

ЗМІСТ

Перелік познач та скорочень.....	4
Вступ.....	5
1 Публічні дані, методи їх збереження та перевірки на достовірність	6
1.1 Аналіз використання фальсифікованих документів, причини та наслідки їх використання	6
1.2 Використання технології блокчейн для уникнення можливості фальсифікації даних	8
1.3 Порівняльний аналіз існуючих аналогів.....	10
1.3.1 Tokenізація з використанням Ethereum.....	10
1.3.2 Tokenізація з використанням Bitcoin	12
1.3.3 Порівняння сучасних систем tokenізацій.....	13
1.4 Постановка задачі дослідження.....	14
2 Технології та методи що використовуються при розробці систем tokenізації на основі блокчейн	16
2.1 Технології що використовуються в блокчейн.....	16
2.1.1 Хеш-функції.....	16
2.1.2 Однорангові мережі	25
2.1.3 Технологія блокчейн.....	28
2.2 Моделі бізнес процесів	32
2.3 Вимоги до програмної системи	33
2.3.1 Функціональні вимоги	34
2.3.2 Нефункціональні вимоги	34
3 Архітектура і проектування системи	35
3.1 Побудова загальної архітектури програмної системи.....	35
3.1.1 Бізнес процеси	35
3.2 Моделі даних блокчейну	40
3.3 Інструменти для реалізації	41
4 Розробка та тестування програмної системи	42
4.1 Особливості програмної реалізації системи.....	42

4.1.1 Розробка API для роботи з базою даних.....	42
4.1.2 Розробка P2P API	43
4.1.3 Розробка системи управління блокчейном.....	44
4.1.4 Розробка інтерфейсу	45
4.2 Процес тестування	48
4.2.1 Модульне тестування.....	49
4.2.2 Функціональне тестування.....	50
4.2.3 Нефункціональне тестування.....	50
4.3 Приклади використання	52
4.4 Можливі вдосконалення системи	53
Висновки	54
Список джерел інформації	55

ПЕРЕЛІК ПОЗНАК ТА СКОРОЧЕНЬ

БД – база даних.

БЧ – (англ. Blockchain) – ланцюг блоків з даними.

ДП – Дипломний проект.

ПЗ – Програмне забезпечення.

ПС – програмна система.

ПТУ - професійно-технічне училище.

IDE (англ. Integrated Development Environment) – інтегроване середовище розробки.

P2P – (англ. Peer-to-peer) рівноправна мережа.

ВСТУП

Протягом сучасної історії людства завжди гостро стояло питання коректності даних, їх валідності та можливості їх фальсифікації. В епоху комп'ютерних технологій, почали з'являтися безліч способів довести оригінальність даних, але більшість способів мають одну спільну проблему – вони можуть бути змінені зсередини системи, а отже в теорії дані можна з фальсифікувати, якщо мати прямий доступ до бази даних.

Актуальність роботи полягає у вирішенні вище описаних проблем та недоліків сучасних систем збереження інформації. В майбутньому це допоможе зменшити рівень корупції та зробить неможливим використання фальшивих документів.

Для реалізації поставленої цілі найкращим рішенням буде використання досить нової технології збереження даних блокчейн, що завдяки своїй основній характеристиці, децентралізованості, унеможливорює внесення змін до вже внесених даних. А завдяки технології електронного підпису, робить майже неможливим видачу фальсифікованих документів.

Кінцева мета дипломної роботи є розробка програмних компонентів системи, що надасть можливість будь якій особі чи структурі збереження публічної інформації, і також надасть можливість будь-кому цю інформацію перевірити на достовірність тим самим вирішити сучасну проблему фальсифікації документів.

Об'єктом дослідження є проектування та розробка програмних компонентів для системи токенизації із застосуванням технологій блокчейн.

Предмет дослідження: токенизація даних в децентралізованій базі даних блокчейн.

В дипломній роботі вирішуються наступні завдання: аналіз переваг та недоліків технології блокчейн; визначення загальної структури системи блокчейн; основні особливості блокчейн; проектування архітектури, розробка та інтеграція програмних компонентів системи токенизації.

1 ПУБЛІЧНІ ДАНІ, МЕТОДИ ЇХ ЗБЕРЕЖЕННЯ ТА ПЕРЕВІРКИ НА ДОСТОВІРНІСТЬ

1.1 Аналіз використання фальсифікованих документів, причини та наслідки їх використання

Підроблення документів є доволі поширеним суспільно небезпечним діянням. Так, згідно зі статистичними даними Генеральної прокуратури України у 2014 році було зареєстровано 14830 випадків підроблення документів, печаток, штампів і бланків, у 2015 році правопорушень було 14003, у 2016 році – 13958, а у 2017 - 14105 [1].

Розглянувши гістограму та її лінію тренда, що зображений на рисунку 1.1, можна зробити висновки про те, що кількість випадків фальсифікації документів кожного року майже не змінюється.

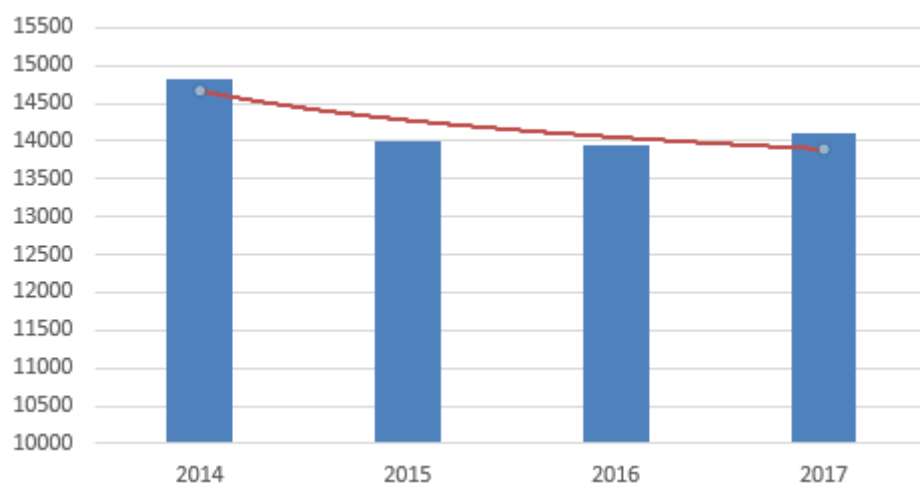


Рисунок 1.1 – Випадки підроблених документів

Аналіз експертної практики свідчить про збільшення фактів виявлення підроблених документів що регулюють земельні відносини. Згідно зі статистикою лише Вінницького науково-дослідного експертно-криміналістичного центру МВС України на експертне дослідження надходять документи, що надають право користуватись та розпоряджатись землею.

Так, наприклад, у 2012 році судовими експертами досліджувались 28 таких документів, у 2014 році – 48, у 2015 – 76, 2016 – 85, у 2017 – 112, у 2018 – 120 [2]. Збільшення надходження матеріалів, об'єктами яких є документи, що регулюють земельні відносини, також підтверджується і загальною державною статистикою Державного експертного центру МВС України [3].

Розглянувши гістограму та її лінію тренда, що зображений на рисунку 1.2, можна зробити висновки про те, що кількість випадків фальсифікації документів, що стосуються надання права користуватися та розпоряджатися землею, кожного року зростає.

