Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувачка кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«14» червня 2022р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього ступеня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Програмне забезпечення для збереження цілісності даних в розподілених інформаційних системах на базі технології Blockchain»

Виконавець: студент IV курсу, групи КБ-43мс

_____ **Кирило МАЛИШЕВ**
(підпис) (ім'я прізвище)

	Прізвище, ініціали	**Підпис**
Керівник	Сергій ДАКОВ	

Нормоконтроль	Юрій ЩЕБЛАНІН	

Київ 2022

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____Наталія ЛУКОВА-ЧУЙКО
«01» листопада 2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-43мс** **Малишев Кирило Едуардович**
(група) (прізвище ім'я по-батькові)

Тема дипломної роботи Програмне забезпечення для збереження цілісності даних в розподілених інформаційних системах на базі технології Blockchain

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №5 від 29.10.2021 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Структурні моделі, алгоритми, програмне забезпечення для збереження цілісності даних в розподілених інформаційних системах на базі технології Blockchain

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Архітектури і різновиди реалізації технології блокчейн.
Проблеми цілісності даних та способи їх вирішення за допомого технології Blockchain.
Створення програмного забезпечення для збереження цілісності даних на базі технології blockchain.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Збереження цілісності інформації, використовуючи технологію блокчейн.

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 29.10.2021 року

Завдання видав _____ Сергій ДАКОВ
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ Кирило МАЛИШЕВ
(підпис) (ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	29.10.2021 – 27.01.2022	*виконано*
2	Аналіз літератури	28.01.2022 – 15.02.2022	*виконано*
3	Дослідження особливостей блокчейн	16.02.2022 – 27.02.2022	*виконано*
4	Дослідження різних алгоритмів досягення консенсусу	28.02.2022 – 20.03.2022	*виконано*
5	Аналіз використання блокчейн у різних сферах застосування	21.03.2022 – 03.04.2022	*виконано*
6	Поівняння існуючих реалізацій блокчейн для збереження даних	04.04.2022 – 24.04.2022	*виконано*
7	Проектування власної архітектури блокчейн та вибір алгоритму консенсусу	25.04.2022 – 20.05.2022	*виконано*
8	Розробка програмного забезпечення для збереження цілісності, використовуючи технологію блокчейн	21.05.2022 – 04.06.2022	*виконано*
9	Підготовка до захисту	05.06.2022 – 13.06.2022	*виконано*

Завдання видав _____ Сергій ДАКОВ
(підпис) (ініціали, прізвище)

Завдання прийняв до виконання _____ Кирило МАЛИШЕВ
(підпис) (ініціали, прізвище)

Термін подання дипломної роботи до ЕК 06 червня 2022 року

РЕФЕРАТ

Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел та додатків. Основний текст займає 66 сторінок, включає в себе зміст, вступ, три розділи дипломної роботи, висновки та список джерел. Крім того, робота містить 1 додаток із загальною кількістю сторінок 7. У пояснювальній записці дипломної роботи міститься 12 рисунків і 2 таблиці.

Метою роботи є розробка програмного забезпечення для збереження цілісності даних в розподілених інформаційних системах на базі технології Blockchain.

Для досягнення зазначеної мети поставлено наступні завдання:

- Дослідити архітектури і різновиди технології блокчейн на прикладі реально існуючих.
- Провести аналіз використання блокчейн для побудови розподіленого реєстру даних.
- Описати вимоги до розробляємого програмного забезпечення.
- Розробити програмне забезпечення для збереження цілісності інформації, використовуючи технологію блокчейн.

Об'єктом дослідження є процес збереження цілісності даних в розподілених інформаційних системах.

Предметом дослідження є механізм технології блокчейн для збереження цілісності даних.

Практичною цінністю отриманих результатів є розроблене програмне забезпечення для збереження цілісності даних в розподілених інформаційних системах.

Ключові слова: програмне забезпечення, цілісність даних, блокчейн, захист інформації.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

DLT	–	distributed ledger technology
DeFi	–	decentralized finance
NFT	–	non-fungible token
PoW	–	proof of work
PoS	–	proof of stake
TPS	–	transactions per second
CRUD	–	create, read, update, delete
IPFS	–	interplanetary file system
IPNS	–	interplanetary name system
WORM	–	write once, read many data
DCS	–	decentralized cloud storage
IPLD	–	interplanetary linked data
DHT	–	distributed hash table,

ЗМІСТ

ВСТУП

Термін «блокчейн» стає дедалі помітнішим у світі банків, інвестицій та криптовалют, а також у сфері кібербезпеки. Блокчейн захищений за допомогою різних механізмів. Він є основою структури більшості криптовалют і запобігає дублювання чи навіть знищення цифрових грошей. Однак використання технології блокчейн вивчається і в інших контекстах, оскільки вона забезпечує незмінність даних та дуже цінний захист інформації. Важливо розуміти основні концепції та механізми, які забезпечують захист цих інноваційних систем.

Дві найважливіші особливості блокчейну – це консенсус і незмінність. Консенсус – це здатність вузлів мережі блокчейн дійти згоди щодо стану мережі та дійсності транзакцій. Зазвичай процес досягнення консенсусу здійснюється за допомогою алгоритму консенсусу. Незмінність – це здатність блокчейну запобігати зміні транзакцій, які вже були підтверджені. Ці транзакції можуть включати в себе будь-які дані, заздалегіть обозначені системою.

Використання та тестування блокчейну у все нових сферах життєдіяльності людини є одним з найбільших напрямків розвитку інформаційних технологій та кібербезпеки за останні роки.

У сфері кібербезпеки блокчейн може застосовуватися у системах, які вимагають як перевірки цілісності, так і незмінності. Blockchain захищає дані від зловмисників, запобігає потенційному шахрайству та знижує ймовірність крадіжки або компрометації даних. Це стає можливим завдяки розподіленій архітектурі блокчейну. Щоб пошкодити його, необхідно атакувати кожну окрему систему (ноду), що обробляє блокчейн, а оскільки в ланцюжку можуть бути тисячі систем, ця сила в кількості означає, що атакувати дані недоцільно.

У першому розділі дипломної роботи описано загальну інформацію про блокчейн. Оглянуто архітектуру та принцип роботи. Наведено переваги та недоліки використання даної технології та приклади їх використання у таких сферах як:

- Банківська справа та фінанси.

- Харчова промисловість
- Охорона здоров'я.
- Записи про власність.
- Валюта.
- Ланцюжки поставок.
- Голосування.

У другому розділі розкрито проблематику збереження цілісності даних та способи їх вирішення з використанням технології блокчейн. Оглянуто популярні децентралізовані сервіси збереження даних в інтернеті, наведено їх переваги й недоліки.

У третьому розділі описано поетапну розробку власного програмного забезпечення для збереження цілісності даних з використання технології блокчейн. Наведено інформацію про середовище розробки, вибір мови програмування та інших інструментів розробки для виконання поставленої задачі.

РОЗДІЛ 1
АРХІТЕКТУРИ І РІЗНОВИДИ РЕАЛІЗАЦІЇ ТЕХНОЛОГІЇ БЛОКЧЕЙН

1.1 Загальна інформація про Blockchain

Технологія Blockchain є зростаючою областю, представляє інтерес для багатьох галузей у Європі та за її межами. Технологія Blockchain здатна перетворити сталі бізнес-процеси та радикально змінити роботу з регуляторами. Проте блокчейн залишається технологією експериментальної – багато проблем його використання поки що не вирішено.

Інтерес до блокчейну продовжує зростати: ще в 2016 році багато банків, бірж та фінтех-компаній оголосили про запуск власних проектів з розвитку технології. Блокчейн залишається однією з найгарячіших тем у сфері фінансових послуг та на фондових ринках, і є всі підстави очікувати зростання швидкості його розповсюдження. Відразу кілька великих фінансових організацій сформували команди на дослідження можливостей технології, і деякі учасники ринку об'єдналися у консорціуми вироблення стандартів її використання. Згідно з доповіддю, представленою на Світовому економічному форумі у 2016-му, за три останні роки у вивчення блокчейну та можливостей його застосування в індустрії фінансових послуг було вкладено понад 1,4 мільярда доларів.

Технологія дійсно здатна захистити дані, з якими доводиться працювати, зробивши їх більш доступними і прозорими. До того ж, блокчейн може помітно знизити витрати і мінімізувати час, необхідний для вирішення проблем та усунення помилок.

Блокчейн виник як технології для запуску в обіг біткойна, і спочатку використовувався виключно для управління криптовалютами. Проте з моменту його появи у 2009 році сфера застосування суттєво розширилася. Нині ж у найрізноманітніших статтях, на форумах і конференціях обговорюються нові варіанти використання технології, зокрема у торгової звітності; при безготівкових

розрахунках, перевірках та виплатах; у бухгалтерському обліку; моніторинг; управлінні ризиками; аудиті; управлінському та фінансовому обліку; комплаєнс (у тому числі запобігання фінансовим злочинам, хоча, звичайно, боротьбою з шахрайством можливості блокчейну в цій сфері не обмежені). Справа в тому, що інформація, збережена за допомогою блокчейну, може бути записана в загальному реєстрі, доступному в реальному часі або дуже близькому до нього. Отже, всі зацікавлені сторони можуть брати безпосередню участь у процесі — навіть ті, хто раніше міг розраховувати лише на стандартний звіт після завершення транзакції.

Впровадження блокчейну за визначенням складний процес, але основна ідея технології проста: розподілений реєстр або база даних, запущена одночасно на безлічі (іноді йдеться про мільйони) вузлів, розподілених по всьому світу між різними користувачами та організаціями. Унікальність блокчейну полягає у незмінності чи незворотності, яку гарантує криптографічна система захисту. Наприклад, коли транзакції з реєстру згруповані в блоки та записуються в базу даних, запис випереджає криптографічна верифікація, внаслідок чого змінити стан реєстру шляхом будь-яких махінацій практично неможливо. На користь довіри до блокчейну говорить і те, що будь-які зміни даних у ланцюжку блоків можливі тільки якщо учасники мережі підтверджують легітимність транзакції відповідно до загальних правил і протоколів.

Мета блокчейну – дозволити записувати та розповсюджувати цифрову інформацію, але не редагувати її. Таким чином, блокчейн є основою для незмінних бухгалтерських книг або записів про транзакції, які не можуть бути змінені, видалені або знищені. Саме тому блокчейн також відомий як технологія розподілених бухгалтерських книг (DLT).

Вперше запропонована як дослідницький проект у 1991 році, концепція блокчейну передувала своєму першому широко поширеному застосуванню у використанні: Біткойн, у 2009 році. З того часу використання блокчейн отримало бурхливий розвиток завдяки створенню різноманітних криптовалют, децентралізованих фінансових додатків (DeFi), невзаємозамінних токенів (NFT) та смарт-контрактів.

Блокчейн (Рисунок 1.1) дозволяє розподілити дані, що зберігаються в цій базі даних між кількома мережевими вузлами в різних місцях. Це не тільки створює надмірність, але й підтримує вірність даних, що зберігаються в ній – якщо хтось спробує змінити запис в одному примірнику бази даних, інші вузли не будуть змінені, що не дозволить зловмиснику зробити це. Якщо один користувач зіпсує запис про транзакції Bitcoin, інші вузли проведуть перехресні посилання один на одного і легко визначать вузол з невірною інформацією. Ця система допомагає встановити точний та прозорий порядок подій. Таким чином, жоден вузол в мережі не може змінити інформацію, що зберігається в ній.

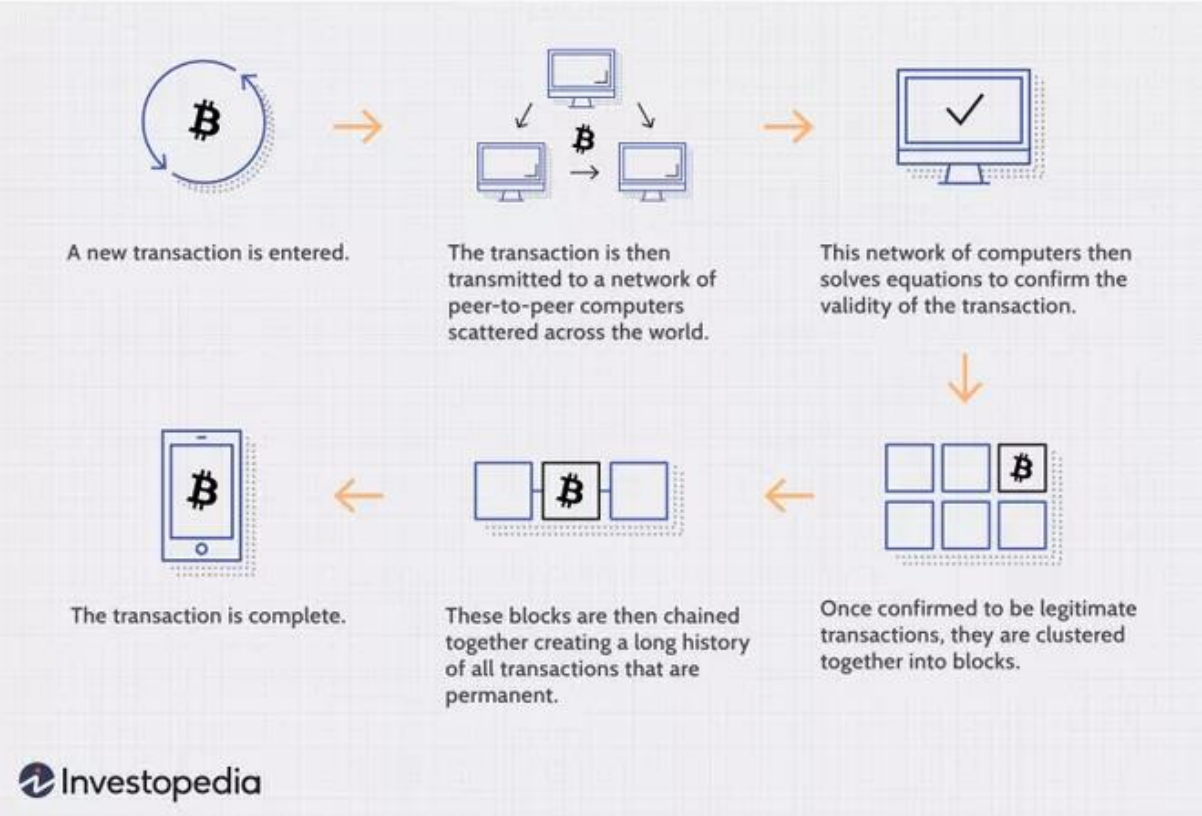

Рисунок 1.1 – Життєвий цикл транзакції в боокчейні Біткойн.

Завдяки цьому інформація та історія (наприклад, транзакцій криптовалют) необоротні. Таким записом може бути список транзакцій але в блокчейне може зберігатися і інша інформація, наприклад, юридичні контракти, державні ідентифікації або інвентаризацію продукції компанії.

Щоб підтвердити нові записи в блокчейні, потрібна згода більшості обчислювальних потужностей децентралізованої мережі. Щоб недобросовісні учасники не могли підтвердити неправомірні транзакції або подвійні витрати, блокчейн захищений механізмом консенсусу, таким як доказ роботи (PoW) або доказ частки (PoS). Ці механізми дозволяють досягти згоди навіть тоді, коли жоден вузол не є основним.

Блокчейн має наступні особливості:

Прозорість. Завдяки децентралізованій природі блокчейну Біткойна всі транзакції можна прозоро переглядати, або маючи особистий вузол, або використовуючи блокчейн-дослідники (blockchain explorer), які дозволяють будь-кому бачити транзакції, що відбуваються в реальному часі. Кожен вузол має свою власну копію ланцюжка, який оновлюється в міру підтвердження та додавання нових блоків. Це означає, що при бажанні можна відстежити біткоїн, де б він не знаходився.

