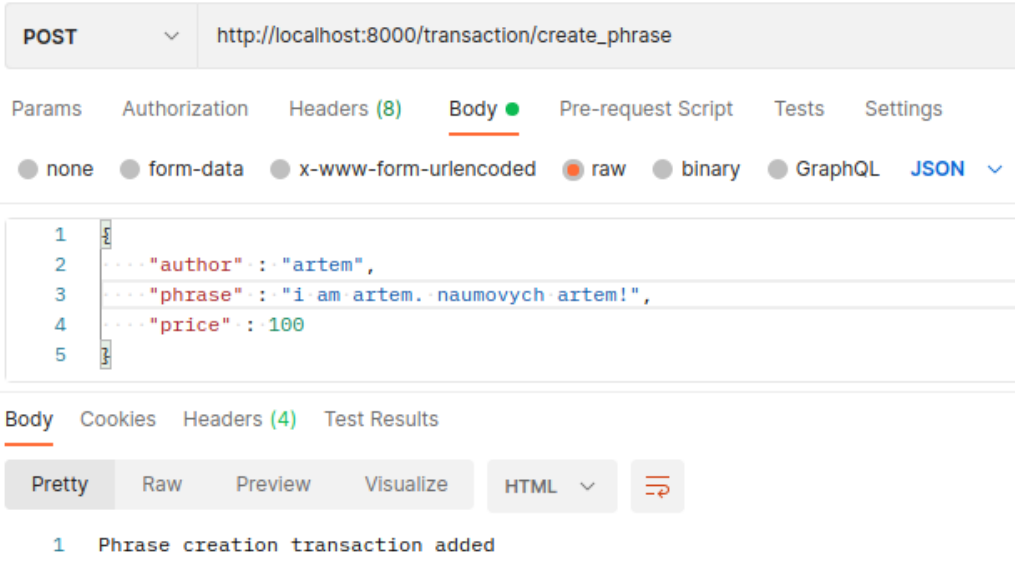
Файл main.py є точкою входу в програму. При запуску можна вказати деякі параметри: адресу вузла(хост та порт), а також ім’я користувача. Зрозуміло, що для повноцінного продукту повинна бути реалізована роботи з використанням публічного та публічного ключей та підписанням ними транзакцій та інших сутностей блокчейну. Проте в даній роботі це не використовується.

При запуску файлу створюється новий об’єкт класу Blockchain та запускається сам сервер, який підтримує наступний список маршрутів:

- blockchain - підтримує тільки GET запити та повертає список блоків поточного блокчейну
- save - підтримує тільки POST запити та записує стан блокчейну до файлу.
- transaction/send_coins - підтримує тільки POST запити. Отримує параметри для нової транзакції переказу коштів та додає її до відповідного списку current_transactions об’єкту поточного блокчейну
- transaction/create_phrase - підтримує тільки POST запити. Отримує параметри для нової транзакції створення фрази та додає її до відповідного списку current_transactions об’єкту поточного блокчейну
- transaction/buy_phrase - підтримує тільки POST запити. Отримує параметри для нової транзакції купівлі фрази та додає її до відповідного списку current_transactions об’єкту поточного блокчейну
- mine - підтримує тільки GET запити. Викликає метод mine_new_block та повертає об’єкт з ключами block та invalid_transactions, за якими міститься інформація про новий змайнений блок та некоректні транзакції, які були знайдені
- add_node - підтримує тільки POST запити. Приймає список адресів нових вузлів, які треба додати до множини сусідніх та повертає об’єкт з ключем new_other_nodes, за яким міститься оновлений список вузлів
- update_blockchain - підтримує тільки GET запити. За допомогою методу update відбувається спроба знайти довший ланцюжок, ніж мається та повертає True якщо він був знайдений, та False якщо поточний ланцюжок має найбільшу довжину
- get_balance - підтримує тільки GET запити. Повертає поточний стан про рахунки користувачів
- get_phrases - підтримує тільки GET запити. Повертає поточний стан про фрази

- is_valid_chain - підтримує тільки GET запити. Визначає валідність блокчейну та повертає довжину ланцюжка якщо він коректний, та індекс блоку, в якому порушується цілісність, інакше.