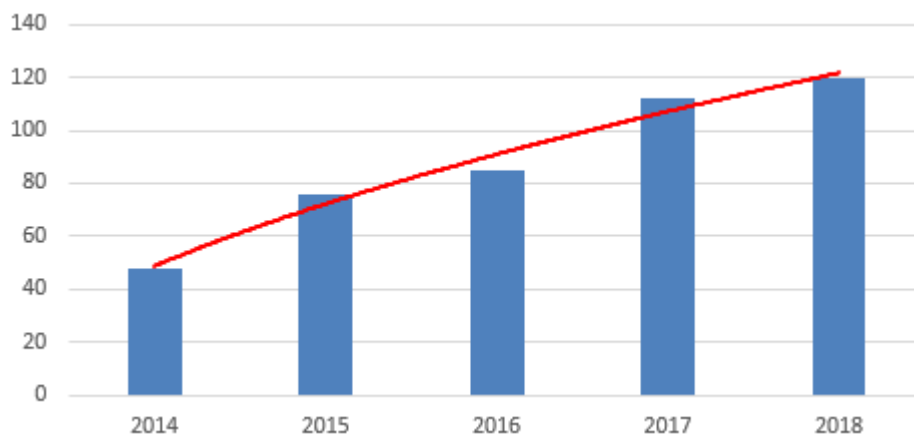


Рисунок 1.2 – Кількість судових справ Вінницького науково-дослідного експертно-криміналістичного центру, що розглядали підробку документів, що до землеволодіння

Доволі поширеним явищем є також підроблення документів, що посвідчують вищу освіту. Про це свідчить доступність купівлі підроблених документів через Інтернет. Ціни на різних сайтах у середньому однакові: диплом професійно-технічного училища (ПТУ) вартує від 350 дол. США, Звичайний диплом бакалавра чи магістра обійдеться приблизно у 800 дол. США [4]. На жаль, офіційної статистики щодо виявлених підроблених документів про вищу освіту немає, проте різноманіття пропозицій на інтернет ресурсах свідчить про велику популярність даних послуг.

Підробка документів є не лише самостійним злочином, також це призводить до більш важких правопорушень: службова некомпетентність, зловживання службовим станом, шахрайство.

Як можна зрозуміти, опираючись на статистику, основною проблемою є саме легкість у створенні підробок та великі складнощі, з боку правоохоронних органів, що до розпізнання фальсифікованих документів, з подальшим затриманням фабриканти.

Таким чином використання підроблених документів залишається актуальним та проблемним для правоохоронних та судових органів, а також громадян.

1.2 Використання технології блокчейн для уникнення можливості фальсифікації даних

Блокчейн - це спільно використовуваний, постійний реєстр, який спрощує процес запису транзакцій та обліку активів у мережі. Актив може бути матеріальним (будинок, автомобіль, гроші, земля) чи нематеріальним (інтелектуальна власність, патенти, авторські права, брендинг).

Існує кілька підходів до створення блокчейн-мережі. Це може бути загальнодоступна або приватна мережа, блокчейн ексклюзивний або блокчейн-консорціум [5].

До загальнодоступної блокчейн-мережі (наприклад, Bitcoin) може приєднатися будь-який користувач. До недоліків такої мережі належать високі вимоги до обчислювальної потужності, низький рівень конфіденційності транзакцій та слабкий захист. Це критерії важливі при використанні блокчейну у корпоративних середовищах.

Приватна блокчейн-мережа, як і загальнодоступна блокчейн-мережа, є децентралізованою одноранговою мережею. Проте управління такою мережею здійснюється однією організацією, яка відповідає за управління учасниками, виконання протоколу консенсусу та підтримку загального реєстру. Залежно

від сценарію використання такий підхід дозволяє істотно підвищити достовірність і надійність інформації, що передається між учасниками.

Приватні та загальнодоступні мережі можуть бути ексклюзивними. Це накладає певні обмеження на коло осіб, яким дозволено брати участь у мережі або лише окремих транзакціях. Учасникам необхідно отримати запрошення чи дозвіл на приєднання.

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

В державному секторі блокчейн вирішує одразу декілька проблем: надлишкову бюрократію, корупцію та фальсифікацію при голосуванні. Усі документи не потрібно зберігати в паперовому вигляді в різного роду архівах, вони надійно збережені у електронному варіанті, а правдивість підтверджується електронним підписом. Також завдяки унікальності електронного підпису кожного окремого користувача, неможливим стає фальсифікація будь якого голосування, адже він засвідчує конкретну особу, що знаходиться в системі і не дозволить голосувати повторно.

Також блокчейн можна використовувати в охоронній сфері. Через все доступність даних можна з легкістю визначити підозрілі перекази коштів, не правомірно оформлені угоди, сфальсифіковані документи та посвідчення. Це полегшить роботу правоохоронних органів та сильно покращить ефективність їх роботи автоматизувавши пошук підозрілих даних.

Бізнес залежить від даних. Швидкість отримання та точність даних грають вирішальну роль. Блокчейн ідеально підходить для надання такої інформації, оскільки він пропонує уповноваженим учасникам мережі загальний та повністю прозорий доступ до інформації у незмінному реєстрі. Мережа блокчейна дозволяє відстежувати замовлення, платежі, облікові записи та багато іншого. І оскільки всі учасники мають спільний доступ до

єдиного джерела достовірних даних, ви можете в будь-який момент переглянути всі відомості про транзакції.

1.3 Порівняльний аналіз існуючих аналогів

На сьогодні існує безліч різних систем токенизацій, що використовують блокчейн. Найпопулярніші з них Ethereum та Bitcoin. Хоч основною функцією цих систем є обмін активами, в них також представлені і інші можливості використання, що реалізують потенціал технології блокчейн, наприклад таке явище як смарт-контракт.

Нижче буде розглянуто та порівняно дані програмні рішення.

1.3.1 Токенизація з використанням Ethereum

Ethereum — криптовалюта та платформа для створення децентралізованих онлайн-сервісів на базі блокчейна, що працюють з урахуванням смарт-контрактів. Реалізовано як єдину децентралізовану віртуальну машину.

Смарт-контракт — альтернатива юридичним договорам. У юридичних контрактах третьою стороною є судова система країни, де укладено договір, саме вона відповідає за виконання контракту.

Смарт-контракти — це такий самий договір, тільки цифровий. Він існує всередині системи Ethereum та його виконання гарантується комп'ютерною програмою, а в фундаменті — сувора математична система [6].

Схематичне зображення смарт-контракту показано на рисунку 1.3.

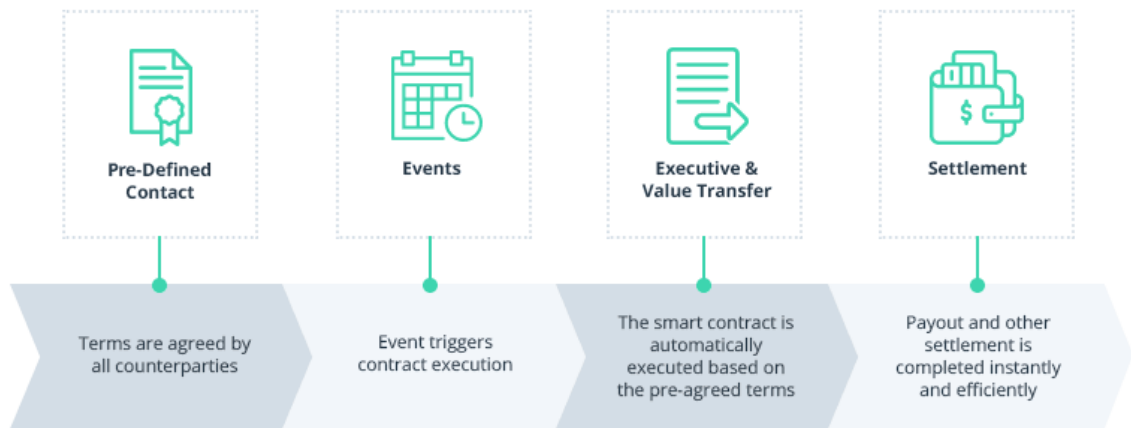


Рисунок 1.3 – Схематичне зображення роботи смарт-контракта

Переваги смарт-контракту [7]:

- Ефективність витрат;
- Швидкість обробки;
- Автономність;
- Надійність.