Так, наприклад, у минулому біржі піддавалися хакерським атакам, у яких ті, хто зберігав криптовалюту на біржі втрачали все. Хоча зловмисник може бути повністю анонімним, здобуті ним криптовалюти легко відстежити. Якби валюта, вкрадена внаслідок злому, була кудись переведена чи витрачена, про це стало б відомо.

Записи, що зберігаються в блокчейні, зашифровані. Це означає, що тільки власник запису може розшифрувати її, щоб розкрити свою особистість (за допомогою пари відкритих та закритих ключів). В результаті, користувачі блокчейн можуть залишатися анонімними, зберігаючи при цьому прозорість.

Технологія блокчейн досягає децентралізованої безпеки та довіри декількома способами. Нові блоки завжди зберігаються лінійно та в хронологічному порядку. Тобто вони завжди додаються в "кінець" існуючого ланцюжку. Після того як блок доданий до кінця блокчейну, повернутися назад і змінити його вміст вкрай складно, практично неможливо, якщо тільки більшість учасників мережі не прийдуть до консенсусу. Це пов'язано з тим, що кожен блок містить свій власний хеш, а також хеш попереднього блоку та згадану мітку часу. Хеш-коди створюються за

допомогою математичної функції, яка перетворює цифрову інформацію на рядок цифр та літер. Якщо ця інформація якимось чином редагується, змінюється і хеш-код.

Припустимо, зловмисник, який також керує вузлом у мережі блокчейн, хоче змінити блокчейн і вкрасти криптовалюту у всіх інших. Якби вони змінили свою єдину копію, то вона вже не співпадала б з копією решти. Коли всі інші звірятимуть свої копії одна з одною, вони побачать, що ця копія виділяється, і хакерська версія ланцюжка буде відкинута як нелегітимна.

Для успіху такого злому потрібно, щоб зловмисник одночасно контролював і змінював 51% або більше копій блокчейну, щоб його нова копія стала копією більшості і таким чином узгодженим ланцюжком. Така атака також вимагатиме величезної кількості грошей і ресурсів, так як доведеться переробляти всі блоки, оскільки тепер вони матимуть різні тимчасові мітки та хеш-коди.

Враховуючи розмір багатьох криптовалютних мереж та швидкість їх зростання, вартість такої атаки, ймовірно, виявиться непереборною. Це було б не тільки надзвичайно дорого, а й безрезультатно. Така дія не залишилася б непоміченою, оскільки члени мережі побачили б такі радикальні зміни в блокчейні. Потім члени мережі зробили б хард-форк (повернулися б до стану блокчейну до додавання хакерського блоку і підтримували нову версію ланцюга, який не був зачеплений). Це призвело б до різкого падіння вартості атакованої версії токена, що зробило б атаку зрештою безглуздою, оскільки зловмисник отримав контроль над марним активом. Мережа побудована таким чином, що участь у ній є набагато економічно вигіднішим, ніж атака на неї.

Блоки на блокчейні Біткойна зберігають дані про фінансові операції. Сьогодні на блокчейні працює понад 10 000 інших криптовалютних систем та з’являються нові напрямки його застосування. Так виявилося, що блокчейн – це надійний спосіб зберігання даних та інших видів транзакцій.

Серед компаній, що вже впровадили блокчейн є такі провідін компанії в IT сегменті як: Walmart, Pfizer, AIG, Siemens, Unilever та багато інших. Так IBM

створила блокчейн Food Trust для відстеження шляху, який проходять продукти харчування, щоб дістатися місця призначення.

Харчова промисловість стала свідком незліченних спалахів кишкової палички, сальмонели та листерії, а також випадкового потрапляння небезпечних матеріалів до продуктів харчування. У минулому, щоб знайти джерело цих спалахів або причину хвороби від того, що їдять люди, були потрібні тижні. Використання блокчейну дає брендам можливість відслідковувати шлях продукту харчування від його походження через кожну зупинку, яку він робить, і, нарешті, його доставку. Якщо продукт виявиться зараженим, його можна буде відстежити на всіх етапах, починаючи з кожної зупинки і закінчуючи його походженням. Таким чином ці компанії можуть бачити все: з чим продукт міг контактувати, що дозволяє виявити проблему набагато швидше і, можливо, врятувати життя людей.

Банківська справа та фінанси. Це галузь, що найбільше отримує переваг від запровадження блокчейну у свої бізнес-операції . Фінансові установи працюють лише у робочий час, зазвичай п'ять днів на тиждень. Це означає, що якщо ви спробуєте внести чек у п'ятницю о 6 годині вечора, то, швидше за все, вам доведеться чекати до ранку понеділка, щоб побачити, як гроші надійдуть на ваш рахунок. Навіть якщо ви внесете гроші в робочий час, перевірка транзакції може зайняти від одного до трьох днів через величезний обсяг операцій, які необхідно обробити банкам. Блокчейн, з іншого боку, ніколи не спить.

Інтегрувавши блокчейн у банки, споживачі можуть побачити, що їхні транзакції обробляються всього за пару хвилин – в основному за час, необхідний для додавання блоку до блокчейну, незалежно від свят, часу доби чи тижня. За допомогою блокчейну банки також отримують можливість швидшого та безпечнішого обміну коштами між установами. У біржовому бізнесі процес розрахунку та клірингу може тривати до трьох днів (або довше, якщо торгівля ведеться на міжнародному рівні), що означає, що гроші та акції заморожуються на цей період часу.

Враховуючи розмір задіяних сум, навіть кілька днів, протягом яких гроші перебувають у дорозі, можуть нести значні витрати та ризики для банків.

Валюта. Блокчейн є основою таких криптовалют, як Bitcoin. Американський долар контролюється Федеральною резервною системою. За такої системи центрального управління дані користувача та його валюта технічно залежать від банку чи уряду. Якщо банк користувача зламаний, приватна інформація клієнта під загрозою. Якщо банк клієнта валиться чи клієнт живе у країні з нестабільним урядом, вартість його валюти може бути під загрозою. У 2008 році кілька збанкрутілих банків було врятовано – частково за допомогою грошей платників податків. Саме через ці побоювання Bitcoin був уперше задуманий і розроблений.

Розподіляючи свої операції по мережі комп'ютерів, блокчейн дозволяє Біткойну та іншим криптовалютам працювати без участі центрального органу влади. Це не лише знижує ризик, а й усуває багато комісій за обробку та проведення транзакцій. Це також може дати тим, хто перебуває у країнах з нестабільною валютою або фінансовою інфраструктурою, більш стабільну валюту з великою кількістю додатків та ширшою мережею осіб та установ, з якими вони можуть вести бізнес як усередині країни, так і на міжнародному рівні.

Використання криптовалютних гаманців для ощадних рахунків або як платіжний засіб особливо актуальне для тих, хто не має державної ідентифікації. Деякі країни можуть бути охоплені війною або мати уряди, де немає реальної інфраструктури для ідентифікації особистості. Громадяни таких країн можуть не мати доступу до ощадних чи брокерських рахунків – тобто, безпечно зберігати свої статки.

Охорона здоров'я. Медичні установи можуть використовувати блокчейн для безпечного зберігання медичних карток своїх пацієнтів. Коли медична карта створена та підписана, вона може бути записана в блокчейн, що дає пацієнтам доказ та впевненість у тому, що запис не може бути змінено. Ці особисті медичні записи можна закодувати та зберігати в блокчейні із закритим ключем, щоб вони були доступні лише певним особам, забезпечуючи тим самим конфіденційність.

Записи про власність. Сьогодні фізичний акт має бути доставлений державному службовцю до місцевого офісу реєстрації, де він вручну вноситься до

центральної бази даних округу та публічного індексу. У разі майнової суперечки претензії на власність мають бути узгоджені з публічним індексом.

Цей процес не тільки дорогий і вимагає багато часу, але й схильний до людських помилок, а кожна неточність робить відстеження прав власності менш ефективним. Блокчейн здатний усунути необхідність сканування документів та пошуку фізичних файлів у місцевому офісі реєстрації. Якщо право власності на нерухомість зберігається та перевіряється в блокчейні, власники можуть бути впевнені в тому, що їхній договір є точним і постійно реєструється.

У зруйнованих війною країнах чи районах, де практично відсутня державна чи фінансова інфраструктура, і тим більше немає бюро реєстрації, довести право власності на нерухомість може бути практично неможливо. Якщо група людей, що мешкають у такій місцевості, зможе використовувати блокчейн, то можна буде встановити прозорі та чіткі часові лінії володіння власністю.

Ланцюжки поставок. Як і в прикладі IBM Food Trust, постачальники можуть використовувати блокчейн для запису походження матеріалів, які вони придбали. Це дозволить компаніям перевіряти справжність не лише своїх продуктів, а й таких поширених маркувань, як "Organic", "Local" та "Fair Trade".

Forbes у своїх дослідженнях стверджує, що харчова промисловість все частіше використовує блокчейн для відстеження шляху та безпеки продуктів харчування протягом усього шляху від ферми до споживача.

Голосування. Як згадувалося вище, блокчейн може бути використаний для створення сучасної системи голосування. Голосування з використанням блокчейну потенційно може усунути фальсифікацію виборів та підвищити явку виборців, що було перевірено на проміжних виборах у Західній Вірджинії у листопаді 2018 року. Протокол блокчейн також забезпечить прозорість виборчого процесу, скоротить кількість персоналу, необхідного для проведення виборів, та надасть чиновникам миттєві результати. Це усуне необхідність у перерахунку голосів або будь-які реальні побоювання, що шахрайство може загрожувати виборам.

Переваги блокчейн:

Смарт контракти. Смарт контракт – це комп'ютерний код, який може бути вбудований у блокчейн для полегшення, перевірки чи погодження контрактної угоди. Розумні контракти діють з урахуванням набору умов, із якими погоджуються користувачі. Коли ці умови виконуються, умови угоди виконуються автоматично.

Точність ланцюга блоків даних. Транзакції в мережі блокчейн затверджуються мережею тисяч комп'ютерів. Це усуває майже всю людську участь у процесі перевірки, що призводить до меншої кількості людських помилок та точного запису інформації. Навіть якщо комп'ютер у мережі припуститься обчислювальної помилки, вона буде внесена лише в одну копію блокчейну. Щоб ця помилка поширилася на решту блокчейну, вона повинна бути допущена як мінімум 51% комп'ютерів мережі, що практично неможливо для великої мережі розміром з мережу Біткойна.

Скорочення витрат. Як правило, споживачі платять банку за підтвердження транзакції, нотаріуса за підписання документа або священика за укладання шлюбу. Блокчейн усуває необхідність перевірки третьою стороною, а разом з нею і пов'язані з цим витрати. Власники бізнесу несуть невелику комісію при прийомі платежів за допомогою кредитних карток, оскільки банки та компанії, що займаються обробкою платежів, мають обробляти ці транзакції. Біткойн, з іншого боку, не має центрального органу влади та має обмежені комісії за транзакції.

Децентралізація. Блокчейн не зберігає свою інформацію у центральному місці. Натомість блокчейн копіюється і поширюється по мережі комп'ютерів. Щоразу, коли до блокчейну додається новий блок, кожен комп'ютер в мережі оновлює свій блокчейн, щоб відобразити цю зміну. Завдяки поширенню інформації по мережі, а не зберіганню її в одній центральній базі даних, блокчейн стає складніше підробити. Якщо копія блокчейну потрапить до рук зловмисника, то під загрозою опиниться лише одна копія інформації, а не вся мережа.

Ефективні транзакції. Транзакції, які проводяться через центральний орган, можуть тривати кілька днів. Наприклад, якщо ви спробуєте внести чек у п'ятницю ввечері, ви можете не побачити кошти на своєму рахунку до ранку понеділка. У той час як фінансові установи працюють у робочий час, зазвичай п'ять днів на тиждень,

блокчейн працює 24 години на добу, сім днів на тиждень та 365 днів на рік. Транзакції можуть бути завершені лише за пару хвилин і вважатися безпечними вже за кілька годин. Це особливо корисно для транскордонних угод, які зазвичай займають набагато більше часу через проблеми з часовими поясами та тим, що всі сторони повинні підтвердити обробку платежу.

Приватні транзакції. Багато мереж блокчейн працюють як публічні бази даних, тобто будь-яка людина, яка має підключення до Інтернету, може переглянути список історії транзакцій у мережі. Хоча користувачі можуть отримати доступ до детальної інформації про транзакції, вони не можуть отримати доступ до ідентифікуючої інформації про користувачів, які здійснюють ці транзакції. Мережі блокчейн, такі як біткоін, анонімні, але насправді вони лише конфіденційні.

Коли користувач здійснює публічну транзакцію, його унікальний код, який називається відкритим ключем, як згадувалося раніше, записується в блокчейн. Їхня особиста інформація не записується. Якщо людина здійснила купівлю біткойнів на біржі, що вимагає ідентифікації, то його особистість, як і раніше, пов'язана з її адресою в блокчейні, але транзакція, навіть якщо вона пов'язана з ім'ям людини, не розкриває жодної особистої інформації.

Безпечні транзакції. Як тільки транзакція зареєстрована, її справжність має бути перевірена мережею блокчейн. Тисячі комп'ютерів у блокчейні поспішають підтвердити правильність деталей покупки. Після того, як комп'ютер підтверджує транзакцію, вона додається в блокчейн. Кожен блок блокчейну містить свій унікальний хеш, а також унікальний хеш попереднього блоку. Коли інформація в блоці якимось чином редагується, хеш-код цього блоку змінюється, однак хеш-код наступного блоку не змінюється. Ця невідповідність робить вкрай складною зміну інформації в блокчейні без попередження.

Прозорість. Більшість блокчейнів є повністю відкритим програмним забезпеченням. Це означає, що будь-хто може ознайомитися з його кодом. Це дає аудиторам можливість перевіряти такі криптовалюти, як Bitcoin, щодо безпеки. Це також означає, що немає реальної влади над тим, хто контролює код Bitcoin або як він редагується. Тому будь-хто може запропонувати зміни чи модернізацію системи.

Якщо більшість користувачів мережі згодні з тим, що нова версія коду з оновленням є розумною та вартісною, то Біткойн може бути оновлений.

Банкінг для будь-кого. Однією з найбільших особливостей блокчейну – є можливість його використання будь-якою людиною, незалежно від етнічної приналежності, статі чи культурного походження. За даними Світового банку, близько 1,7 мільярда дорослих людей не мають банківських рахунків або будь-яких засобів для зберігання своїх грошей або багатства. .

Ці люди часто заробляють малу кількість грошей, які їм платять готівкою. Потім їм доводиться зберігати ці готівкові гроші в схованках у своїх будинках або інших місцях проживання, що робить їх вразливими для пограбування або непотрібного насильства. Ключі від Біткойн-гаманця можна зберігати на аркуші паперу, дешевому мобільному телефоні або навіть запам'ятати, якщо це необхідно. Для більшості людей, швидше за все, ці варіанти легше сховати, ніж невелику купу готівки під матрацом.

Блокчейн майбутнього також шукає рішення не тільки для того, щоб стати розрахунковою одиницею для зберігання статків, але і для зберігання медичних записів, прав власності та безлічі інших юридичних контрактів.

Недоліки блокчейн:

Вартість технології. Хоча блокчейн може заощадити гроші користувачів на комісії за транзакції, технологія далеко не безкоштовна. Наприклад, система PoW, яку мережа Біткойну використовує на підтвердження транзакцій, споживає величезну кількість обчислювальної енергії. У реальному світі потужність мільйонів комп'ютерів у мережі біткоїну близька до річного споживання Норвегії та України.