Для того, щоб запустити роботу вузла треба виконати команду: python3 main.py --port “port” --user “user”. Наприклад, python3 main.py --port 8000 --user artem та python3 main.py --port 8001 --user ilon. Можемо створити фразу від користувача artem на першому вузлі, надіславши POST запит на маршрут http://localhost:8000/transaction/create_phrase (див. рис. 4.1).



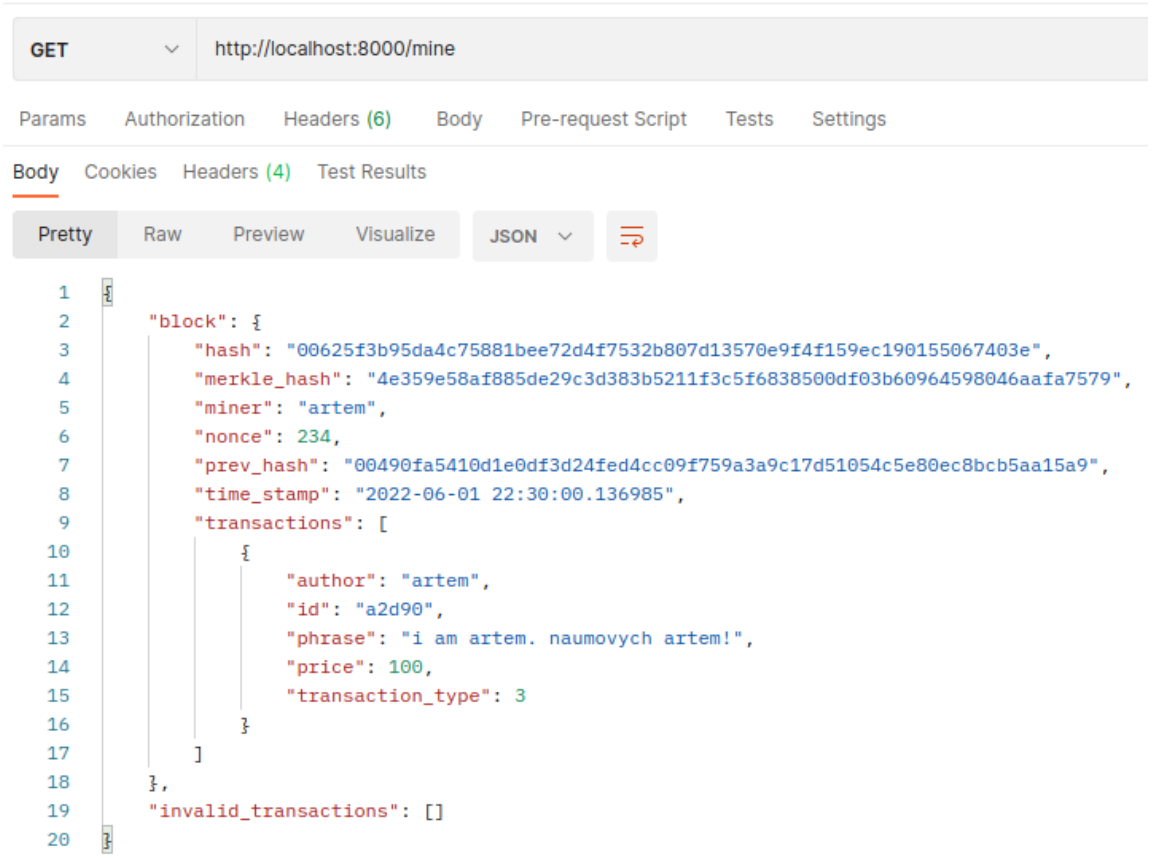
“Рисунок 4.1 - створення нової фрази користувачем artem”

Спробуємо змайнити новий блок на першому вузлі. Для цього надішлемо GET запит на маршрут http://localhost:8000/mine. У відповідь отримаємо JSON-об’єкт з новим блоком та списком некоректних транзакцій (див. рис. 4.2). Відразу можна перевірити чи починається хеш з необхідної кількості нулів.

За маршрутом http://localhost:8001/blockchain можна переконатися що ланцюжок містить тільки один блок - генезис-блок. Для того щоб отримати актуальну версію блокчейну надішлемо GET запит на маршрут http://localhost:8001/update_blockchain, попередньо додавши localhost:8000 до множини сусідніх вузлів для другого вузла.