Для реалізації смарт-контрактів використовують спеціальну мову програмування Solidity (об'єктно-орієнтована, Тьюрінг повна мова програмування смарт-контрактів), програміст просто пише код, що встановлює обмеження та накладає умови для сторін, а в системі блокчейн цей код виконується.

Як приклад, можна привести тривіальну задачу – голосування. Програміст описує функцію голосування та перегляду результатів і додає її до системи блокчейн. Далі користувачі можуть використовувати ці функції для того, щоб проголосувати, чи подивитись результат відповідно. Потрібно зазначити, що як тільки користувач проголосує його унікальний ідентифікатор та прийняте рішення буде збережено в смарт-контракті і повторно голосувати, чи змінити своє рішення він вже не зможе.

Завдяки смарт-контрактам можна вирішувати і простіші операції, такі як звичайна токенизація даних користувачів. Проте слід зазначити, що Ethereum розроблений з ухилом на варіативність в використанні, що значить навіть для

простого запису даних, користувач має написати просту функцію, що буде зберігати дані до транзакції. В цьому і є головна проблема – важкість у використанні. Людина має бути освідченою в роботі системи, а інакше існує головна вразливість сучасних програмних систем, а саме помилка на основі людського фактору.

Також виконання смарт-контрактів вартує віртуальних коштів користувача. Кожен крок в програмі вартує певну одиницю валюти. Зроблено це в першу чергу для того, щоб зробити виконання будь якого коду скінченним. І звісно це дає, як перевагу у вигляді безкінечного варіантів реалізації контрактів, так і недолік у вигляді меншої доступності для користувачів, що просто токенизують дані, тобто вносять їх до блокчейну.

1.3.2 Токенізація з використанням Bitcoin

Bitcoin - пірінгова платіжна система, що використовує однойменну одиницю для обліку операцій. Для забезпечення функціонування та захисту системи використовуються криптографічні методи, але при цьому вся інформація про транзакції між адресами системи доступна у відкритому вигляді [8].

Bitcoin - найперша і найвідоміша криптовалюта. Спочатку блокчейн біткоіна не розроблявся для застосування смарт-контрактів всередині блокчейна біткойна, але в ньому можуть виконуватися смарт-контракти з обмеженим функціоналом.

Bitcoin підтримує такі різновиди смарт-контрактів:

- Смарт-контракт Заповіт - де кошти на гаманці переходять спадкоємцям, якщо гаманець неактивний;
- Смарт-контракт Мультисиг - де потрібно кілька підписів для транзакції;
- Смарт-контракт Ескроу – де потрібна третя сторона для розблокування коштів на смарт-контракті;
- Смарт-контракт Сейф – де кошти заморожуються на гаманці до певної дати.

Bitcoin зосереджується на токенизації активів, що робить його простішим у використанні для операцій з коштами користувачів, проте не надає можливості токенизації іншого роду інформації. Смарт-контракти, що мали б розширити спектр можливостей даного блокчейну, лише більше заточують його використання для транзакцій активів та валюти.

1.3.3 Порівняння сучасних систем токенизацій

Ethereum має набагато більший потенціал для токенизації ніж Bitcoin, проте потребує участь професіональних розробників для створення смарт-контрактів, що ускладнює користування для звичайних користувачів, котрим, частіш за все, потрібно вирішувати досить тривіальні задачі. Натомість Bitcoin надає скромний функціонал, що надає можливість в токенизації лише активів, що робить його більш простим у використанні, проте не реалізує спектр токенизації різного роду документів, посвідчень та договорів. Ще однією перевагою Bitcoin є висока надійність, що підтверджена багаторічним використанням (Bitcoin з'явився у 2009 році і ні разу не давав збоїв). Хоч Ethereum також не давав збоїв, він є досить новим рішенням (Ethereum з'явився у 2015 році), а отже не має такої довгої історії користування як Bitcoin і користується меншою довірою користувачів. Також смарт-контракти, хоч надійні з боку системи блокчейн, можуть мати вразливості, що допущені розробниками.

Важливим критерієм блокчейну є швидкість обробки транзакції. По даному пункту лідером є Ethereum він використовує модифікацію функції хешування SHA3-BLAKE, яка майже не поступається в надійності алгоритму SHA3-KECCAK, що використовує Bitcoin, але працює в тричі швидше. Проте частково така висока швидкість нівелюється тим, що як правило смарт-контракти більш затратні в обчисленні ніж звичайні транзакції.

На таблиці 1.1 показаний результат проведеного аналізу між найпоширенішими блокчейн системами.

Таблиця 1.1 – Порівняльний аналіз Ethereum та Bitcoin

Порівняльна характеристика	Ethereum	Bitcoin
Надійність	Надійний, проте існує можливість похибки зі сторони розробника смарт-контрактів	Надійний
Функціональність	Має майже нескінчену кількість способів використання завдяки смарт контрактам	Має обмежений функціонал, що націлений на керуванні активами
Швидкість	Висока	Середня, але алгоритм хешування є надійнішим ніж у конкурентів

При аналізі найпоширеніших блокчейн систем, можна зробити висновок про те, що ні одна з них не задовольняє повний спектр потреб користувачів в токенизації даних, і кожна має певні недоліки.

Цю нішу можна заповнити, розробивши зручну, швидку та надійну систему для токенизації будь якої інформації користувачів.

1.4 Постановка задачі дослідження

Метою роботи є підвищення безпеки даних шляхом їх токенизації на основі технологій блокчейн.

Для досягнення поставленої мети у дипломній роботі сформовано та вирішено наступні завдання:

- аналіз сучасні аналоги систем токенизації;
- надання порівняльного аналік методів хешування;
- надання порівняльного аналізу мережових рішень децентралізованих систем;
- надання порівняльного аналізу різних підходів технології блокчейн;

- надання обґрунтування обраних технологій;
- проектування та реалізація системи токенизації з використанням технології блокчейн;
- проектування та реалізація програмного забезпечення з імплементацією системи токенизації;
- надання огляду функціональних можливостей розробленої системи;
- тестування системи токенизації;
- надання аналізу виконаної роботи та можливості розвитку розробленої програмної системи.

Таким чином, було сформовано мету даної дипломної роботи, а також визначено завдання, необхідні для досягнення цієї мети.