Незважаючи на витрати на видобуток біткоїну, користувачі продовжують збільшувати свої рахунки за електроенергію для підтвердження транзакцій у блокчейні. Це відбувається тому, що коли майнери (люди, що використовують свої пристрої для підтвердження транзакцій та створення нових блоків) додають блок у блокчейн біткоїну, вони отримують винагороду у вигляді достатньої кількості криптовалюти, щоб їхній час та енергія були виправдані.

Деякі розв'язання цих проблем уже починають з'являтися. Наприклад, так звані ферми для майнінгу біткоінов були створені для використання сонячної енергії, надлишкового газу з місць гідророзриву пласта або енергії від вітряних електростанцій.

Швидкість та неефективність даних. Біткойн є ідеальним прикладом можливої неефективності блокчейну. Система PoW біткойна вимагає близько 10 хвилин для додавання нового блоку блокчейн. При такій швидкості, за оцінками, мережа блокчейну може обробляти тільки близько семи транзакцій в секунду (TPS, transactions per second). Хоча інші криптовалюти, такі як Ethereum, працюють краще, ніж біткоін, вони все одно обмежені блокчейном. Так класична банківська система Visa, наприклад, може обробляти 65.000 TPS.

Вирішення цієї проблеми розробляються вже багато років. В даний час існують блокчейни, здатні обробляти понад 30.000 TPS.

Інша проблема полягає в тому, що кожен блок може зберігати лише певний обсяг даних. Суперечка про розмір блоку був і залишається одним із найактуальніших питань для масштабованості блокчейн у майбутньому.

Незаконна діяльність. Хоча конфіденційність у мережі блокчейн захищає користувачів від злому та зберігає приватне життя, вона також дозволяє здійснювати незаконну торгівлю та діяльність у мережі блокчейн.

Темне павутиння (dark web) дозволяє користувачам купувати та продавати незаконні товари без стеження за допомогою браузера Tor та здійснювати незаконні покупки в біткоінах або інших криптовалютах. Правила, що діють у США, вимагають від постачальників фінансових послуг отримувати інформацію про своїх клієнтів при відкритті рахунку, перевіряти особистість кожного клієнта і підтверджувати, що клієнти не фігурують у списках відомих або передбачуваних терористичних організацій. Ця система може розглядатися і як плюс, і як мінус. Вона дає будь-якій людині доступ до фінансових рахунків, але також дозволяє злочинцям легше здійснювати угоди. Багато хто стверджує, що позитивні сторони використання криптовалют, такі як банківське обслуговування не охоплених банківським обслуговуванням людей, переважують негативні, особливо коли більша

частина незаконної діяльності все ще здійснюється за допомогою готівки, що не відстежується.

Хоча Біткойн і використовувався на початку для таких цілей, його прозорість і зрілість як фінансового активу призвели до того, що незаконна діяльність перемістилася в інші криптовалюти, такі як Monero і Dash (блокчейни з акцентом на анонімності людей).

Регулювання. Хоча в міру розвитку децентралізованої мережі Біткойна стає все важче і майже неможливо покласти край чомусь подібному, уряди теоретично можуть зробити незаконним володіння криптовалютами або участь у їхній мережі.

Згодом ця проблема стала менш актуальною, оскільки такі великі компанії, як PayPal, почали дозволяти володіння та використання криптовалюту на своїй платформі.

1.2 Різновиди Blockchain

Ланцюжки блоків класифікуються за доступністю реєстру даних – саме за цим параметром ділять блокчейн на класи. Хоча сам собою поділ на види є лише умовністю, адже сама технологія залишається тією ж. Але спеціалісти у сфері блокчейну мають свої методи класифікування цієї системи:

Канадська версія. Творець блокчейну Ефіріум Віталік Бутерін влітку 2015 року опублікував статтю в блозі компанії, в якій він класифікував блокчейн на 3 види:

- Громадський блокчейн (public blockchain) – повністю відкритий, де кожен може брати участь у відповідності, де транзакції ніким не контролюються і здійснюються у вільному порядку.
- Блокчейн консорціуму (consortium blockchain) – у ньому процедура узгодження контролюється відібраними вузлами.
- Приватний блокчейн (fully private blockchain) – тут усі транзакції відстежує та контролює централізований орган.

Британська версія. Майже схожу класифікацію здійснив сер Марк Волпорт - головний науковий радник Великобританії, який у своїй доповіді по розподіленим реєстрам та потенціалу блокчейну у сфері державного управління розділив блокчейн на ці 3 види:

- Unpermissioned public ledgers – відкриті публічні реєстри.
- Permissioned public ledgers – закриті публічні реєстри.
- Permissioned private ledgers - приватні приватні реєстри.

1.2.1 Громадський блокчейн (public blockchain)

Це найпопулярнійший вид блокчейну. На ньому засновані такі відомі блокчейн платформи як Біткойн та Ефіріум, що стали швидко популярними за допомогою розвитку криптовалют. Цей вид блокчейну не має управляючого органу, який підтверджує транзакції. Так, наприклад, Біткойн – це загальний регістр. Якщо відбувається будь-яка транзакція, то про це дізнається вся мережа.

Під час тразакції відбувається громадське анонсування. Люди в мережі Біткойн приймають повідомлення про створення транзакції і розпочинають процес підтвердження. Транзакцію підтверджує не якась певна людина. Невідомо, хто це буде. Сенс у тому, що жодна людина не має переваг для підтвердження транзакцій. Відкритий блокочейн можна використовувати для справді демократичної системи. Будь-хто може створювати смарт-контракти, переміщувати гроші чи вносити нові дані. У цьому вигляді блокчейну користувачі мають певну ступінь анонімності. У відкритому блокчейні можна захистити важливу інформацію.

Таким чином, використовуючи даний вид блокчейну, можна створити, як приклад, додаток, у якому будь-хто може висловити свою думку про політичні партії. Ціллю використання блокчейну буде захист анонімності користувачів. Будь-який користувач з будь-якої точки світу може висловлювати свою думку в цьому додатку. Тут немає центрального органу, який може усунути цю думку, вона залишається у мережі назавжди.

1.2.2 Блокчейн-консорціум (consortium blockchain)

У цьому виді блокчейну транзакції підтверджують певні люди. Це може бути керуючий орган, старший співробітник, уряд, установа тощо. Користувачі можуть переглядати дані (особливо важлива інформація може бути прихована).

Даний вид блокчейну можна використати щоб внести прозорість у певні бізнес-процеси компанії.

Таким чином є публічна інформація та інформація з обмеженим доступом, при цьому зберішається основна особливість блокчейну – незмінність та збереження цілісності даних.

У цьому прикладі звичайний користувач може лише переглянути інформацію про певну подію. Він не може вносити записи. Так, певні уповноважені люди за допомогою спеціального дозволу можуть записати дані, які підлягають запису у реєстр.

Користувачам не потрібна можливість записувати дані в такий блокчейн. Як і в будь-якому іншому блокчейні, всі дані залишаються в мережі назавжди.

1.2.3 Повністю приватний блокчейн (fully private blockchain)

Цей вид блокчейну повторює властивості публічного відкритого блокчейну, але на відміну від нього дані відкриті не всім.

Даний вид використовується переважно приватним бізнесом для підтвердження факту певних подій, наприклад, фінансових транзакцій між компаніями-партнерами.

У цьому прикладі транзакції з іншими учасниками бізнес-процесу є приватною інформацією, яку не повинні бачити інші люди. Проте ці дані також записуються назавжди. Коли учасники проекту взаємодіють, їм не потрібно використовувати окремий реєстр. Усі транзакції вважаються миттєвими.

1.3 Проблеми технології блокчейн

Незважаючи на революційність і переваги технології, вона не лишена й недоліків. Транзакції криптовалют захищені, а схеми шифрування з відкритим ключем майже неможливо зламати. Основні проблеми:

Людський фактор. Це проблеми, що пов'язані з небезпечним зберіганням ключів або їх розкраданням методами соціальної інженерії.

Наявність слабких місць може бути обумовлено також самою реалізацією платформи блокчейн, яка може виявитися небезпечною через використане середовище розробки або наявність вразливостей в IT-архітектурі системи.

Зі стрімким розвитком квантових комп'ютерів, які теоретично можуть розкрити всі алгоритми шифрування з відкритими ключами, постає й питання захисту від таких атак.

Занадто молода та не перевірена часом технологія, що стає причиною недовіри з боку бізнесу. Також використання блокчейну у приватних справах унеможливлює дороговизна технології на даний момент.

Висновки за розділом 1

Кількість діючих блокчейнів зростає з кожним днем все швидше та швидше. Станом на 2022 рік існує понад 10 000 активних криптовалют, заснованих на блокчейні, і ще кілька сотень некриптовалютних блокчейнів.

Публічний блокчейн, також відомий як відкритий або невирішений блокчейн, - це блокчейн, в якому будь-хто може вільно приєднатися до мережі та створити свій вузол. Через свій відкритий характер ці блокчейни мають бути захищені криптографією та системою консенсусу, такою як доказ роботи (PoW).

З іншого боку, закритий або дозволений блокчейн вимагає, щоб кожен вузол був схвалений перед приєднанням. Оскільки вузли вважаються довіреними, рівні безпеки не повинні бути такими надійними.

При всій своїй складності потенціал блокчейну як децентралізованої форми ведення записів практично безмежний. Від більшої конфіденційності користувачів та підвищеної безпеки до нижчої плати за обробку даних та меншої кількості помилок, технологія блокчейн цілком може знайти застосування і за межами описаних вище можливостей. Але є деякі недоліки.

Плюси:

- Підвищена точність за рахунок виключення участі людини у перевірці
- Скорочення витрат за рахунок виключення перевірки третьою стороною
- Децентралізація ускладнює фальсифікацію.
- Транзакції безпечні, приватні та ефективні
- Прозора технологія
- Надає альтернативу банківському обслуговуванню та спосіб захисту особистої інформації для громадян країн з нестабільним чи слаборозвиненим урядом.

Мінуси:

- Значні технологічні витрати, пов'язані зі здобиччю біткоїну
- Низька швидкість транзакцій за секунду
- Історія використання у незаконній діяльності, наприклад, у "темній павутині".
- Регулювання варіюється в залежності від юрисдикції та залишається невизначеним
- Обмеження щодо зберігання даних

РОЗДІЛ 2
ПРОБЛЕМИ ЦІЛІСНОСТІ ДАНИХ ТА СПОСОБИ ЇХ ВИРІШЕННЯ ЗА ДОПОМОГО ТЕХНОЛОГІЇ BLOCKCHAIN

2.1 Збереження даних в інтернеті

Найпростіший спосіб зберегти інформацію в інтернеті – використовувати хмарні сервіси, наприклад Google Drive, MySQL, MongoDB. Користувач отримує доступ до сховищ компанії на безоплатній або платній основі. При цьому компанія зберігає контроль над базою даних і надає клієнтам право доступу до сховищ, а також відповідає за безпеку і збереження даних.

Даний процес виглядає наступним чином:

1. Користувач за допомогою сайту або десктопного додатку завантажує дані на сервери компанії.

2. Сервіс вносить відомості в центр обробки даних.

3. Всякий раз, коли ви хочете отримати доступ до завантаженої інформації, ваш пристрій відправляє запит в центр обробки даних і він надає доступ до інформації.

Це стандартна модель, яка домінує на ринку. Вона має дві переваги:

- Користувач може оперувати чотирма ключовими функціями: Create, Read, Update і Delete (CRUD).
- Користувач може швидко завантажувати і вивантажувати дані.

Дана класична модель має й суттєві недоліки:

- Як показує практика, такі сервіси часто піддаються зламу зламу, вони ненадійні і інформація в них використовується без відома і згоди користувача (наприклад, в маркетингу). Саме тому крадіжка персональних даних стала нормою в XXI столітті. Наприклад, 19 листопада 2018 року влада США опублікувала результати розслідування у справі про крадіжку персональних даних 500 мільйонів

клієнтів мережі готелів Marriott. Так само зловмисники регулярно отримують доступ до приватних даних знаменитостей з iCloud.

- Використання традиційних способів зберігання – шлях до монополізації ринку, оскільки чим більше сервісів, що пропонують послугу сховища, тим дешевше у ході конкуренції самі послуги. Монополізація ринку зазвичай є причиною зниження якості послуг і уповільнення темпів науково-технічного зростання.

2.2 Сховища даних на блокчейні

В основі блокчейну – послідовность блоків, кожен з яких несе в собі якийсь обсяг інформації. Даний обсяг обмежений технічною реалізацією блокчейну. Так для біткойна цей максимальний об’єм не може переважати 1 MB даних. Обмеження показує максимальний розмір файлу, який можна завантажити в блокчейн. Для зберігання інформації про транзакції 1 MB достатньо, однак, якщо потрібно зберегти зображення або відеофайл, потрібно шукати інше рішення.

Одним з варіантів рішення даної проблеми є встановлення більш високої планки максимального розміру блоку даних, або перетворення даного параметру зі статичного в динамічний, що буде змінюватися в залежності від навантаження на блокчейн. Але в цьому випадку виникає інша проблема – надмірно висока вартість зберігання інформації. Справа в тому, що обробка даних в публічних блокчейнах не безкоштовна. На це витрачається обчислювальні ресурси, які коштують реальних грошей, і чим більше розмір файлу (даних), тим більше ресурсу йде.

Так у 2017 році підрахували, що завантаження 1 KB інформації в блокчейн обходиться користувачам в 2 американських долара. Виходить, завантаження в блокчейн, наприклад, текстового файлу розміром в 600 KB обійдеться користувачам в 1 200 дол, фільму розміром в 5 GB – більше 10 млн дол.

2.3 Варіанти використання блокчейну для збереження даних

2.3.1 Повне зберігання даних в блокчейні

Проблему з обмеженням максимального розміру блоку можна вирішити декількома способами. Найпростіший з них має на увазі:

- Дроблення файлу на сегменти (осколки), розмір яких менше розміру блоку. Таким чином, навіть найбільший файл можна записати на блокчейн з невеликим розміром блоку.
- Шифрування даних в осколках, щоб тільки їх власник міг зрозуміти, що в них записано. Це дозволить зберігати інформацію у відкритому блокчейні і бути впевненим в її конфіденційності.
- Розподіл осколків по мережі блокчейну. Завдяки цьому файл буде збережений в незмінному вигляді, поки хоча б один користувач синхронізований з блокчейном.

Цей підхід запозичений у торрент-трекерів, але для зберігання даних за допомогою блокчейну він не підходить, навіть якщо прибрати комісії за створення транзакцій через наступні причини:

- Запис інформації в блокчейн здійснюється за допомогою транзакцій, вони, в свою чергу, вимагають підтвердження. Великий файл може займати кілька тисяч транзакцій, тобто декількох годин, а то і днів обробки.
- Інформація в блокчейне незмінна. Отже, ви не можете видалити або змінити непотрібні дані. Всі потрапили в мережу файли і їх варіації назавжди залишаться в блокчейні і теоретично хтось рано чи пізно зможе їх переглянути. Якщо блокчейн втратить популярність і залишиться один користувач, то останній зможе одноосібно ним управляти, змінюючи правила системи як душі завгодно.

Незмінність призведе до ще однієї проблеми – лавиноподібного зростання розміру блокчейну. Якщо інформацію не можна видалити, вона буде тільки накопичуватися, що з часом зробить розмір блокчейну занадто великим для

звичайного користувача. Наприклад, уа даний момент розмір популярних публічних блокчейнів становить:

- Біткойн – 220 GB.
- Ефіріум – 600 GB.

Це вже занадто багато для смартфонів, планшетів і значної частини ноутбуків.