Проробимо аналогічну процедуру створення фрази від користувача ilon. За маршрутом http://localhost:8001/get_phrases можемо дізнатися, що id цієї фрази виявилося b4d19.

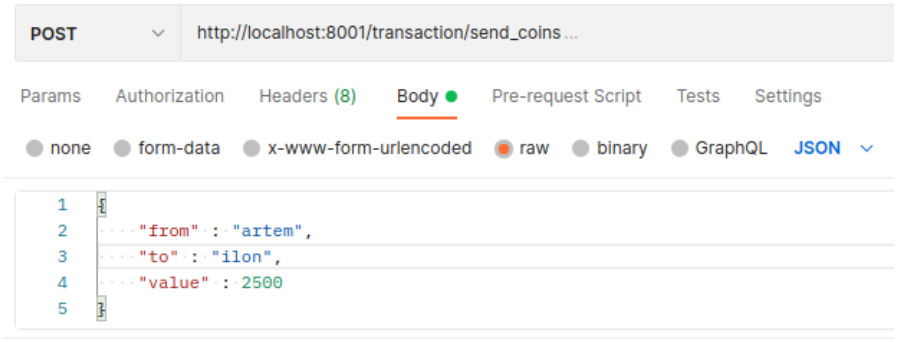
Далі за маршрутом http://localhost:8001/transaction/buy_phrase можемо провести транзакція купівлю фрази b4d19 користувачем artem(див. рис. 4.3), а після цього змайнити новий блок у другого користувача та переглянути стан усіх фраз, переконавшись що володар фрази змінився, а її автор - ні.



“Рисунок 4.2 - майнинг блоку на першому вузлі”

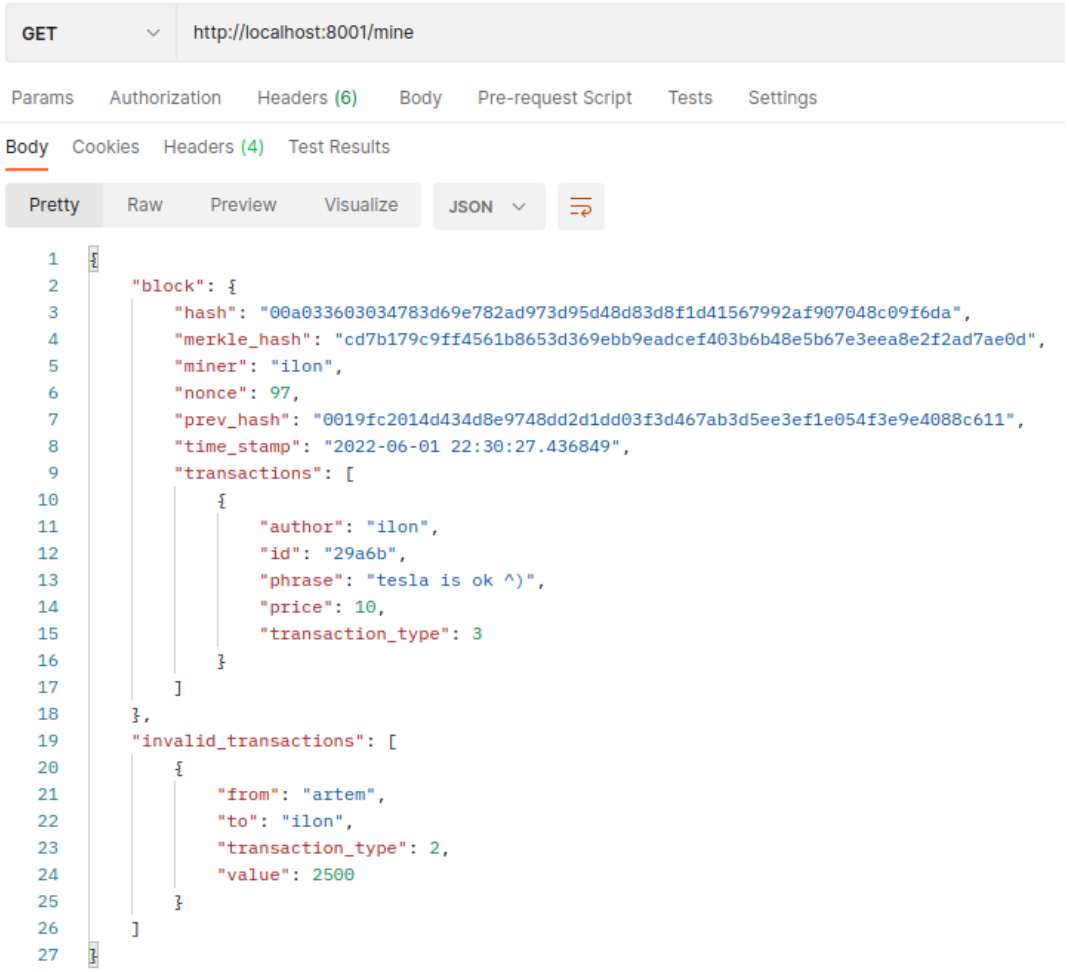
“Рисунок 4.3 - отримання інформації про фрази на другому вузлі”