2 ТЕХНОЛОГІЇ ТА МЕТОДИ ЩО ВИКОРИСТОВУЮТЬСЯ ПРИ РОЗРОБЦІ СИСТЕМ ТОКЕНІЗАЦІЇ НА ОСНОВІ БЛОКЧЕЙН

2.1 Технології що використовуються в блокчейн

Блокчейн системи використовують декілька технологій, що дозволяють досягти найвищого (в порівнянні з централізованими аналогами) рівня захисту. В основному використовуються технологія хешування для електронних-підписів та побудови ланцюжків блоків. А також peer-to-peer мережі, що дозволяють бути системі децентралізованою.

Аналіз методів хешування та однорангових систем (peer-to-peer), а також самої технології блокчейн буде проведено в наступних розділах.

2.1.1 Хеш-функції

Хеш-функції використовуються в блокчейн для генерації унікальних цифрових підписів, та генерації хешу для кожного блоку.

Криптографічна хеш-функція - це математичний алгоритм, який відображає дані довільного розміру бітового масиву фіксованого розміру.

Результат, що виробляється хеш-функцією, називається «хеш-сумою» або просто «хешем», а вхідні дані часто називають «повідомленням».

Для хеш-функції виконуються такі умови:

- хеш-функція є детермінованою, тобто одне те саме повідомлення призводить до одного і того ж хеш-значення;
- значення хеш-функції швидко обчислюється для будь-якого повідомлення;
- майже неможливо знайти повідомлення, яке дає задане хеш-значення;
- майже неможливо знайти два різні повідомлення з однаковим хеш-значенням;
- невелика зміна в повідомленні змінює хеш настільки сильно, що нове та старе значення здаються некорелюючими.

Криптографічна хеш-функція має вміти протистояти всім відомим типам криптоаналітичних атак. У теоретичній криптографії рівень безпеки

хеш-функції визначається з використанням наступних властивостей: Pre-image resistance, Second pre-image resistance, Collision resistance.

Collision resistance відбувається коли різні входні дані виробляють однаковий хеш. Таким чином, хеш-функція вважається стійкою до колізій до того моменту, поки не буде виявлено пару повідомлень, що дає однаковий вихід. Варто зазначити, що колізії завжди існують для будь-якої хеш-функції з тієї причини, що можливі входи нескінченні, а кількість виходів скінченне. Хеш-функція вважається стійкою до колізій, коли ймовірність виявлення колізії настільки мала, що цього знадобляться мільйони років обчислень.

Pre-image Resistance, цю властивість називають опором прообразу. Хеш-функція вважається захищеною від знаходження прообразу, якщо є дуже низька ймовірність того, що зломисник знайде повідомлення, яке згенерувало заданий хеш. Ця властивість є важливою для захисту даних, оскільки хеш повідомлення може довести його справжність без необхідності розкриття інформації.

Second pre-image resistance, цю властивість називають опором другого прообразу. Атака по знаходженню другого прообразу відбувається, коли зломисник знаходить певний вхід, який генерує той самий хеш, що інший вхід, який йому вже відомий. Іншими словами, зломисник, знаючи, що $\text{hash}(m1) = h$, намагається знайти $m2$ таке, що $\text{hash}(m2) = h$ [9].

Існує безліч алгоритмів для створення хешу: MD5 (англ. Message Digest 5), SHA1 (англ. Secure Hash Algorithm), SHA2, SHA3-KECCAK, SHA3-BLAKE та інші.

MD5 – 128-бітний алгоритм хешування, розроблений професором Рональдом Л. Рівестом. Є застарілим та не надійним алгоритмом. В першу чергу алгоритм не відповідає властивості collision resistance, і на сучасних комп'ютерах, методом перебору можна знайти колізії за 1-2 секунди.

SHA1 – просто покращив MD5, збільшивши довжину виведення (160 біт), кількість односпрямованих операцій та складність цих односторонніх операцій.

SHA2 – дуже часто називається сімейством хеш-функцій SHA-2, оскільки містить багато хешів різних розмірів, включаючи 224, 256, 384 та 512-бітові послідовності. Коли хтось каже, що використовує SHA-2, довжина його хеша невідома, але зараз найпопулярнішим є 256-бітний варіант. Хоча деякі математичні характеристики SHA-2 збігаються з SHA-1, і в ньому виявлено незначні недоліки, у криптомир він як і раніше вважається «стійким». Без сумніву, він краще, ніж SHA-1 і будь-який критичний сертифікат, програму або апаратний пристрій, що використовують SHA-1.

SHA3-KECCAK -у 2006 році Національний інститут стандартів та технологій (NIST) запустив конкурс, щоб знайти альтернативу SHA2, яка буде принципово відрізнитися у своїй архітектурі, щоб стати стандартом. Таким чином, SHA3 з'явився як частина великої схеми алгоритмів хешування, відомої як KECCAK. Незважаючи на назву, SHA3 сильно відрізняється своїм внутрішнім механізмом, відомим як «конструкція губки», яка використовує випадкові перестановки для «Всмоктування» та «Витискання» даних, працюючи як джерело випадковості для майбутніх входів, що входять до алгоритму хешування.

SHA3-BLAKE – це альтернативний алгоритм, що зайняв друге місце в конкурсі NIST. Він є компромісом у питанні захищеності та швидкості, тобто все ще надійніше ніж SHA2, але ще й швидше за алгоритм KECCAK. Модифікація SHA3-BLAKE використовується у крипто-валюти Ethereum 2.0 і називається BLAKE2b і в тричі швидше обчислюється на сучасних комп'ютерах.

З огляду найпопулярніших алгоритмів хеш-функцій, кращою є SHA3-KECCAK, тому що є стандартом, а отже її ефективність підтверджена. Ще одним плюсом є розповсюдженість даного алгоритму, що свідчить про довіру

з боку крупних корпорацій до SHA3-КЕССАК. Тому у дипломній роботі буде використовуватися саме цей алгоритм.

Алгоритми КЕССАК - побудовані на основі конструкції криптографічної губки, в якій дані спочатку вбираються в губку, а потім результат Z віджимається з губки.

Будь-яка губчаста функція Кессак використовує одну із семи перестановок КЕССАК - f , яка позначається Кессак - $f[b]$, де $b \in \{25, 50, 100, 200, 400, 800, 1600\}$.

КЕССАК - f перестановки є ітераційними конструкціями, що складаються з послідовності майже однакових раундів. Число раундів n_r залежить від ширини перестановки і задається як $n_r = 12 + 2l$, де $2l = b/25$.

Як стандарт SHA-3 була обрана перестановка КЕССАК - $f[1600]$, для неї кількість раундів $n_r = 24$.

Далі розглядатимемо КЕССАК - $f[1600]$.

Поняття рядок стану є рядком довжини 1600 біт, який ділиться на r і c частини, які називаються швидкістю і ємністю стану відповідно.

Співвідношення поділу залежить від конкретного алгоритму сімейства, наприклад для SHA3-256 $r = 1088$, $c = 512$.

У SHA-3 рядок стану S представлена у вигляді масиву 5×5 слів довжини $w=64$ біт, всього $5 \times 5 \times 64 = 1600$ бит. У КЕССАК також можуть використовуватися слова довжини w , рівні меншим ступеням 2.