Підсумовуючи, можна дійти висновку, що зберігання інформації безпосередньо в публічному блокчейні не найкраща ідея, якщо мова йде про великі дані. Цей варіант підійде лише в тих випадках, коли обсяг інформації знаходиться в межах декількох кілобайт. Наприклад, коли мова йде про фінансові транзакції, персональні дані або документообіг.

2.3.2 Однорангові файлові системи

Прикладом такої системи є мережа протоколу InterPlanetary File System (IPFS). Цей протокол має властивості блокчейн та побудований з використанням протоколу BitTorrent, основна ідея якого полягає у розбивці файлів на частини і їх зберігання в декількох примірниках на комп'ютерах учасників системи.

Коли користувач додає файл до IPFS, файл розбивається на дрібніші фрагменти, криптографічно хешується і отримують унікальний відбиток, що називається ідентифікатором вмісту (CID). Цей CID діє як постійний запис вашого файлу у тому вигляді, в якому він існує зараз.

Коли інші вузли шукають ваш файл, вони запитують своїх вузлів-аналогів, хто зберігає вміст, на який посилається CID файлу. Коли вони переглядають або завантажують ваш файл, вони кешують копію і стають ще одним постачальником вашого вмісту, поки їх кеш не буде очищений.

Вузол може закріпити вміст, щоб зберігати (і надавати) його вічно, або відкинути вміст, який він давно не використав, щоб заощадити місце. Це означає, що кожен вузол у мережі зберігає лише той вміст, який його цікавить, плюс деяку інформацію про індексування, яка допомагає визначити, який вузол зберігає.

Якщо ви додаєте в IPFS нову версію файлу, його криптографічний хеш змінюється, і тому він отримує новий CID. Це означає, що файли, що зберігаються в IPFS, стійкі до зламу та цензури – будь-які зміни у файлі не перезаписують оригінал, а загальні фрагменти файлів можна використати повторно, щоб мінімізувати витрати на зберігання.

Також IPFS (Рисунок 2.1) може знайти останню версію вашого файлу за допомогою децентралізованої системи іменування IPNS, а DNSLink можна використовувати для зіставлення CID з іменами DNS.

У даного підходу збереження файлів є кілька переваг:

- файл буде викачаний користувачами, тільки якщо він комусь цікавий;
- популярні файли завантажуються та поширюються дуже швидко;
- дані адресно залежні, тому підробити внутрішній зміст файлу неможливо;
- це однорангове рішення (peer-to-peer).

З недоліків можна відзначити, що завантаження файлу в мережу відбувається, тільки якщо користувач знаходиться онлайн і така система обслуговує виключно статичні дані. Крім того, отримати доступ до файлу можна лише знаючи ім'я/шлях до нього.

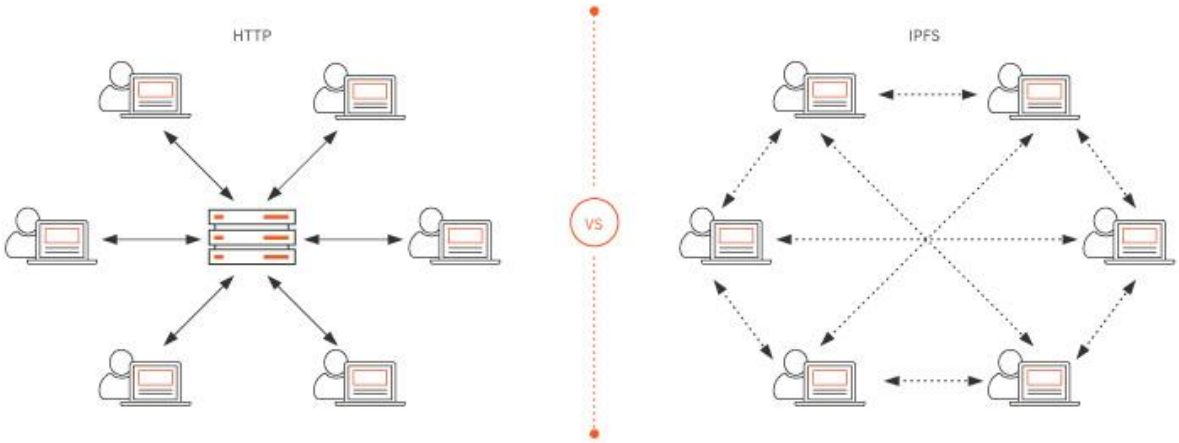

Рисунок 2.1 – Порівняння протоколів HTTP і IPFS

Блокчейн в цій схемі використовується в якості посередника, який пов'язує учасників між собою і відповідає за перевірку автентичності і цілісності файлів.

Крім того, його можна використовувати для монетизації процесу: сіди отримують гроші за роздачу файлів, бенкети платять за їх скачування.

2.3.3 Децентралізовані хмарні сховища

Це практично ті самі звичайні хмарні сховища на зразок Dropbox або Google Drive, тільки дані розміщуються не на серверах компаній, а на пристроях користувачів, які здають їх в оренду. Подібних сервісів існує багато, наприклад Storj (Рисунок 2.2).

Використовуючи подібні рішення не потрібно постійно перебувати в мережі, щоб розповсюджувати інформацію іншим учасникам мережі. Досить один раз завантажити файл в хмарне сховище. Такі сховища стабільні, швидкі і мають величезні ємності.

Однак вони підходять тільки для обслуговування статичних даних і не підтримують пошук за змістом. Крім того, вони не безкоштовні, оскільки люди орендують обладнання один у одного.

Об'єктне зберігання ідеально підходить для файлів, які записуються один раз і багато разів читаються (WORM - Write Once, Read Many data). Об'єктне зберігання даних стало основою багатьох різних додатків та варіантів використання. Розподіл і децентралізоване об'єктне сховище оптимально для великих файлів, особливо якщо до цих файлів регулярно звертаються з географічно різних місць.

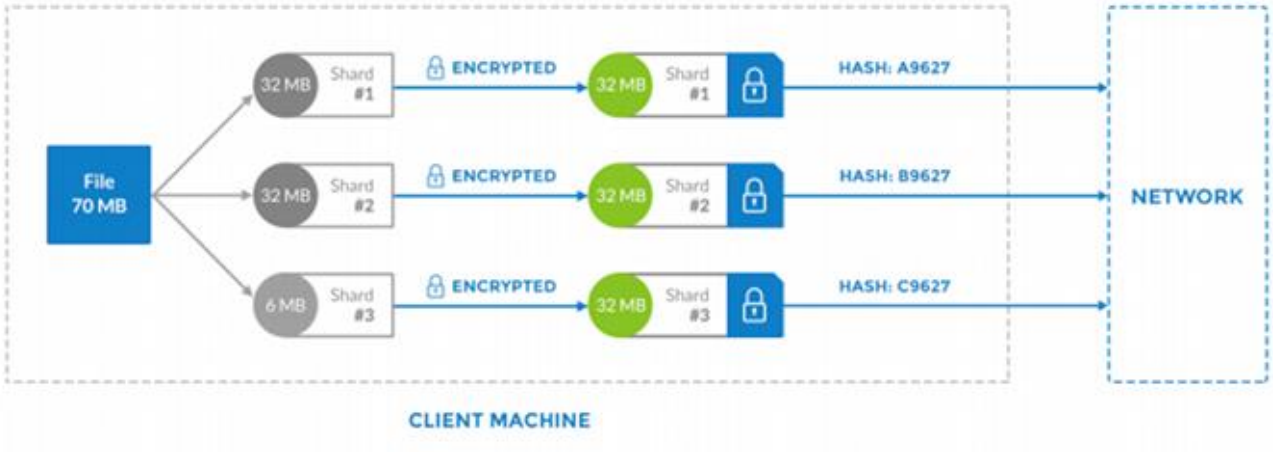

Рисунок 2.2 – Візуалізація процесу вивантаження в Storj

Об'єктне сховище загалом має широкий спектр варіантів використання. Нижче наведено випадки використання, які найкраще підходять для Storj DCS. Якщо ви не бачите у списку свого варіанта використання, не хвилюйтеся - це просто дасть вам уявлення про те, як Storj DCS має використовуватись.

2.3.4 Розподілені бази даних

Якщо потрібно зберігати великі обсяги структурованої інформації і шукати контент за запитами, слід звернути увагу на NoSQL. SQL не підходять, оскільки інформація в них не може бути розподілена на частини через обмеження теореми Брюера – у розподіленій системі можна вибрати тільки 2 з 3-х властивостей:

- C (consistency) – узгодженість. Кожне читання дасть вам останній запис.
- A (availability) – доступність. Кожен вузол (не впав) завжди успішно виконує запити (читання та запис).
- P (partition tolerance) – стійкість до розподілу. Навіть якщо між вузлами немає зв'язку, вони продовжують працювати незалежно один від одного.. Щоб зробити базу даних по-справжньому розподіленою, потрібно пожертвувати доступністю і узгодженістю.

Саме це роблять бази даних NoSQL – жертвують узгодженістю вузлів блокчейну, зменшуючи узгодженість (вузли стають узгодженими тільки через певний час) та збільшуючи доступності. На основі цього підходу реалізовано багато проектів, наприклад RethinkDB, Apache Cassandra, MongoDB та інші. Перевагами такого пдходу є:

- Відмовостійкість.
- Висока швидкість.
- Проста горизонтальна масштабованість.
- Підтримка багатої мови запитів.

Не зважаючи на велику кількість переваг у них є один істотний недолік – всі вузли повинні довіряти один одному. Це важливо, оскільки якщо серед вузлів з'явиться шкідливий елемент, він зможе самостійно знищити всю базу даних.

2.4 Приклади використання блокчейну для зберігання інформації

2.4.1 Хмарне сховище BigChainDB

Хмарне сховище з величезним обсягом і дуже швидкими транзакціями. Збудовано на кластері RethinkDB і використовує механізми NoSQL для зберігання блоків, завдяки чому володіє високою відмовостійкістю і пропускною здатністю.

BigchainDB – це масштабована база даних блокчейна. Вона розроблена, щоб об'єднати найкращі властивості з розподілених баз даних та блокчейну. Має такі властивості, як: масштабованість (пропускна здатність, ємність, низька затримка) та

можливість запитів. децентралізованість (жоден суб'єкт не володіє та не контролює мережу), незмінність (стійкість до злому) та має активи (ви володієте активом, якщо володієте закритим ключем, так звана роздільна здатність у стилі блокчейна).

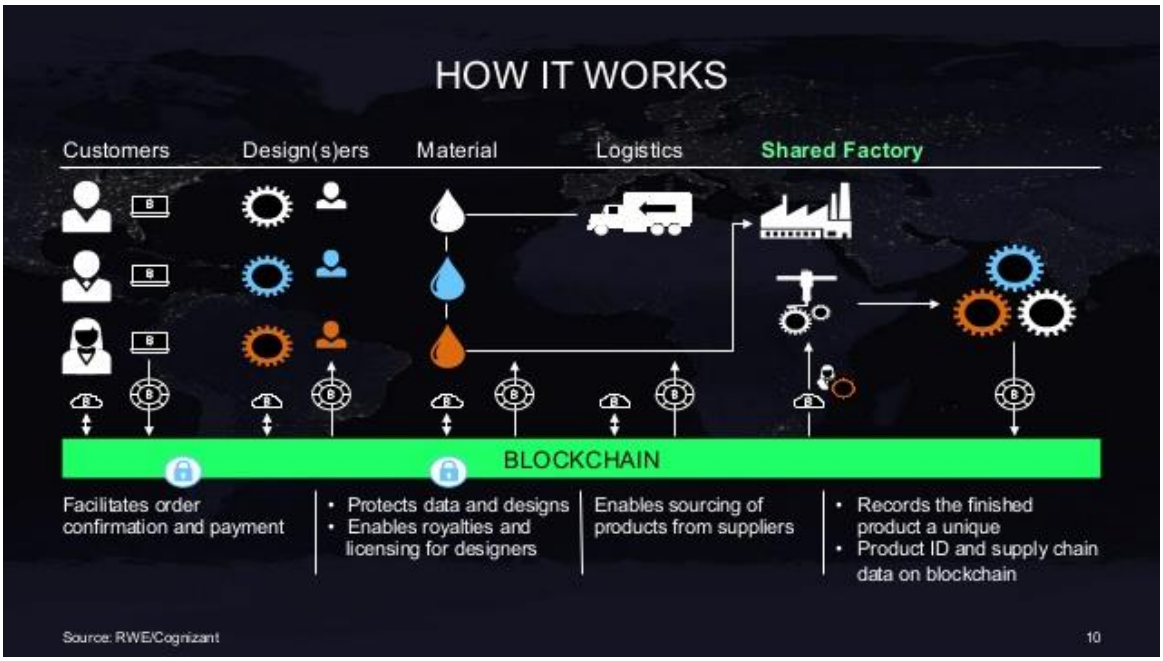

Рисунок 2.3 – Схема роботи BigChainDB

Всі учасники мережі BigChainDB підключені до єдиного кластеру і мають повні права на запис, зміну і видалення інформації, тому даний кейс не підходить

для публічних рішень. Але його можна використовувати для приватних корпоративних завдань:

- Діловодство та Юриспруденція;
- бухгалтерська звітність;
- відстеження активів;
- фінансові дані.

BigchainDB підтримує як публічне, і приватне розгортання. Записи займають менше секунди, оскільки перевірка ґрунтується на федерації голосуючих вузлів. Запити в BigchainDB поки не реалізовані, але безпосередньо використовуватимуть функціональність запитів базової бази даних. За результатами експериментів розробників, архітектура BigchainDB має теоретичну пропускну здатність в 1 мільйон записів на секунду та зберігання петабайтів даних (за допомогою технології шардінгу).

Будучи децентралізованою базою даних, BigchainDB (Рисунок 2.4) доповнює технології децентралізованої обробки даних, такі як Ethereum Virtual Machine (EVM) та децентралізовані файлові системи, такі як IPFS. Вона може бути використана в рамках децентралізованих обчислювальних платформ, таких як BlockApps-Stratos або Eris-Tendermint.

	BITCOIN	DISTRIBUTED DATABASES	BIGCHAINDB
Immutability	✓		✓
No Central Authority	✓		✓
Assets Over Network	✓		✓
High Throughput		✓	✓
Low Latency		✓	✓
High Capacity		✓	✓
Rich Permissioning		✓	✓
Query Capabilities		✓	✓

Рисунок 2.4 – Порівняння блокчейну BigChainDB з блокчейном Біткойн та класичними розподіленими базами даних

2.4.2 Блокчейн Sia

Sia – це децентралізована платформа хмарного зберігання даних, захищена технологією блокчейн. Мережа зберігання даних Sia використовує ємність жорстких дисків по всьому світу, що використовується для створення більш надійного і дешевого способу зберігання даних, ніж традиційні постачальники хмарних сховищ. У Sia є свій блокчейн та корисний токен, який забезпечує його роботу – Siacoin.

Дані користувача у блокчейні конфіденційні та зберігаються у нодах по всьому світу, щоб уникнути відмови системи та забезпечити максимально можливий час безвідмовної роботи. Доступ до власних файлів є тільки у власника ключів. Жодна стороння компанія не може отримати доступ до файлів. Дані не можуть бути видалені. Файли не можуть бути зламані.

Sia має цілу екосистему з орендарями, хостерами, майнерами та численною спільнотою розробників, які створюють на базі Sia такі проекти, як мережа другого рівня Skynet та корпоративний сервіс зберігання даних Filebase.

Коли файл завантажується на Sia, він поділяється на частини, шифрується та відправляється у всьому світі. Орендарі завантажують файли, хостери зберігають ці файли. Як тільки ви завантажуєте свої файли, мережа гарантує, що вони завжди будуть доступні вам, копіюючи їх кілька разів. І вони ніколи не будуть доступні хостерам, тому що вони одержують лише частини цілих файлів, які вже зашифровані.