Спробуємо перевірити як транзакції валідуються. Для цього створимо транзакцію переказу коштів від користувача artem до ilon, сумму вкажемо точно більше ніж баланс, наприклад, 2500 монет(див. рис. 4.4).



“Рисунок 4.4 - приклад не дійсної транзакції”

Додамо ще будь-яку кількість транзакцій та спробуємо змайнити блок, та побачимо які транзакції до нього потрапили а які позначилися як невалідні(див. рис. 4.5).



“Рисунок 4.5 - майнинг блоку користувачем ilon”

ВИСНОВКИ

У ході кваліфікаційної роботи було досліджено технологію блокчейну. Технологія не є новою, але вона знаходить своє місце у багатьох напрямках. Блокчейн може бути застосований в різних видах бізнесу. Він надає миттєву, спільну та повністю прозору інформацію. З блокчейн мережею можна відстежувати замовлення, платежі, рахунки та багато чого іншого. Найбільш відомим напрямком все ще залишаються криптовалюти, і монета біткоїн, як перший представник, але останніми роками набирають все більших оборотів NFT, тема яких була розглянута у програмній частині роботи.

Як результат було створено децентралізовану систему, яка підтримує транзакції різного вигляду:

- переказ коштів між рахунками
- створення фраз від користувачів
- купівля фраз іншими користувачами за свої кошти

В мережі реалізована робота як з взаємозамінними токенами - монетами, так і з не взаємозамінним - фразами від користувачів.

Створена система має певні недоліки в безпечності, так як відсутні приватні та публічні ключі, а отже і цифрові підписи. Це означає що на даний момент у проекті можна підробити транзакції.

В проекті великий акцент був зроблений на роботі з не взаємозамінними токенами, щоб зосередити увагу на тезі, що блокчейн - це не завжди про криптовалюти. Крім криптовалют цінність можуть представляти висловлювання відомих людей, картини, цифрові включно, та мистецтво загалом. Таким чином, блокчейн дає змогу подивитися на звичні речі під незвичним кутом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 https://www.geeksforgeeks.org/what-are-hash-functions-and-how-to-choose-a-good-hash-function

2 https://wiki.tntu.edu.ua/Хеш_функція

3 https://uk.wikipedia.org/wiki/Хеш-функція

4 https://en.wikipedia.org/wiki/Cryptographic_hash_function

5 https://uk.wikipedia.org/wiki/Лавиновий_ефект

6 https://en.wikipedia.org/wiki/MD4

7 https://en.wikipedia.org/wiki/MD5

8 https://www.freecodecamp.org/news/md5-vs-sha-1-vs-sha-2-which-is-the-most-secure-encryption-hash-and-how-to-check-them/

9 https://en.wikipedia.org/wiki/SHA-2

10 https://en.bitcoinwiki.org/wiki/SHA-256

11 https://academy.binance.com/ru/articles/merkle-trees-and-merkle-roots-explained

12 https://uk.wikipedia.org/wiki/Дерево_Меркла

13 https://www.forbes.com/advisor/investing/cryptocurrency/what-is-blockchain/

14 https://en.wikipedia.org/wiki/Bitcoin_network

15 https://forklog.com/chto-takoe-proof-of-work-i-proof-of-stake/

16 https://bytwork.com/articles/algoritm-kheshirovaniya-sha-256-kak-rabotaet-gde-ispolzuetsya-spisok-mo