Алгоритм отримання хеш-функції можна розділити на кілька етапів:

- За допомогою функції доповнення вихідне повідомлення M доповнюється до рядка P довжини кратної r .
- Рядок P ділиться на n блоків довжини r : P_0, P_0, \dots, P_{n-1} .
- «Всмоктування»: кожен блок P_i доповнюється нулями до рядка довжиною b біт ($b = r + c$) і підсумовується за модулем 2 з рядком стану S , далі результат підсумовування подається у функцію перестановки f і виходить новий рядок стану S , який знову підсумовується за модулем 2 з блоком P_{i-1} і

далі знову подається у функцію перестановки f . Перед початком роботи криптографічної губки всі елементи S дорівнюють нулю.

- «Віджимання»: поки довжина результату Z менша ніж d , де d - кількість біт у вихідному масиві хеш-функції, r перших біт рядка стану S додається до результату Z . Після кожної такої операції до рядка стану застосовується функція перестановок f і дані продовжують «віджиматися» далі, доки не буде досягнуто значення довжини вихідних даних d .

На рисунку 2.1 зображено візуалізацію алгоритму SHA3-KECCAK.

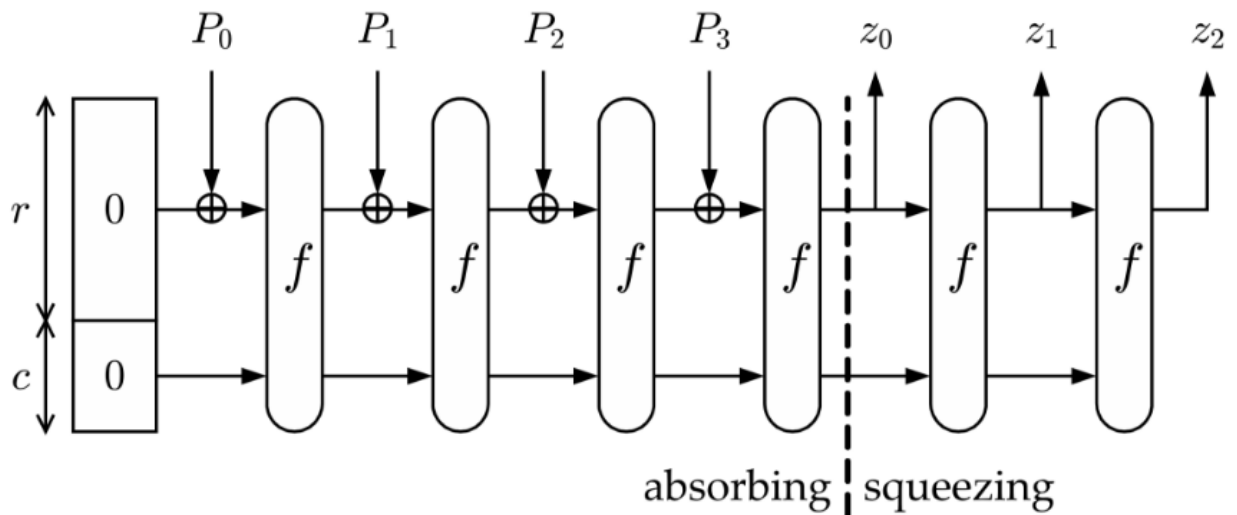


Рисунок 2.1 – Алгоритм KECCAK

Функція доповнення

У SHA-3 використовується наступний шаблон доповнення $10...1$: до повідомлення додається 1, після нього від 0 до $r - 1$ нульових біт і наприкінці додається 1.

$r - 1$ нульовий біт може бути додано, коли останній блок повідомлення має довжину $r - 1$ біт. В цьому випадку останній блок доповнюється одиницею і до нього додається блок, що складається з $r - 1$ нульових біт та одиниці в кінці.

Якщо довжина вихідного повідомлення M ділиться на r , то цьому випадку до повідомлення додається блок, що починається і закінчується одиницями, між якими знаходяться $r - 2$ нульових біт. Це робиться для того, щоб для повідомлення, що закінчується послідовністю біт як функції доповнення, і для повідомлення без цих біт значення хеш-функції були різні.

Перший одиничний біт функції доповнення потрібен, щоб результати хеш-функції від повідомлень, що відрізняються декількома нульовими бітами в кінці, були різні.

Функція перестановок

Базова функція перестановки складається з $12 + 21$ раундів по п'ять кроків:

1. Крок Θ
2. Крок ρ
3. Крок π
4. Крок X
5. Крок i

Далі будемо використовувати такі позначення:

Оскільки стан S має форму масиву $5 \times 5 \times 64$, ми можемо позначити кожен біт стану як $a[x][y][z]$.

Позначимо $A[x][y][z]$ результат перетворення стану функцією перестановки.

Також позначимо $ROT(a, d)$ функцію, яка виконує таку відповідність:

$$ROT(a[x][y][z], d) = a[x][y][z + d \bmod w]$$

$ROT(a, d)$ - звичайна функція трансляції, яка зіставляє біту z біт $z + d \bmod w$.

де w - Довжина слова (64 біт у нашому випадку).

Крок Θ :

Ефект відображення Θ можна описати так: воно додає до кожного біта $a[x][y][z]$ побітову суму двох стовпців $a[x-1][.][z]$ і $a[x+1][.][z-1]$.

Схематичне представлення функції зображено на рисунку 2.2.

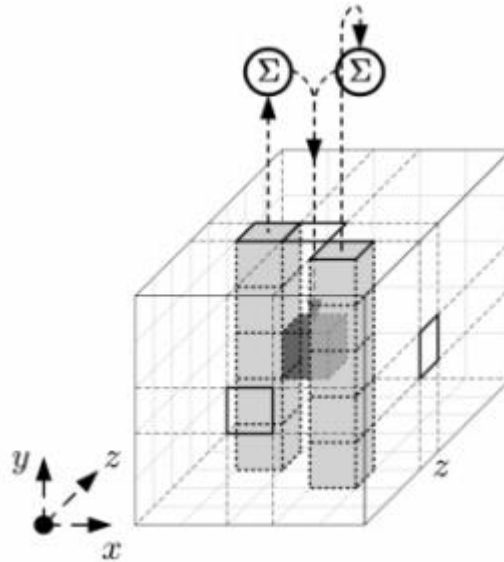


Рисунок 2.2 – Схематичне зображення кроку Θ

Крок ρ :

Відображення ρ спрямоване трансляції всередині слів (вздовж осі z).

Схематичне представлення кроку ρ зображено на рисунку 2.3.

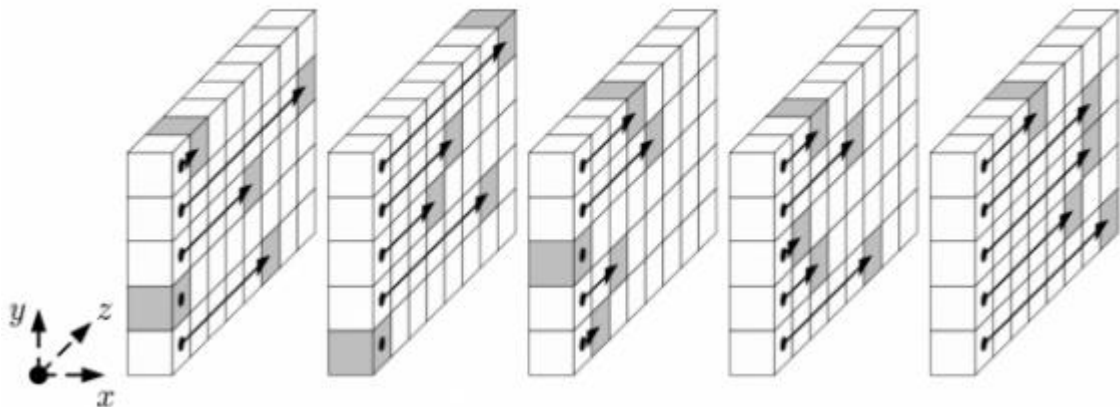


Рисунок 2.3 - Схематичне зображення кроку ρ

Крок π :

Схематичне представлення кроку π зображено на рисунку 2.4.