Блокчейн пропонує більш дешеве, швидке і безпечне сховище. Однак це зовсім не означає, що такі послуги дешевше, ніж у Google, Amazon або DropBox. Просто сервіси, побудовані на блокчейні, отримують прибуток не тільки з орендних ставок, але і з комісій з проведення транзакцій, пов'язаних із завантаженням і витяганням даних.

Попит на подібні сховища постійно зростає, оскільки зростає ринок і людям подобається можливість використання нової технології, навіть якщо вони в ній майже нічого не розуміють.

2.4.3 Децентралізоване хмарне сховище TiesDB

TiesDB (Рисунок 2.5) пропонує першу в його історії публічну децентралізовану розподілену базу даних. Це рішення покликане задовольнити зростання попиту ринку децентралізованих додатків на зберігання нефінансових даних.

TiesDB призначена для структурованого зберігання великих масивів даних у публічній, децентралізованій, розподіленій мережі, які можуть бути надійно збережені у блокчейні. База даних є публічною, децентралізованою і має можливість пошуку даних, що є одним із найважливіших параметрів.

Її особливість – страхові депозити, які орендодавці повинні внести на смарт-контракти, щоб отримати доступ до монетизації. Без такого страхового внеску доступна лише клієнтська частина платформи.

Страховий внесок знаходиться всередині смарт-контракту, поки орендодавець не вирішить припинити свою діяльність, виконавши перед цим всі свої зобов'язання, за які він отримав гроші. Якщо ж орендодавець видалить файли інших користувачів або просто зникне, гроші вилучаться з смарт-контракту і розподіляться всередині системи.

Abilities/Data Storages	Ties.DB	IPFS	BigChainDB
Type	Database	File System	Blockchain
Distribution	●●●●	●●○○	●●●○
Publicity	●●●●	●●●●	●●●●
Resistance to the Byzantine Generals' Problem	●●●●	●●●●	●○○○
Sharding Support	●●●●	●○○○	●●○○
Speed	●●●●	●●●●	●●●○
Ability to store srtuctured data	●●●●	○○○○	●●●●
Ability to delete data	●●●●	○○○○	○○○○
Request language with an ability to conduct search using more than the primary key	●●●●	○○○○	○○○○

Рисунок 2.5 – Порівняння TiesDB, BigchainDB та IPFS

Формально TiesDB не є блокчейном, оскільки це просто децентралізоване хмарне сховище, яке використовує смарт-контракти для мотивації і покарання учасників, а також зберігання інформації про ставки, взаємні розрахунки учасників і страхові депозити.

TiesDB користується блокчейном для узгодження суперечливих відомостей про внесення змін до бази даних і пов'язаних з ними фінансових операцій. Наприклад, якщо відомості про товар зберігаються в базах даних TiesDB, інформація про кількість товарів також повинна бути внесена в базу даних. Якщо цього не зробити, покупець може заплатити за товар, якого більше немає в наявності.

2.4.4 Блокчейн Filecoin

Filecoin – це p2p мережа для зберігання файлів із вбудованими економічними стимулами для забезпечення надійного зберігання файлів протягом тривалого часу.

У Filecoin користувачі платять за зберігання файлів у провайдерів. Провайдери сховищ – це комп'ютери, які відповідають за зберігання файлів і доводять, що вони зберігали файли правильно протягом тривалого часу. Будь-хто бажає зберігати свої файли або отримувати гроші за зберігання файлів інших користувачів може приєднатися до Filecoin. Доступне сховище та ціни на нього не контролюються якоюсь однією компанією. Натомість Filecoin сприяє створенню відкритих ринків для зберігання та отримання файлів, в яких може брати участь будь-хто.

Filecoin включає блокчейн і власну криптовалюту (FIL). Постачальники послуг зберігання заробляють одиниці FIL за зберігання файлів. Блокчейн Filecoin реєструє транзакції надсилання та отримання FIL, а також докази постачальників послуг зберігання про те, що вони правильно зберігають свої файли.

Filecoin дозволяє користувачам зберігати свої файли за конкурентними цінами та перевіряти правильність їх зберігання.

Користувачі можуть вибрати бажаний компроміс між вартістю, надмірністю та швидкістю, вибираючи постачальника послуг зберігання, чия пропозиція зі зберігання найкраще відповідає їх потребам. Програми, які використовують Filecoin, можуть домовлятися про зберігання будь-якого постачальника послуг зберігання в мережі. На відміну від централізованих систем зберігання, не потрібно реалізовувати окремий API для кожного провайдера.

У будь-який момент користувачі можуть переконатися, що їх файли зберігаються правильно, подивившись докази на блокчейні Filecoin.

Filecoin та IPFS працюють на основі однієї і тієї ж технології на багатьох рівнях:

- IPLD (модель даних для взаємодіючих протоколів) визначає формати даних із прив'язкою до вмісту, наприклад, блокчейн або спосіб зберігання файлів в IPFS.
- libp2p забезпечує можливості одноранговой мережі, безпеку з'єднань та функції виявлення ключів та розподілу даних, такі як розподілені хеш-таблиці (DHT) та Pubsub.
- Мультиформати визначають перспективні ідентифікатори та типи даних.
- Graphsync та Bitswap забезпечують швидку та ефективну передачу даних IPLD між вузлами.

2.4.5 Блокчейн Maidsafe

MaidSafe – це проект децентралізованого інтернету. Концепція MaidSafe на кілька років передувала Біткойну, хоча на разі вона все ще знаходиться в бета-фазі. Коли мережа SAFE буде завершена, вона буде працювати аналогічно до мережі TOR, тобто розповсюдження та доступ до інтернет-контенту буде здійснюватися в осередковій мережі P2P.

MaidSafecoin – це тимчасова криптовалютна монета, яка використовується для альфа- та бета-версій мережі SAFE, що розшифровується як Secure Access For Everyone. MAID – це абревіатура від Massive Array of Internet Disks.

Використання децентралізованого інтернету протистоїть цензурі, оскільки сервери найважче знайти. Саме тому пірати перейшли від централізованих серверів Napster до децентралізованих протоколів Gnutella та BitTorrent.

Замість майнінгу користувачі мережі SAFE спільно використовуватимуть системні ресурси так само, як вони контролюють швидкість завантаження та вивантаження в сучасних клієнтах BitTorrent. Однак замість пропускної спроможності мережі користувачі також спільно використовуватимуть дисковий простір, пам'ять і обчислювальну потужність процесора.

Основна ідея даного рішення – це створення повністю зашифрованої P2P-мережі, яка буде базою даних для анонімного обміну інформацією через зашифровані шари, як TOR (The Onion Router), тільки для хмарних сховищ. Це стає можливим завдяки трьом елементам безпеки для покоївки:

- Self-шифрування: шифруючі самі себе, шифруючі самі себе. він розбивається на безліч невеликих осколків, які самостійно зашифровуються і поширюються по мережі, які самостійно зашифровуються і поширюються по мережі. У такій формі файл стає нечитабельним для всіх, крім власника.
- Децентралізоване кешування даних. Дані будуть зберігатися в безпечному мережу по всьому світу, а не на серверах однієї компанії або мережі компаній. Це дозволить зробити платформу автономною і підвищить рівень захищеності інформації.
- Доступність даних. Мережа постійно створює і підтримує дублікати всіх файлів, які вона зберігає. Ця функція веде до надлишкової інформації, що має захистити її від втрати внаслідок відключення окремих вузлів.

Висновки за розділом 2

При використанні блокчейну для зберігання даних важливо пам'ятати, що поточні технології не дозволяють зберігати великі обсяги інформації всередині ланцюжка блоків. Тому блокчейн в цій галузі використовується у вигляді

посередника і бухгалтерської книги, яка стежить за дотриманням умов угоди з надання сховища однією людиною іншій.

Це означає, що ні блокчейн, ні смарт-контракти, ні криптографія не захищають інформацію в децентралізованих сховищах. В даному випадку інформація має той же захист, що і в традиційних сховищах.

У другому розділі розкрито проблематику збереження цілісності даних та способи їх вирішення з використанням технології блокчейн. Оглянуто популярні децентралізовані сервіси збереження даних в інтернеті, наведено їх переваги й недоліки.

РОЗДІЛ 3
СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБЕРЕЖЕННЯ ЦІЛІСНОСТІ ДАНИХ НА БАЗІ ТЕХНОЛОГІЇ BLOCKCHAIN

3.1 Середовище розробки VS Code

Для написання програмного коду використано інтегроване середовище розробки Microsoft Visual Code (Рисунок 3.1). Це середовище обрано через його простоту й широкий перелік необхідних функціональних можливостей:

1. Робота з інтерпретованою мовою програмування Python.

2. Підтримка синтаксису, виправлення помилок та автодоповнення таких складових розробки як код мовою Python та дані у форматі Json.

3. Використання плагінів, що розширює наявний функціонал до необхідного.

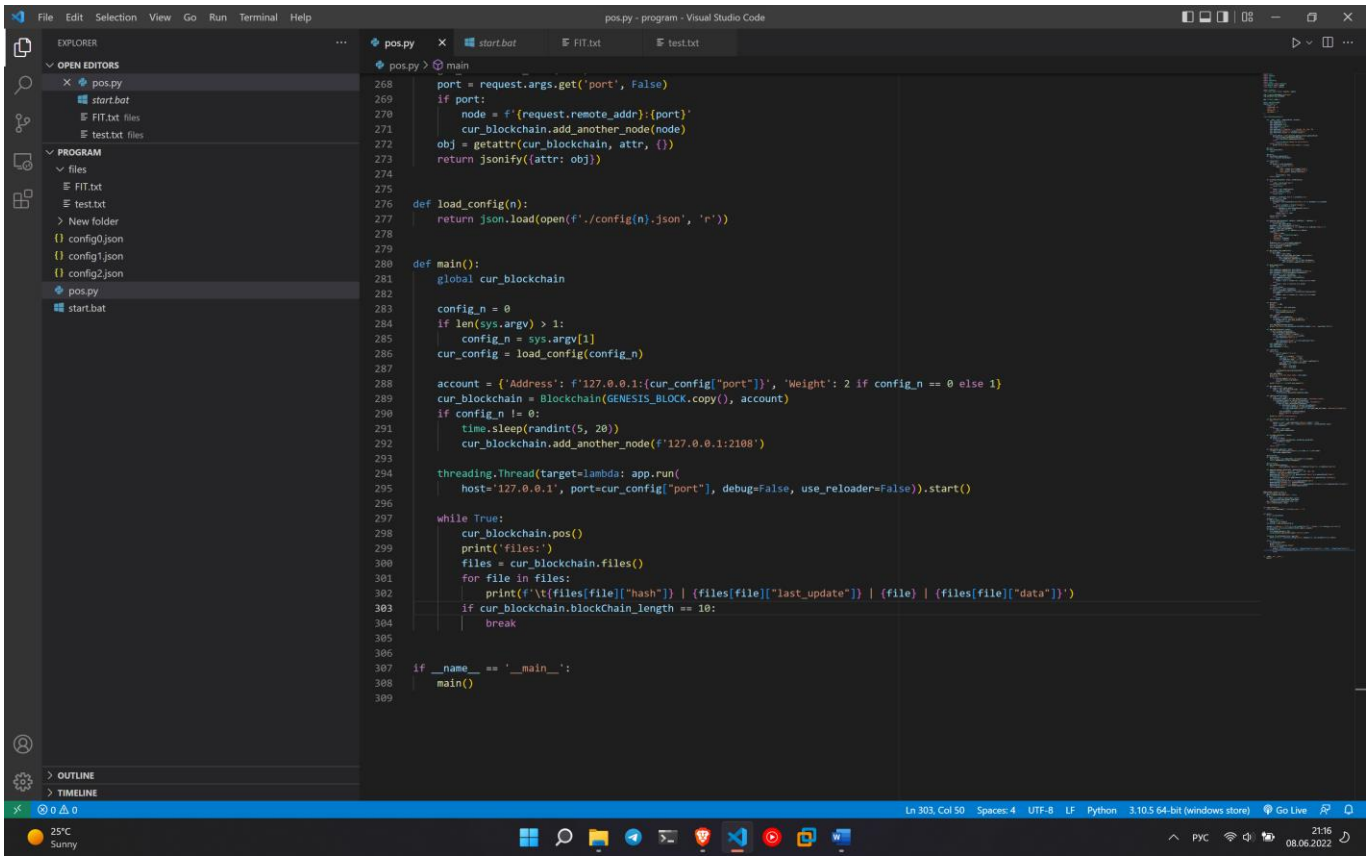

Рисунок 3.1 – Інтегроване середовище Visual Code

Для комфортної роботи стандартного функціоналу може не вистачати. У таких випадках користувачі використовують плагіни для збільшення переліку доступних можливостей при роботі з інтегрованим середовищем розробки. Для встановлення таких плагінів існує вбудоване в Visual Code місце, де є можливість переглядати доступні плагіни, їх детальний опис. Для встановлення необхідного плагіну достатньо знайти його в загальному переліку на натиснути кнопку «встановити».

Встановлено наступний перелік плагінів:

1. Code runner. Дозволяє виконувати не весь код, а тільки виділену частину. Це допомогає розробникам швидше знайти недоліки в коді та виконати лише проблемну частину, що в свою чергу підвищує продуктивність написання програмного коду.

4. Python. Даний плагін відповідає за підтримку мови програмування версії 3.7+, відладку коду, проведення автоматизованого тестування.

5. Pylance. Розширення для плагіну Python. Pylance працює на основі Pyright, інструменту статичної перевірки типів від Microsoft. Додає перевірку написаного коду «на льоту», що сприяє написанню більш якісного коду.

6. Remote SSH. Дозволяє керувати будь-яким комп’ютером через протокол SSH. За допомогою даного плагіну можна розробляти програмні продукти, використовуючи потужності стороннього комп’ютеру, також надає можливість налаштовувати віддалені комп'ютери, встановлювати необхідне ПЗ, розгортати власні програмні системи, використовувати можливості командного рядку не покидаючи середовище Visual Code.

Мовою програмування обрано Python. На даний момент це одна з найпопулярніших мов для написання скриптів, що найбільш стрімко розвиваються. Ця мова програмування має простий синтаксис, динамічну типізацію об’єктів та великий перелік якісних та безкоштовних бібліотек, що сприяє доступності використання для людей, які не мали справу з програмуванням. Python використовують для написання різного роду програм: веб-додатки, штучний інтелект, автоматизовані системи, мобільні додатки, декстопні програми з інтерфейсом тощо.

3.2 Особливості проектованого програмного забезпечення

Дві найважливіші особливості блокчейну – це децентралізація та незмінність записів. Це розподілена база даних, яку комп'ютери в мережі, які називають вузлами, ведуть спільно. Мережу не можна зруйнувати, вивівши з ладу якийсь центральний сервер.

Записи в блокчейні, звані блоками, пов'язані між собою за допомогою протокольної програми, і жоден існуючий блок не може бути видалений або змінений. Додавання нового блоку - єдиний спосіб оновити блокчейн і будь-який вузол може зробити це без будь-якого центрального органу.

Проектоване програмне забезпечення являє собою окремий вузол мережі блокчейн. Паралельно працюючі вузли на різних пристроях утворюють децентралізовану мережу (Рисунок 3.2). Порівнюючи з централізованою мережею децентралізація значно покращує захист даних у блокчейні: інформація не зберігається на центральному сервері, а продубльована на кожному окремому вузлі.

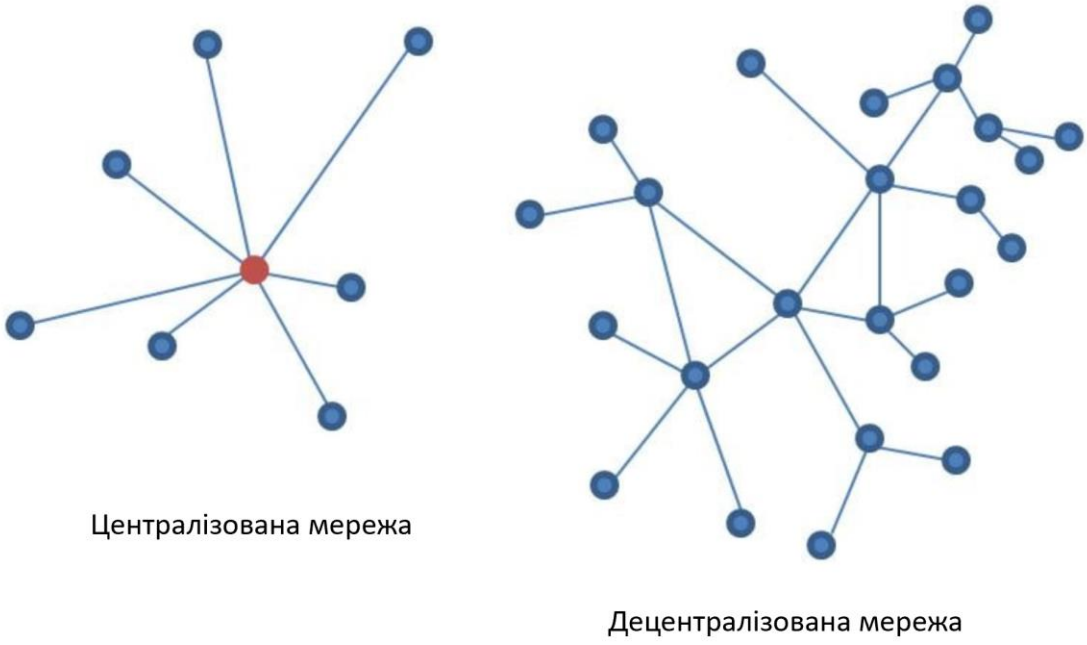

Рисунок 3.2 – Порівняння централізованої і децентралізованої мережі

Коли користувач блокчейну намагається додати новий блок даних до блокчейну, він розсилає блок усім вузлам мережі. На підставі легітимності блоку вузли можуть прийняти або відхилити блок. Коли вузол приймає новий блок транзакцій, він зберігає його і зберігає поверх інших блоків, які вже є. Окремі вузли виконують наступні функції:

- Перевіряють, чи є блок даних дійсним і приймають або відхиляють його.
- Валідують та зберігають блоки транзакцій (зберігають історію блокчейну).
- Вузли транслюють і поширюють цю історію іншим вузлам, яким може бути потрібна синхронізація з блокчейном (необхідно отримати свіжу інформацію про історію).

Блокчейн зберігає дані у ланцюжку послідовних блоків (Рисунок 3.3). Блок складається з декількох елементів:

- Дані. В блок записуються інформація про збережені файли у системі, що відслідковуються. Це може бути хеш сума файлу, його ім’я та наповнення. Також в дану секцію заноситься деяка системна інформація програмного забезпечення (час створення блоку тощо).
- Хеш попереднього блоку.
- Хеш створюваного блоку, який обчислюється на основі попереднього хешу та даних нового блоку. Таким чином утворюється рекурсивна залежність послідовно створених блоків.
-

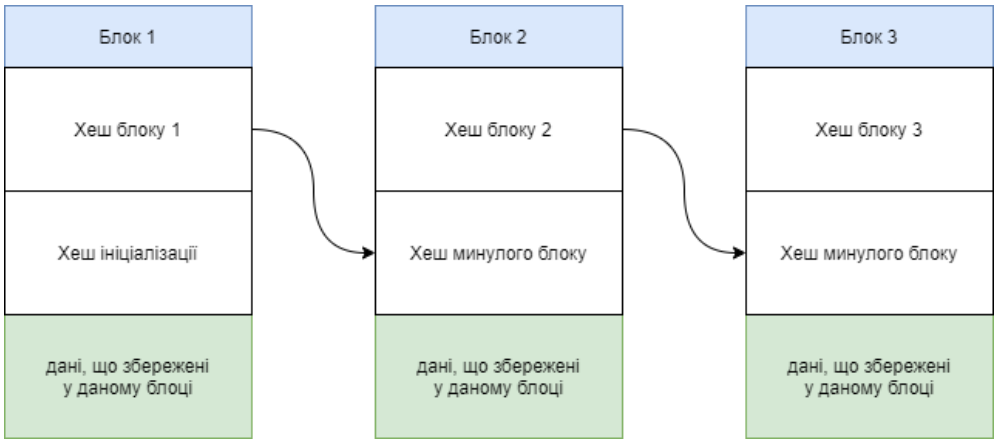

Рисунок 3.3 – Ланцюжок блоків блокчейну

3.3 Вибір алгоритму консенсусу

Якщо вузол нехтує визначеними стандартами і створює блок, інші вузли ігнорують його. Однак якщо вузол, що не відповідає стандартам, продовжує створювати блоки порушуючи стандарти, а кілька інших вузлів також починають створювати блоки поверх блоків, що не відповідають стандартам, то у системі виникає суперечка паралельно створених ланцюжків. Спільнота може вибрати жорсткий форк і повернути стан мережі до виникнення суперечки, однак часті жорсткі форки впливають на стабільність. Необхідний механізм консенсусу для запобігання таким несумісним вузлам, що викликають часті жорсткі форки.

Також можуть існувати шкідливі вузли, які перевантажують інші вузли мережі, використовуючи атаку "розподілена відмова в обслуговуванні" (DDoS). Для захисту від цього потрібний механізм консенсусу. Існує безліч алгоритмів досягнення консенсусу, але з них можна виділити 2 основних, від яких і походять всі інші:

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

Для реалізації власного програмного забезпечення обрано алгоритм Proof of Stake (PoS) через більшу доступність та значно меншу витрату ресурсів.

3.3.1 Алгоритм консенсусу Proof of Work (PoW)

Синтія Дворк та Моні Наор вперше розробили концепцію POW у 1993 році, хоча Маркус Якобссон дав їй назву ще у 1999 році. Біткойн – найвідоміша реалізація POW.

Транзакції в блокчейні Біткойна групуються в пулі пам'яті під назвою "mempool", блок створюється кожні 10 хвилин. Кожна транзакція в mempool потребує перевірки, і цим займаються "майнери". Процес перевірки транзакцій називається "майнінг".

Користувач Bitcoin, який запитує транзакцію, надає дані транзакції майнеру, який потім перевіряє транзакцію та включає її до наступного доступного блоку. Однак для включення транзакції в наступний блок майнер необхідно знати криптографічне хеш-значення останнього записаного блоку, яке приховано від усіх. На це хеш-значення необхідно посилатися під час створення нового блоку.

Щоб знайти хеш останнього блоку, майнер повинен перебирати одне число за іншим, демонструючи грубу обчислювальну силу, при цьому не потрібно жодних навичок. Майнери отримують винагороду у вигляді частки біткойну, тому це конкурентний процес. Успішним майнером стає той, хто перемагає решту в цій грі і вирішує цю масивну математичну головоломку, використовуючи величезну обчислювальну потужність. Після знаходження хеша останнього записаного блоку майнер повідомляє його в мережу для перевірки іншими вузлами та створює новий блок із транзакціями в mempool після перевірки.

Криптографічна головоломка, яку вирішує майнер, є асиметричною. Вона в міру складна для майнера, але інші вузли в мережі можуть легко побачити докази того, що було проведено потужний перебір чисел. З часом майнери знаходять головоломку легшою, і час генерації блоку скорочується з 10 хвилин. Таким чином, кожні 14 днів головоломка переглядається, щоб зробити її складнішою. Це фактично означає, що надалі потрібно більше обчислювальної потужності.

Провести DDoS-атаку, захопивши 51% від загальної обчислювальної потужності мережі, дуже дорого, і зловмисник зрештою витратить більше, ніж зможе отримати. Таким чином, POW робить блокчейн дуже безпечним.

Однак, за цю високу безпеку доводиться дорого платити. Обчислювальна потужність вузлів, що постійно зростає, вимагає величезну кількість електроенергії.

Крім того, окремим майнерам важко постійно модернізувати своє обладнання для вирішення все більш складних математичних головоломок та оплачувати рахунки, що ростуть, за електроенергію. Відтак зростає централізація майнінгу, коли на сцені домінують великі організовані майнінгові установки. Така непряма централізація суперечить основному принципу блокчейну – децентралізації.

3.3.2 Алгоритм консенсусу Proof of Stake (PoS)

На відміну від Pow у PoS мережа вузлів вирішує поставити в залог свої власні активи для підтвердження транзакції. Вони називаються «стейкери». Чим більша сума ставки і що довше термін дії ставки, то вище шанси стейкера отримати відповідальність за підтвердження транзакції.

В такій мережі вже майнінг відсутній. Це позбавляє необхідності вирішувати складну криптографічну головоломку. Також відпадає потреба у постійному оновленні апаратного забезпечення та різкому зростанні витрат на електроенергію. Процес підтвердження транзакцій називається «кування» (forging).

Крім того, немає необхідності залучати всю мережу до процесу підтвердження транзакцій, що покращує масштабованість.

3.4 Поетапне проектування власного блокчейну

Програмне забезпечення має єдиний об’єкт – «Blockchain» (Рисунок 3.4). Це самостійний об’єкт, створений на основі одноіменного класу власної розробки, який має усе необхідне для функціонування та створення децентралізованої мережі блокчейн.

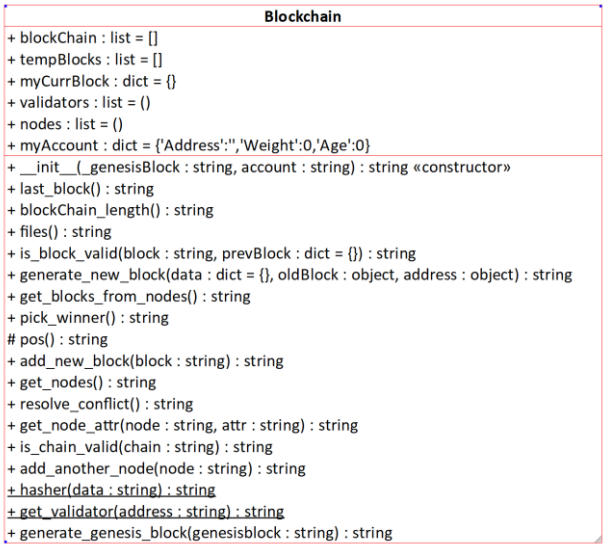

Рисунок 3.4 – Клас «Blockchain»

Даний клас має 6 глобальних змінних:

- blockChain – список, що зберігає легітимний ланцюжок послідовно створених блоків даних.
- tempBlocks – словник, тимчасове сховище створених різними вузлами блоків для подальшого вибору блоку-переможця, що продовжить поточний ланцюжок.
- myCurrBlock – словник, останній згенерований блок поточним вузлом.
- validators – список доступних валідаторів.
- nodes – список доступних вузлів, між якими відбуваєтья обмін блоками за алгоритмом консенсусу.
- MyAccount – дані про поточний вузол, зберігає статистичну інформацію, яка приймає участь в алгоритмі консенсусу для вибору наступного валідатора ланцюжку блокчейн.

Основні функції класу:

- Функція ініціалізації класу *__init__*. Дана функція створює перший блок у ланцюжку (genesis block), який є основою всіх наступних блоків, також функція ініціалізує значення статистики використання вузла:

```
def __init__(self, _genesisBlock, account):
...self.blockChain = []
...self.tempBlocks = []
...self.myCurrBlock = {}
...self.validators = list()
...self.nodes = list()
...self.myAccount = {'Address': '', 'Weight': 0, 'Age': 0}
...self.myAccount['Address'] = account['Address']
...self.myAccount['Weight'] = account['Weight']
...try:
……genesisBlock = self.generate_genesis_block(_genesisBlock)
……if self.is_block_valid(genesisBlock):
……….self.blockChain.append(genesisBlock)
```

……else:

……….raise Exception('Unable to verify block')

…except Exception as e:

……print('Invalid genesis block.\nOR\n' + str(e))

- Функція *files*. За допомогою функції можна в будь-який момент часу отримати словник файлів, що зберігаються в блокчейні, а також отримати додаткові дані про файли (ім’я, історія та дату зміни файлу, його хеш-суму):

def files(self):

…files = {}

…for block in self.blockChain:

……for name in block['Data']:

……….file = {

……….'hash': block['Data'][name]['hash'],

……….'data': block['Data'][name]['data'],

……….'last_update': block['Timestamp']

……….}

……….files[name] = file

…return files

- Функція *is_block_valid*. Функція перевіряє переданий у неї новий блок на коректність і можливість додавання у поточний ланцюжок за допомогою власного перерахунку хешу нового блоку на основі останнього доступного в блокчейні:

def is_block_valid(self, block, prevBlock={}):

…try:

……_hash = block.pop('Hash')

…except KeyError as e:

……return False

…try:

……hash2 = self.hasher(block)

……assert _hash == hash2

…except AssertionError as e:

```
……return False
...prevHash = prevBlock['Hash'] if prevBlock else ''
...block['Hash'] = _hash
...if self.blockChain:
……prevHash = self.blockChain[-1]['Hash'] if not prevHash else prevHash
……try:
……….assert prevHash == block['PrevHash']
……except AssertionError as e:
……….if prevHash == self.blockChain[0]['Hash']:
…….……block['Hash'] = _hash
…….……return True
……….block['Hash'] = _hash
……….return False
...block['Hash'] = _hash
...return True
```

- Функція *generate_new_block*. Функція генерує новий блок з переданими у неї даними (інформація про файли) та обчислює його хеш-суму. Наступним кроком цей блок передається у доступні вузли для подальшого виконання алгоритму досягнення консенсусу:

```
def generate_new_block(self, data={}, oldBlock='', address=''):
...if self.myCurrBlock:
……return self.myCurrBlock
...prevHash = self.blockChain[-1]['Hash']
...index = len(self.blockChain) if not oldBlock else oldBlock['Index'] + 1
...address = self.get_validator(
……self.myAccount) if not address else address
...newBlock = {
……'Index': index,
……'Timestamp': str(datetime.now()),
……'Data': data,
```

……'PrevHash': prevHash,

……'Validator': address

...}

...newBlock['Hash'] = self.hasher(newBlock)

...assert self.is_block_valid(newBlock)

...self.myCurrBlock = newBlock

...return newBlock

- Функція *get_blocks_from_nodes*. Блокчейн, використовуючи дану функцію та інтерфейс взаємодії між вузлами, отримує останні згенеровані блоки доступних вузлів:

def get_blocks_from_nodes(self):

...if self.nodes:

……for node in self.nodes:

……….resp = self.get_node_attr(node, 'myCurrBlock')

……….if self.is_block_valid(resp):

…….…….self.tempBlocks.append(resp)

…….…….if resp['Validator'] not in self.validators:

…….……..self.validators.append(resp['Validator'])

- Функція *pick_winner*. Функція опираючись на список доступних валідаторів та їх статистику обирає чий саме згенерований блок стане наступним у ланцюжку блокчейн:

def pick_winner(self):

...winner = []

...self.tempBlocks.append(self.myCurrBlock)

...self.validators.append(self.myCurrBlock['Validator'])

...self.validators = list(sorted(self.validators))

...for validator in self.validators:

……acct = (validator.rsplit(sep=', '))

*……acct.append(int(acct[1]) * int(acct[2]))*

……if winner and acct[-1]:

……winner = acct if winner[-1] < acct[-1] else winner

……else:

……winner = acct if acct[-1] else winner

…if winner:

……return winner

…for validator in self.validators:

……acct = (validator.rsplit(sep=', '))

……acct.append((int(acct[1]) + int(acct[2]))/len(acct[0]))

……if winner:

……winner = acct if winner[-1] < acct[-1] else winner

……else:

……winner = acct

…return winner

- Функція *get_nodes*. Використовується для отримання списку вузлів з інших вузлів до яких є доступ:

def get_nodes(self):

…for node in self.nodes.copy():

……nodes = self.get_node_attr(node, 'nodes')

……for new_node in nodes:

……cur_blockchain.add_another_node(new_node)

- Функція *resolve_conflict*. Перед генерацією нового блоку кожен блокчейн порівнюється з сусіднім та уразі виникнення конфлікту викликається дана функція. Вона вираховує у якому саме місці виникає помилка, після чого порівнюється авторитет (значущість валідатора помножене на його час перебування у мережі) валідаторів вузлів. Валідатор, у якого авторитет нижче, копіює останні блоки (починаючи з конфіктного) та заміняє ними свої:

def resolve_conflict(self):

…for node in self.nodes:

……blockChain_length = self.get_node_attr(node, 'blockChain_length')

……if blockChain_length >= len(self.blockChain):

……node_blockChain = self.get_node_attr(node, 'blockChain')

……if self.is_chain_valid(node_blockChain):

……….if (blockChain_length == len(self.blockChain))

……….and (self.blockChain != node_blockChain)

……….and (self.myAccount['Weight'] > self.get_node_attr(node, 'myAccount')['Weight']):

……….return

……….self.blockChain = node_blockChain

……….print('Chain is replaced!!!')