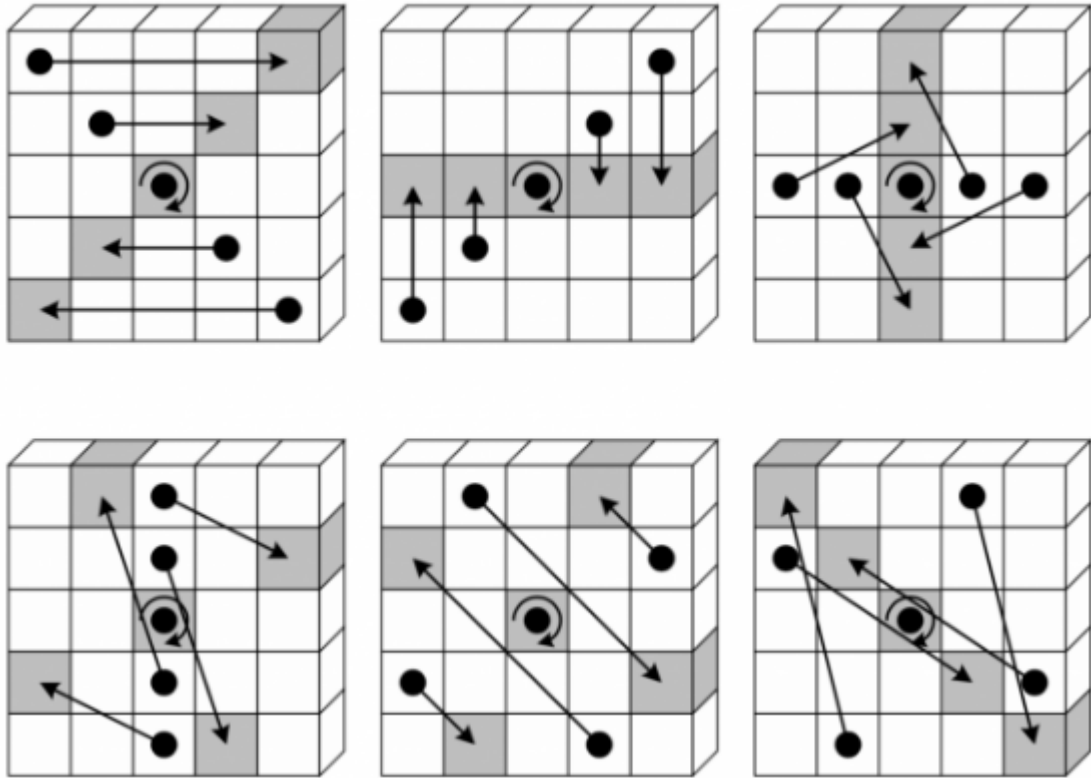


Рисунок 2.4 - Схематичне зображення кроку π

Крок X :

Крок X є єдиним нелінійним перетворенням у Кессак. Схематичне представлення кроку X зображено на рисунку 2.5.

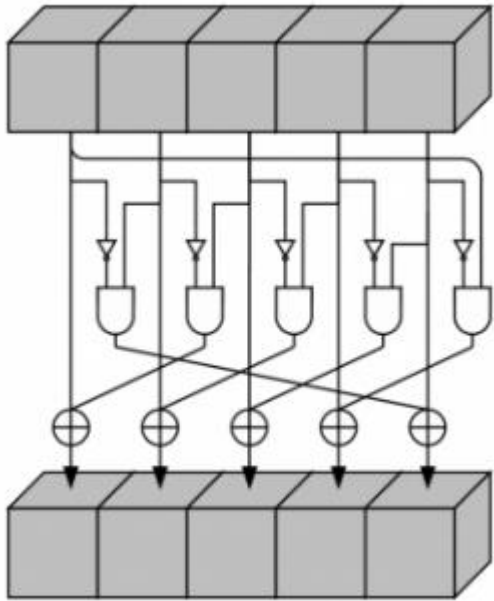


Рисунок 2.5 - Схематичне представлення кроку X

Крок і:

Відображення і складається зі складання з раундовими константами і спрямоване на порушення симетрії. Без нього всі раунди Кессак були б еквівалентними, що робило б його схильним до атак, що використовують симетрію. У міру збільшення і раундові константи додають все більше і більше асиметрії (табл. 1.1).

Таблиця 2.1 – раундові константи, що використовуються на кроці і

RC[0]	0x0000000000000001	RC[12]	0x000000008000808B
RC[1]	0x00000000000008082	RC[13]	0x800000000000008B
RC[2]	0x8000000000000808A	RC[14]	0x80000000000008089
RC[3]	0x8000000080008000	RC[15]	0x80000000000008003
RC[4]	0x0000000000000808B	RC[16]	0x80000000000008002
RC[5]	0x0000000080000001	RC[17]	0x8000000000000080
RC[6]	0x8000000080008081	RC[18]	0x000000000000800A
RC[7]	0x80000000000008009	RC[19]	0x800000008000000A

Усі пройдені кроки можна згрупувати і представити у вигляді псевдокоду як показано на рисунку 2.6.

```

KECCAK- $f[b](A)$ 
  for  $i$  in  $0 \dots n_r - 1$ 
     $A = \text{Round}[b](A, \text{RC}[i])$ 
  return  $A$ 

```

```

Round[ $b$ ]( $A, \text{RC}$ )
   $\theta$  STEP
     $C[x] = A[x, 0] \oplus A[x, 1] \oplus A[x, 2] \oplus A[x, 3] \oplus A[x, 4], \quad \forall x \text{ in } 0 \dots 4$ 
     $D[x] = C[x - 1] \oplus \text{ROT}(C[x + 1], 1), \quad \forall x \text{ in } 0 \dots 4$ 
     $A[x, y] = A[x, y] \oplus D[x], \quad \forall (x, y) \text{ in } (0 \dots 4, 0 \dots 4)$ 

   $\rho$  AND  $\pi$  STEPS
     $B[y, 2x + 3y] = \text{ROT}(A[x, y], r[x, y]), \quad \forall (x, y) \text{ in } (0 \dots 4, 0 \dots 4)$ 

   $\chi$  STEP
     $A[x, y] = B[x, y] \oplus ((\text{NOT } B[x + 1, y]) \text{ AND } B[x + 2, y]), \quad \forall (x, y) \text{ in } (0 \dots 4, 0 \dots 4)$ 

   $\iota$  STEP
     $A[0, 0] = A[0, 0] \oplus \text{RC}$ 

  return  $A$ 

```

Рисунок 2.6 – Псевдокод алгоритму SHA3-KECCAK

2.1.2 Однорангові мережі

У галузі інформаційних технологій, однорангова або пірінгова (P2P від англ. peer-to-peer) мережа складається з групи взаємозалежних пристроїв, які обмінюються між собою файлами і зберігають один і той же набір даних. Кожен учасник (вузол) виступає як індивідуальний пір.