……….return

...print('My chain is authoritative')

- Функція *get_node_attr*. Функція є частиною інтерфейсу взаємодії вузлів, використовуючи протокол HTTP. Використовується для побудови відповідних запитів для отримання інформації з іншого вузла:

def get_node_attr(self, node, attr):

...try:

……params = {'port': self.myAccount['Address'].split(':')[1]}

……resp = requests.get(f'http://{node}/getattr/{attr}', params=params).json()

……return resp[attr]

...except:

……if node in self.nodes:

……..self.nodes.remove(node)

……return {}

- Функція *is_chain_valid*. Функція реалізує перевірка поточного ланцюжку блокчейн за рахунок послідовного переобчислення всіх хеш-сум блоків:

def is_chain_valid(self, chain):

..._prevBlock = ''

...for block in chain:

……if self.is_block_valid(block, prevBlock=_prevBlock):

……..._prevBlock = block

……else:

……..return False

...return True

- Функція *hasher*. Функція використовується для обчислення хеш-сум переданих у неї даних за допомогою алгоритму sha-256:

def hasher(data):

...data_string = json.dumps(data, sort_keys=True).encode()

...return sha256(data_string).hexdigest()

- Функція *get_blockchain_attr*. Реалізує відповідач на запити інших вузлів для передачі необхідних даних між ними:

def get_blockchain_attr(attr):

...port = request.args.get('port', False)

...if port:

……node = f'{request.remote_addr}:{port}'

……cur_blockchain.add_another_node(node)

...obj = getattr(cur_blockchain, attr, {})

...return jsonify({attr: obj})

3.5 Тестування розробленого програмного забезпечення та демонстрація його роботи

Вектор методів тестування поступово зміщується у бік автоматизації, проте ручне тестування є досить популярним. В загальному випадку ручне тестування являє собою пошук недосконалостей програмного забезпечення під час якого фахівець безпосередньо вручну проводить перевірку працездатності програмних компонентів засобами моделювання користувацької поведінки.

Ручне тестування дає можливість отримати перший відгук від потенційного клієнта – тестера, що допоможе зрозуміти наскільки продукт зручних для кінцевого користувача. Є можливість протестувати функціонал програми та виявити її

недоліки. Дослідницьке тестування і можливість імпровізації дозволяє перевірити потенціал програми ширше та переглянути нетипові сценарії її роботи.

Тест-кейс – це перевірка працездатності проєкту або програми за здалегіть створеною інструкцією. Процес написання тест-кейсу являє собою створення текстового опису процесу тестування частини програми або її вцілому. Необхідність тест-кейсу зумовлена потребою перевірки програми, ознайомлення з нею членів команди без читання усього програмного коду, тільки вивчивши зміст тест-кейса.

Серед результатів тест-кейсів виокремлюють:

- Позитивний результат. Дає розуміння розробникам і тестувальнику, що очікувані результати роботи програми збігаються з фактичними.
- Негативний результат. Показує що отриманий результат є відмінним від очікуваного.
- Не завершений результат. Відображає наявність помилки в процесі проведення тестування.

Існують вимоги написання тест-кейсів, що допомагають чітко виокремити алгоритм, за яким необхідно працювати. В загальному випадку можна виділити наступні характеристики якісного тест-кейса:

- Один тест-кейс перевіряє працездатність тільки однієї конкретної речі.
- Тест-кейс не повинен залежати від інших тест-кейсів.
- Чіткість і конкретність очікуваних результатів є запорукою однозначної класифікації результату тесту.
- Тест-кейс повинен містити всю інформацію для його виконання.
- Алгоритм без зайвих деталей.
- Вказання типу даних є важливою частиною формування тест-кейса.

Тестування вручну є менш фінансово дорогим і свої результати дає практично від самого початку проведення робіт. Підтримка ручних тестів для програми з частою зміною функціоналу та наповнення є ідеальним рішенням. Тест-кейси, які допомагають подивитися на роботу продукту під різним кутом тільки допоможуть провести якісний аналіз і виявити слабкі та непередбачені деталі, що в майбутньому позитивно відобразиться на загальному користувачем сприйнятті та досвіді

використання прогарами. Не зважаючи на той факт, що ручні тести вимагають більше часу на своє виконання і відпрацювання та мають ризик «людського» фактора вони є досить популярними серед компаній через свою гнучкість та швидку оцінку для керівників проєкту.

Я протестував створений додаток вручну, за допомогою тест-кейсів. Сам тестовий випадок складається з трьох частин:

1. передумова – список кроків, які виконуються безпосередньо перед початком самого тестування для отримання необхідного середовища;

2. опис тестового випадку – головна частина тест-кейсу, список дій, за допомогою яких здійснюється перевірка функціоналу;

3. постумова – список дій, які повертають систему стан до початку процесу тестування.

У ході тестування створеного програмного комплексу розроблено та перевірені наступні тест-кейси:

1. Перевірка досягнення декількома вузлами консенсусу (Таблиця 3.1).

2. Перевірка занесення в блокчейн актуальної інформації про файли, що відстежуються (Таблиця 3.2).

Таблиця 3.1

Перевірка досягнення декількома вузлами консенсусу.

Передумова		
Підготувати 3 комп’ютери в локальній мережі та встановити на них розроблене програмне забезпечення		
Подія	**Очікуваний результат**	**Фактичний результат**
Тестовий випадок		
Запустити на кожному комп’ютері екземпляр вузлу	Під час запуску необхідно переконатися що всі вузли з’єдналися між собою в є дину систему та почали	Пройдено.

	генерувати блоки.	
Подія	**Очікуваний результат**	**Фактичний результат**
	усіх консолей та зробити висновок чи збігаються дані отримані від різних вузлів.	
Постумова		
Закрити консолі запущених вузлів блокчейну, видалити встановлене програмне забезпечення.		

Тест-кейс успішно пройдений. Процес проходження зображено на Рисунку 3.5

Рисунок 3.5 – Скріншоти виводу різних вузлів в консоль

Таблиця 3.2

Перевірка занесення в блокчейн
актуальної інформаціїпро файли, що відстежуються.

Передумова		
Підготувати 3 комп’ютери в локальній мережі та встановити на них розроблене програмне забезпечення		
Подія	**Очікуваний результат**	**Фактичний результат**

Тестовий випадок		
Подія	**Очікуваний результат**	**Фактичний результат**
Запустити на кожному комп'ютері екземпляр вузлу	Під час запуску необхідно переконатися що всі вузли з'єдналися між собою в є дину систему та почали генерувати блоки.	Пройдено.
У каталозі «files» додати два тестових файлів для відслідкування	У відкритих консолях з'явився запис про щойно додані файл.	Пройдено.
Внести зміну у один із тестових файлів	На протязі наступних п'яти секунд вивід даних в консоль про файли повинен змінитися (хеш-сума, дата останньої перевірки файлу)	
Постумова		
Закрити консолі запущених вузлів блокчейну, видалити встановлене програмне забезпечення.		

Тест-кейс успішно пройдений. Процес проходження зображено на Рисунку 3.6

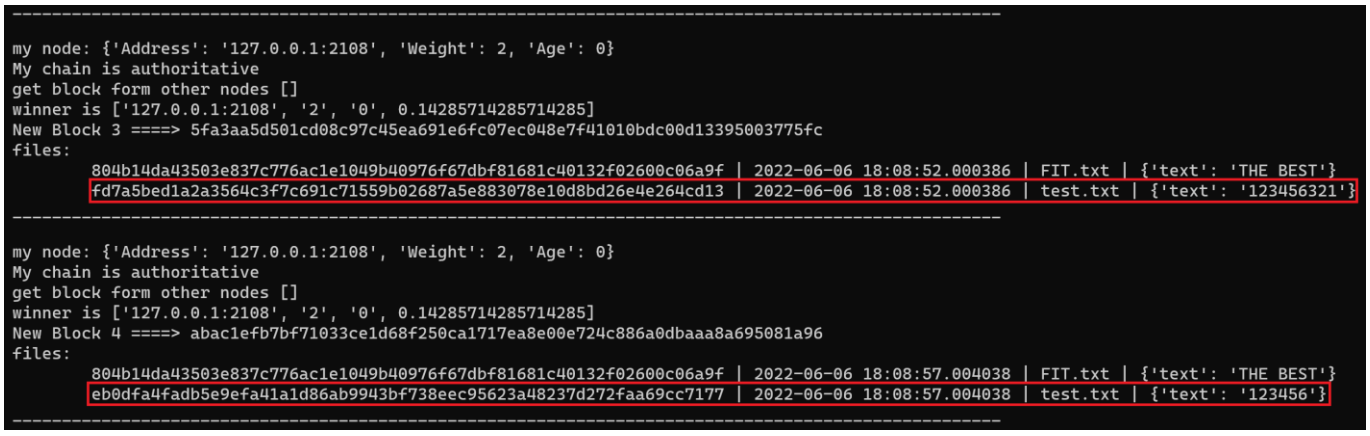

Рисунок 3.6 – Скріншоти виводу різних вузлів в консоль

Висновки за розділом 3

У третьому розділі описано поетапну розробку власного програмного забезпечення для збереження цілісності даних з використання технології блокчейн. Наведено інформацію про середовище розробки, вибір мови програмування та інших інструментів розробки для виконання поставленої задачі. Проведено тестування з використанням наступних тест-кейсів:

1. Перевірка досягнення декількома вузлами консенсусу.
2. Перевірка занесення в блокчейн актуальної інформації про файли, що відстежуються.

Усі поставлені тест-кейси пройдені успішно.

ВИСНОВКИ

Досліджено архітектури і різновиди технології блокчейн на прикладі реально існуючих. Оглянуто загальний принцип роботи. Наведено переваги та недоліки використання даної технології та приклади їх використання у таких сферах як:

- Банківська справа та фінанси.
- Харчова промисловість
- Охорона здоров'я.
- Записи про власність.
- Валюта.
- Ланцюжки поставок.
- Голосування.

Проведено аналіз використання блокчейн для побудови розподіленого реєстру даних. Розкрито проблематику збереження цілісності даних та способи їх вирішення з використанням технології блокчейн. Оглянуто популярні децентралізовані сервіси збереження даних в інтернеті, наведено їх переваги й недоліки.

Описано вимоги до розробляємого програмного забезпечення. ПЗ повинно генерувати блоки кожні 5 секунд, заносити в блок інформацію про файли, що підлягають моніторингу, досягати консенсусу з іншими вузлами, у разі виявленні конфліктів виправляти помилки у автоматичному режимі.

Розроблено програмне забезпечення для збереження цілісності інформації використовуючи технологію блокчейн. Описано поетапну розробку власного програмного забезпечення для збереження цілісності даних з використання технології блокчейн. Наведено інформацію про середовище розробки, вибір мови програмування та інших інструментів розробки для виконання поставленої задачі. Для реалізації власного програмного забезпечення обрано алгоритм Proof of Stake (PoS) через більшу доступність та значно меншу витрату ресурсів. ПЗ виконує наступні функції:

- Перевіряє, чи є блок даних дійсним і приймають або відхиляють його.

- Валідує та зберігає блоки транзакцій (зберігають історію блокчейну).
- Транслює і поширюють цю історію іншим вузлам, яким може бути потрібна синхронізація з блокчейном (необхідно отримати свіжу інформацію про історію).

В блок записуються інформація про збережені файли у системі, що відслідковуються. Це може бути хеш сума файлу, його ім’я та наповнення. Також в дану секцію заноситься деяка системна інформація програмного забезпечення (час створення блоку тощо).

Протестовано створене програмне забезпечення вручну, за допомогою створених тест-кейсів:

1. Перевірка досягнення декількома вузлами консенсусу.
2. Перевірка занесення в блокчейн актуальної інформації про файли, що відстежуються.

Усі поставлені тест-кейси пройдені успішно.

Результатом є розроблене програмне забезпечення для збереження цілісності даних в розподілених інформаційних системах на базі технології Blockchain. Дане ПЗ може використовуватись в приватних підприємствах для створення власного розподіленого реєстру файлів для записів логів системи. Це значно підвищує складність зловмисникам провести такі несанкціановані дії як редагування/видалення файлів або додавання зайвого. Також у випадку зламу системи за допомогою блокчейну можа провести детальний аналіз історії змін файлів, щоб скласти більш повну картину ситуації, що сталась.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Класифікація видів блокчейну [Електронний ресурс] – Режим доступу до ресурсу: https://polygant.net/ru/blog/vidy-blokchejna/.

2. Три поширені види блокчейна [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://bitnovosti.com/2018/04/25/3-popular-types-of-blockchains/.

3. Як використовувати блокчейн для зберігання інформації [Електронний ресурс] – Режим доступу до ресурсу: https://merehead.com/ru/blog/how-to-use-blockchain-to-store-data/.

4. How Blockchain protects system integrity [Електронний ресурс] – Режим доступу до ресурсу: https://blog.simbachain.com/blog/how-blockchain-protects-system-integrity.

5. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system [Електронний ресурс] – Режим доступу: https://bitcoin.org/bitcoin.pdf.

6. Ethereum: A next-generation smart contract and decentralized application platform [Електронний ресурс] – Режим доступу: https://github.com/ethereum/wiki/wiki/White-Paper.

7. Szabo N. The idea of smart contracts [Електронний ресурс] – Режим доступу: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/ Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_idea.html.

8. Weber R. Audit trail system support in advanced computer-based accounting systems [Електронний ресурс] – Режим доступу: https://www.proquest.com/docview/1301314968.

9. Benet. Ipfs - content addressed, versioned, p2p file system (draft 3) [Електронний ресурс] – Режим доступу: https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf.

10. Wood G. Ethereum: A secure decentralised generalised transaction ledger [Електронний ресурс] / G. Wood. – Режим доступу: https://ethereum.github.io/yellowpaper/paper.pdf.

11. Ensuring data integrity using blockchain technology [Електронний ресурс] / I. Zikratov [та ін.] // IEEE Xplore. – Режим доступу: https://ieeexplore.ieee.org/document/8071359/.