P2P-система підтримується розподіленою мережею користувачів. Зазвичай у них відсутній головний адміністратор або сервер, оскільки кожен вузол містить копію всіх файлів, виступаючи як клієнт і сервер для інших вузлів. Таким чином, кожен вузол може завантажувати файли з інших вузлів, а також у зворотному порядку. Це є відмінністю P2P-мережі в порівнянні з її більш традиційними клієнтами, в яких пристрої завантажують файли з централізованого сервера.

В однорангових мережах підключені пристрої обмінюються файлами, які зберігаються на них самих. Використовуючи програми обміну даними, користувачі можуть завантажувати файли на інші пристрої в мережі. Після того, як користувач завантажив цей файл, він може виступати як його джерело.

На рисунку 2.7 схематично порівнюється клієнт-серверна модель та P2P.

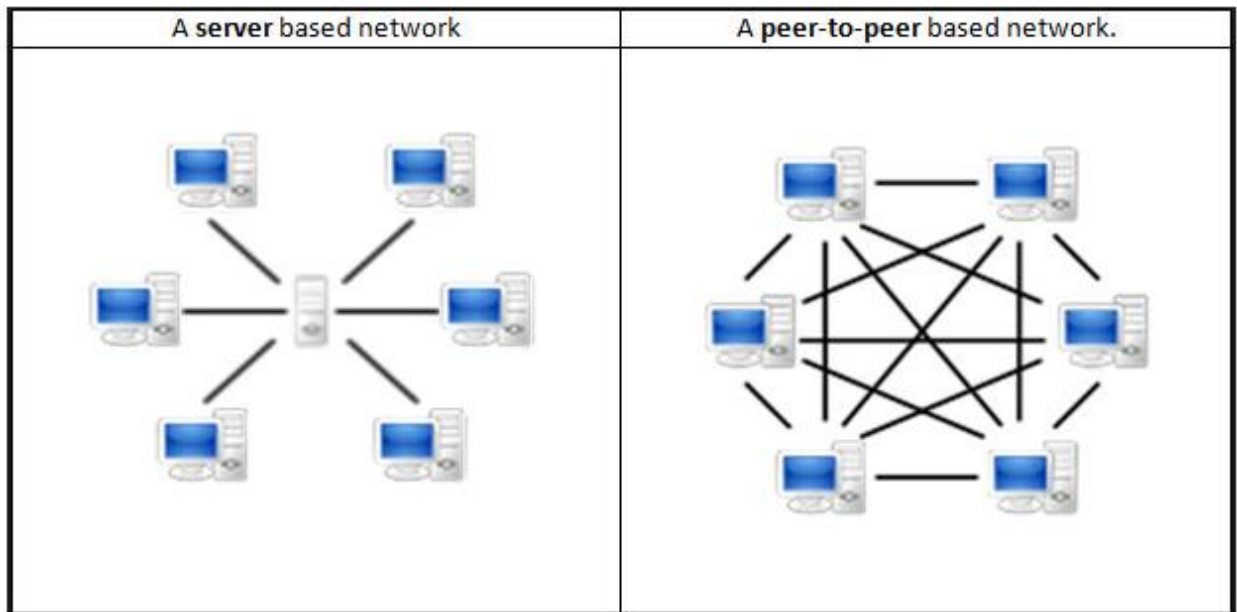


Рисунок 2.7 – Порівняння клієнт-серверна модель та P2P

Однорангові системи класифікуються відповідно до їхньої архітектури. Існує три основні види: неструктурована, структурована та гібридна P2P-мережа[10].

Неструктуровані мережі не представляють будь-якої конкретної організації вузлів. Усі учасники випадково контактують один з одним. І у зв'язку з цим, подібні системи вважаються стійкими до високої активності плинності вузлів (тобто одні вузли приєднуються до мережі, тоді як інші її покидають). Незважаючи на простоту побудови, неструктуровані P2P-мережі можуть вимагати більш високого завантаження центрального процесора та оперативної пам'яті, оскільки пошукові запити відправляються максимально можливій кількості ланок системи.

Порівняно з неструктурованими, структуровані мережі є організованою архітектурою, що дозволяє вузлам ефективніше здійснювати пошук файлів, навіть якщо контент не є широко доступним. У більшості випадків це досягається за рахунок використання хеш-функцій, які полегшують пошук бази даних. У той час як структуровані мережі мають високу працездатність і продуктивність, вони як правило більш централізовані і вимогливі з точки зору встановлення та обслуговування. Крім того, така архітектура менш стійка, коли справа стосується високого рівня плинності вузлів.

Гібридні P2P-мережі поєднують традиційну модель з деякими аспектами однорангової архітектури. Наприклад, завдяки цьому можна створити центральний сервер, який спростить з'єднання між вузлами. У порівнянні з двома іншими видами, гібридні моделі, як правило, демонструють більш високу загальну продуктивність. Вони зазвичай поєднують у собі деякі з основних переваг кожного з підходів і завдяки цьому досягають високого рівня ефективності та децентралізації одночасно.

В сучасних реалізаціях блокчейн в основному використовується неструктурована мережа.

Однорангова неструктурована архітектура мережі блокчейнів надає користувачам безліч переваг. Одним з найважливіших є те, що такі мережі забезпечують велику безпеку, на відміну від традиційного пристрою клієнт-сервер. Розподіл даних у блокчейні серед великої кількості вузлів робить мережу практично несприйнятливою до атак типу «відмова в обслуговуванні» (DoS), яких страждає більшість систем.

Оскільки більшість вузлів повинні досягати консенсусу, перш ніж дані будуть додані до блокчейну, зловмиснику практично нереально внести будь-які зміни. І це насправді так, якщо йдеться про велику мережу, таку як Bitcoin. Невеликі блокчейни більш сприйнятливі до атак, тому що суб'єкт чи група осіб можуть зрештою отримати контроль над більшістю вузлів (що також відомо, як атака 51%).

2.1.3 Технологія блокчейн

Суть технології блокчейн необхідно розкривати через призму такого поняття, як реєстр. Реєстр є формою систематизації та обліку будь-якої інформації. Так, реєстр у його первісному вигляді був покладений в основу комерційної діяльності ще в давні часи і використовувався для фіксування та зберігання різної інформації, в основному про гроші чи майно.

На даний момент алгоритми уможливили створення цифрових розподілених реєстрів, які мають властивості та можливості, що виходять далеко за межі традиційних паперових та електронних реєстрів.

Розподілений реєстр є базою даних, яка розподілена між декількома мережевими вузлами, кожен з яких отримує дані з інших вузлів і зберігає повну копію реєстру. При цьому такі вузли оновлюються незалежно один від одного. Ключовою особливістю розподіленого реєстру є децентралізація, тобто відсутність єдиного центру зберігання та реєстрації даних. При цьому інформація у всіх вузлах розподіленого реєстру має бути валідною та актуальною, що можливо лише за допомогою досягнення згоди між усіма вузлами такого реєстру. Кожен вузол складає та записує оновлення реєстру незалежно від інших вузлів. Потім вузли голосують за оновлення, щоб переконатися, що більшість вузлів згідно з остаточним варіантом. Досягнення згоди щодо однієї з копій реєстру називається консенсусом, цей процес виконується автоматично за допомогою алгоритму консенсусу. Щойно консенсус досягнуто, розподілений реєстр оновлюється, і остання узгоджена версія реєстру зберігається у кожному вузлі.