12. Barinov I. System and method for verifying data integrity using a blockchain network [Електронний ресурс] / I. Barinov, V. Lysenko, S. Beloussov. – Режим доступу: http://www.freepatentsonline.com/y2018/0025181.html.

13. Hayes A. Blockchain explained [Електронний ресурс] / Adam Hayes. – Режим доступу: https://www.investopedia.com/terms/b/blockchain.asp.

14. POW vs. PoS: a comparison of two blockchain consensus algorithms [Електронний ресурс]. – Режим доступу: https://medium.com/EdChain/pow-vs-pos-a-comparison-of-two-blockchain-consensus-algorithms-f3effdae55f5.

15. Roberts P. Can blockchain solve data's integrity problem? [Електронний ресурс] / Paul Roberts. – Режим доступу: https://securityledger.com/2021/04/can-blockchain-solve-datas-integrity-problem/.

16. Crosby M. Blockchain technology: Beyond bitcoin. Applied Innovation [Електронний ресурс] / M. Crosby, P. Pattanayak, S. Verma. – Режим доступу: https://j2-capital.com/wp-content/uploads/2017/11/AIR-2016-Blockchain.pdf.

17. Yakov V. Use of Blockchain for Ensuring Data Integrity in Cloud Databases [Електронний ресурс] / Vainshtein Yakov, Gudes Ehud // SpringerLink. – Режим доступу: https://doi.org/10.1007/978-3-030-78086-9_25.

18. Ristenpart T. Hey, you, get off of my cloud | Proceedings of the 16th ACM conference on Computer and communications security [Електронний ресурс] / T. Ristenpart // ACM Conferences. – Режим доступу: https://doi.org/10.1145/1653662.1653687.

19. Blockchain-Based Cloud Data Integrity Verification Scheme with High Efficiency [Електронний ресурс] / Gaopeng Xie [та ін.] // Security and Communication Networks. – 2021. – Т. 2021. – С. 1–15. – Режим доступу: https://doi.org/10.1155/2021/9921209.

20. Controllable and trustworthy blockchain-based cloud data management [Електронний ресурс] / Liehuang Zhu [та ін.] // Future Generation Computer Systems. –

2019. – Т. 91. – С. 527–535. – Режим доступу: https://doi.org/10.1016/j.future.2018.09.019.

21. LSB: A Lightweight Scalable Blockchain for IoT security and anonymity [Електронний ресурс] / Ali Dorri [та ін.] // Journal of Parallel and Distributed Computing. – 2019. – Т. 134. – С. 180–197. – Режим доступу: https://doi.org/10.1016/j.jpdc.2019.08.005.

22. Haug C. J. Peer-Review Fraud – Hacking the Scientific Publication Process [Електронний ресурс] / Charlotte J. Haug // New England Journal of Medicine. – 2015. – Т. 373, № 25. – С. 2393–2395. – Режим доступу: https://doi.org/10.1056/nejmp1512330.

23. A Distributed-Ledger Consortium Model for Collaborative Innovation [Електронний ресурс] / Chris Khan [та ін.] // Computer. – 2017. – Т. 50, № 9. – С. 29–37. – Режим доступу: https://doi.org/10.1109/mc.2017.3571057.

24. Wang H. Blockchain-Based Private Provable Data Possession [Електронний ресурс] / Huaqun Wang, Qihua Wang, Debiao He // IEEE Transactions on Dependable and Secure Computing. – 2019. – С. 1. – Режим доступу: https://doi.org/10.1109/tdsc.2019.2949809.

25. Blockchain challenges and opportunities: a survey [Електронний ресурс] / Huaimin Wang [та ін.] // International Journal of Web and Grid Services. – 2018. – Т. 14, № 4. – С. 352. – Режим доступу: https://doi.org/10.1504/ijwgs.2018.10016848.

26. Ensuring data integrity using blockchain technology [Електронний ресурс] / Igor Zikratov [та ін.] // 2017 20th Conference of Open Innovations Association (FRUCT), St-Petersburg, Russia, 3–7 квіт. 2017 р. – [Б. м.], 2017. – Режим доступу: https://doi.org/10.23919/fruct.2017.8071359.

27. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends [Електронний ресурс] / Zibin Zheng [та ін.] // 2017 IEEE International Congress on Big Data (BigData Congress), Honolulu, HI, USA, 25–30 черв. 2017 р. – [Б. м.], 2017. – Режим доступу: https://doi.org/10.1109/bigdatacongress.2017.85.

28. Blockchain Business and Its Regulation [Електронний ресурс] / Makoto Yano [та ін.] // Economics, Law, and Institutions in Asia Pacific. – Singapore, 2020. – С. 107–127. – Режим доступу: https://doi.org/10.1007/978-981-15-3376-1_7/.

29. Blockchain Technology: A Review of the Current Challenges of Cryptocurrency [Електронний ресурс] / Diego Valdeolmillos [та ін.] // Advances in Intelligent Systems and Computing. – Cham, 2019. – С. 153–160. – Режим доступу: https://doi.org/10.1007/978-3-030-23813-1_19.

30. Tan B. S. Blockchain as the Database Engine in the Accounting System [Електронний ресурс] / Boon Seng Tan, Kin Yew Low // Australian Accounting Review. – 2019. – Т. 29, № 2. – С. 312–318. – Режим доступу: https://doi.org/10.1111/auar.12278.

31. Blockchain for next generation services in banking and finance: cost, benefit, risk and opportunity analysis [Електронний ресурс] / Mohamad Osmani [та ін.] // Journal of Enterprise Information Management. – 2020. – Ahead-of-print, ahead-of-print. – Режим доступу: https://doi.org/10.1108/jeim-02-2020-0044.

ДОДАТОК А
СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИПЛОМНОЇ РОБОТИ

Тези наукових доповідей:

Даков С.Ю, Малишев К.Е. Логування дій в інформаційних системах з використанням технології blockchain. Міжнародна науково-практична конференція «Прикладні системи та технології в інформаційному суспільстві». 30 вересня 2021 року.

ДОДАТОК Б
ПРОГРАМНИЙ КОД РОЗРОБЛЕНОГО ЗАСТОСУНКУ

```
import json
import logging
import os
import sys
import threading
import time
from datetime import datetime
from hashlib import sha256
from random import randint

import requests
from flask import Flask, jsonify, request

log = logging.getLogger('werkzeug')
log.setLevel(logging.ERROR)

app = Flask(__name__)

GENESIS_BLOCK = {
...'Index': 0,
...'Timestamp': 0,
...'Data': {},
...'PrevHash': '',
...'Validator': ''
}

class Blockchain(object):

...def __init__(self, _genesisBlock, account):
……self.blockChain = []
……self.tempBlocks = []
……self.myCurrBlock = {}
……self.validators = list()
……self.nodes = list()
……self.myAccount = {'Address': '', 'Weight': 0, 'Age': 0}
……self.myAccount['Address'] = account['Address']
……self.myAccount['Weight'] = account['Weight']
……try:
……..genesisBlock = self.generate_genesis_block(_genesisBlock)
……..if self.is_block_valid(genesisBlock):
……….self.blockChain.append(genesisBlock)
……..else:
……….raise Exception('Unable to verify block')
……except Exception as e:
……..print('Invalid genesis block.\nOR\n' + str(e))
```

```
...@property
...def last_block(self):
.....return

...@property
...def blockChain_length(self):
.....return len(self.blockChain)

...def files(self):
.....files = {}
.....for block in self.blockChain:
.........for name in block['Data']:
.............file = {
.................'hash': block['Data'][name]['hash'],
.................'data': block['Data'][name]['data'],
.................'last_update': block['Timestamp']
.............}
.............files[name] = file
.....return files

...def is_block_valid(self, block, prevBlock={}):
.....try:
.........._hash = block.pop('Hash')
.....except KeyError as e:
.........return False
.....try:
.........hash2 = self.hasher(block)
.........assert _hash == hash2
.....except AssertionError as e:
.........return False

.....prevHash = prevBlock['Hash'] if prevBlock else ''
.....block['Hash'] = _hash
.....if self.blockChain:
.........prevHash = self.blockChain[-1]['Hash'] if not prevHash else prevHash
.........try:
.............assert prevHash == block['PrevHash']
.........except AssertionError as e:
.............if prevHash == self.blockChain[0]['Hash']:
.................block['Hash'] = _hash
.................return True
.............block['Hash'] = _hash
.............return False
.....block['Hash'] = _hash
.....return True

...def generate_new_block(self, data={}, oldBlock='', address=''):
.....if self.myCurrBlock:
.........return self.myCurrBlock
.....prevHash = self.blockChain[-1]['Hash']
.....index = len(self.blockChain) if not oldBlock else oldBlock['Index'] + 1
.....address = self.get_validator(
```

```
        ……self.myAccount) if not address else address
        ……newBlock = {
        ……'Index': index,
        ……'Timestamp': str(datetime.now()),
        ……'Data': data,
        ……'PrevHash': prevHash,
        ……'Validator': address
        ……}
        ……newBlock['Hash'] = self.hasher(newBlock)
        ……assert self.is_block_valid(newBlock)
        ……self.myCurrBlock = newBlock
        ……return newBlock

    ...def get_blocks_from_nodes(self):
    ……if self.nodes:
    ……for node in self.nodes:
    ……resp = self.get_node_attr(node, 'myCurrBlock')
    ……if self.is_block_valid(resp):
    ……self.tempBlocks.append(resp)
    ……if resp['Validator'] not in self.validators:
    ……self.validators.append(resp['Validator'])

    ...def pick_winner(self):
    ……winner = []

    ……self.tempBlocks.append(self.myCurrBlock)
    ……self.validators.append(self.myCurrBlock['Validator'])
    ……self.validators = list(sorted(self.validators))
    ……for validator in self.validators:
    ……acct = (validator.rsplit(sep=', '))
    ……acct.append(int(acct[1]) * int(acct[2]))
    ……if winner and acct[-1]:
    ……winner = acct if winner[-1] < acct[-1] else winner
    ……else:
    ……winner = acct if acct[-1] else winner
    ……if winner:
    ……return winner
    ……for validator in self.validators:
    ……acct = (validator.rsplit(sep=', '))
    ……acct.append((int(acct[1]) + int(acct[2]))/len(acct[0]))
    ……if winner:
    ……winner = acct if winner[-1] < acct[-1] else winner
    ……else:
    ……winner = acct
    ……return winner

    ...def pos(self):
    ……print('_' * 100)
    ……print()
    ……print('my node:', self.myAccount)
    ……while True:
    ……if int(time.time()) % 5 == 0:
```

```
……….self.resolve_conflict()
……….break
……self._pos()
……for block in self.tempBlocks:
……….validator = block['Validator'].rsplit(', ')
……….if validator[0] == self.pick_winner()[0]:
……….new_block = block
……….break
……self.add_new_block(new_block)
……print(f'New Block {cur_blockchain.blockChain_length} ====>', new_block['Hash'])

…def add_new_block(self, block):
……if self.is_block_valid(block):
……….self.blockChain.append(block)
……….acct = block['Validator'].rsplit(', ')
……….if self.myAccount['Address'] != acct[0]:
……….self.myAccount['Age'] += 1
……….else:
……….self.myAccount['Weight'] += self.myAccount['Age']
……….self.myAccount['Age'] = 0
……self.tempBlocks = []
……self.myCurrBlock = {}
……self.validators = list()

…def _pos(self):
……while True:
……….if int(time.time()) % 5 == 2:
……….data = {}
……….for name in os.listdir('./files'):
……….file_path = './files/' + name
……….with open(file_path, 'r') as f:
……….file_data = {'text': '\n'.join(f.readlines())}
……….file_hash = self.hasher(file_data)
……….data[name] = {
……….'hash': file_hash,
……….'data': file_data
……….}
……….self.generate_new_block(data=data)
……….break
……self.get_nodes()
……print('get block form other nodes', self.nodes)
……while True:
……….if int(time.time()) % 5 == 4:
……….self.get_blocks_from_nodes()
……….break
……print('winner is', str(self.pick_winner()))

…def get_nodes(self):
……for node in self.nodes.copy():
……….nodes = self.get_node_attr(node, 'nodes')
……….for new_node in nodes:
……….cur_blockchain.add_another_node(new_node)
```

```
...def resolve_conflict(self):
……for node in self.nodes:
……blockChain_length = self.get_node_attr(node, 'blockChain_length')
……if blockChain_length >= len(self.blockChain):
……node_blockChain = self.get_node_attr(node, 'blockChain')
……if self.is_chain_valid(node_blockChain):
……if (blockChain_length == len(self.blockChain))\
……and (self.blockChain != node_blockChain)\
……and (self.myAccount['Weight'] > self.get_node_attr(node, 'myAccount')['Weight']):
……return
……self.blockChain = node_blockChain
……print('Chain is replaced!!!')
……return
……print('My chain is authoritative')

...def get_node_attr(self, node, attr):
……try:
……params = {'port': self.myAccount['Address'].split(':')[1]}
……resp = requests.get(f'http://{node}/getattr/{attr}', params=params).json()
……return resp[attr]
……except:
……if node in self.nodes:
……self.nodes.remove(node)
……return {}

...def is_chain_valid(self, chain):
……_prevBlock = ''
……for block in chain:
……if self.is_block_valid(block, prevBlock=_prevBlock):
……_prevBlock = block
……else:
……return False
……return True

...def add_another_node(self, node):
……if node != self.myAccount['Address'] and node not in self.nodes:
……self.nodes.append(node)

...@staticmethod
...def hasher(data):
……data_string = json.dumps(data, sort_keys=True).encode()
……return sha256(data_string).hexdigest()

...@staticmethod
...def get_validator(address):
……return ', '.join([address['Address'], str(address['Weight']), str(address['Age'])])

...def generate_genesis_block(self, genesisblock):
……address = {'Address': 'genesis', 'Weight': 50, 'Age': 0}
……address = self.get_validator(address)
……genesisblock['Index'] = 0 if not genesisblock['Index'] else genesisblock['Index']
```

```
……genesisblock['Timestamp'] = str(
……datetime.now()) if not genesisblock['Timestamp'] else genesisblock['Timestamp']
……genesisblock['Data'] = {
……} if not genesisblock['Data'] else genesisblock['Data']
……genesisblock['PrevHash'] = '0000000000000000'
……genesisblock['Validator'] = address if not genesisblock['Validator'] else genesisblock['Validator']
……genesisblock['Hash'] = self.hasher(genesisblock)
……return genesisblock


@app.route('/getattr/<attr>')
def get_blockchain_attr(attr):
...port = request.args.get('port', False)
...if port:
……node = f'{request.remote_addr}:{port}'
……cur_blockchain.add_another_node(node)
...obj = getattr(cur_blockchain, attr, {})
...return jsonify({attr: obj})


def load_config(n):
...return json.load(open(f'./config{n}.json', 'r'))


def main():
...global cur_blockchain

...config_n = 0
...if len(sys.argv) > 1:
……config_n = sys.argv[1]
...cur_config = load_config(config_n)

...account = {'Address': f'127.0.0.1:{cur_config["port"]}', 'Weight': 2 if config_n == 0 else 1}
...cur_blockchain = Blockchain(GENESIS_BLOCK.copy(), account)
...if config_n != 0:
……time.sleep(randint(5, 20))
……cur_blockchain.add_another_node(f'127.0.0.1:2108')

...threading.Thread(target=lambda: app.run(
……host='127.0.0.1', port=cur_config["port"], debug=False, use_reloader=False)).start()

...while True:
……cur_blockchain.pos()
……print('files:')
……files = cur_blockchain.files()
……for file in files:
……print(f'\t{files[file]["hash"]} | {files[file]["last_update"]} | {file} | {files[file]["data"]}')
……if cur_blockchain.blockChain_length == 10:
……break
```