Приклад загальної структури розподіленого реєстру відображено на рисунку 2.8.

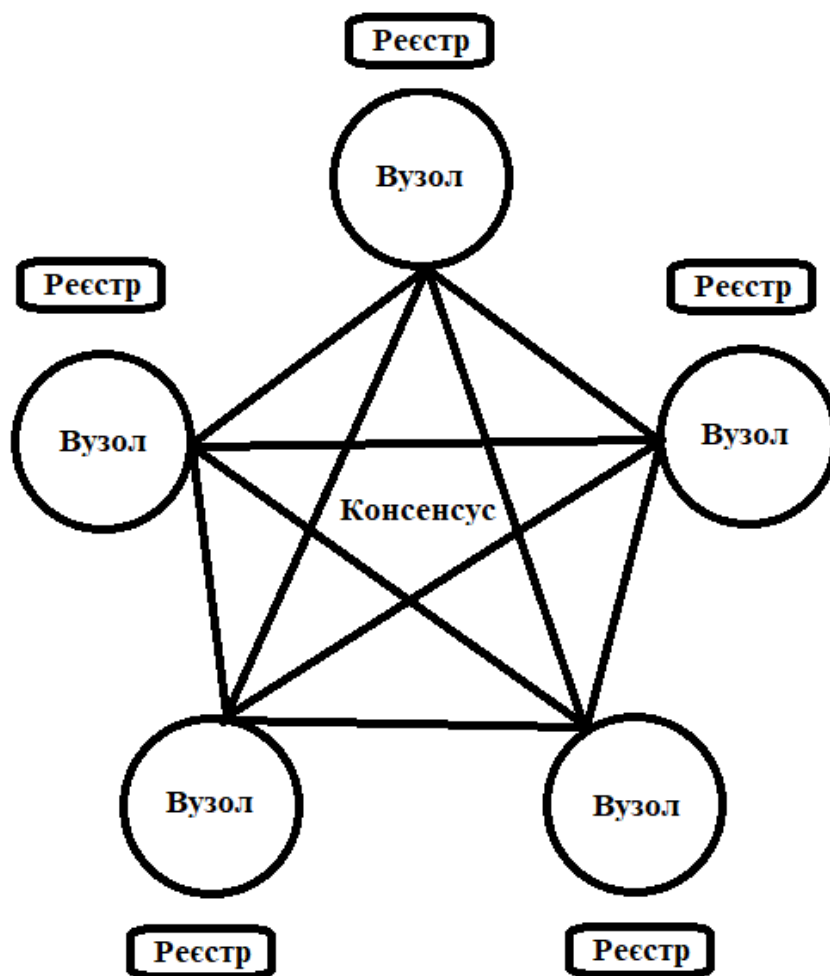


Рисунок 2.8 – Розподілений реєстр

Блокчейн є одним із видів розподіленого реєстру, в якому для досягнення консенсусу між мережними вузлами використовується послідовність блоків. Блоки організовані в хронологічній послідовності, з'єднані один з одним та захищені криптографічними методами, що можна побачити на рисунку 2.9. Кожен такий блок містить хеш-код, обчислений з попереднього блоку, та корисне навантаження. Як корисне навантаження може виступати інформація про транзакції, угоди, укладені договори, внесення до Реєстру даних про фізичну особу, суб'єкт підприємницької діяльності, майно і так далі. За своєю суттю, блокчейн є реєстром записів, що постійно поповнюється, в який можна тільки додавати дані, але при цьому не можна видаляти або змінювати дані, збережені в попередніх блоках [11].

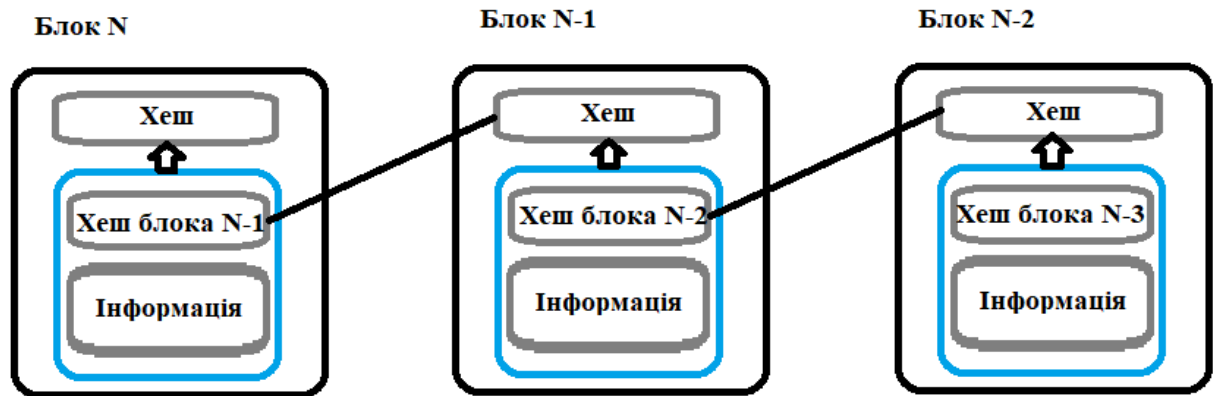


Рисунок 2.9 – Організація блоків в системі блокчейн

Генерація блоку - для створення блоку потрібно знати хеш попереднього блоку, а решту необхідно створювати з наступного змісту (index, hash, data та timestamp). Data - це певна інформація, яка передається кінцевому користувачеві [12].

Зберігання блоків - у пам'яті блоки зберігаються, як однозв'язний список, де вказівником слугує хеш блоку. Кожний блокчейн починається з генезис блоку. Генезис блок – це блок, що генерується першим в списку та має завжди однакові (стартові) дані в собі. В ньому не можуть зберігатися дані користувачів.

Перевірка цілісності блоків - у будь-який момент часу блокчейн повинен бути в змозі перевірити, чи є блок або ланцюжок блоків допустимими з точки зору цілісності. Це особливо актуально, коли користувач отримує нові блоки від інших вузлів і маємо вирішити, приймати їх чи ні.

Вибирається найдовший ланцюжок - авжди має бути лише один явний набір блоків у ланцюжці одночасно. У разі виникнення конфліктів ми вибираємо ланцюг, який має найдовший ряд блоків.

Спілкування з іншими вузлами - важливою функцією вузла є поділ і синхронізація блокчейн з іншими вузлами.

Правила — використовувані для підтримки синхронізації мережі:

- Коли вузол генерує новий блок, він транслює його до мережі;

- Коли вузол підключається до нової одноранговій мережі, він спирається на останній блок;
- Коли вузол виявляє блок, який має індекс більший, ніж поточний відомий блок, він або додає блок у його нинішньому стані у власний ланцюг або підтримує для заповнення блокчейна.

Контроль над вузлом - користувач, певною мірою, повинен мати можливість контролювати вузол. Це робиться шляхом налаштування http-сервера:

- Переглядати список усіх блоків;
- Створювати новий блок із змістом, заданим користувачем;
- Переглядати або додавати однорангових користувачів.

Слід зазначити, що вузол фактично надає два веб-сервери: один для користувача, щоб контролювати вузол (http-сервер) і один для однорангового (peer-to-peer) зв'язку між вузлами. На рисунку 2.10 схематично зображено взаємодії між блокчейном з вузлами та веб-сокетом.

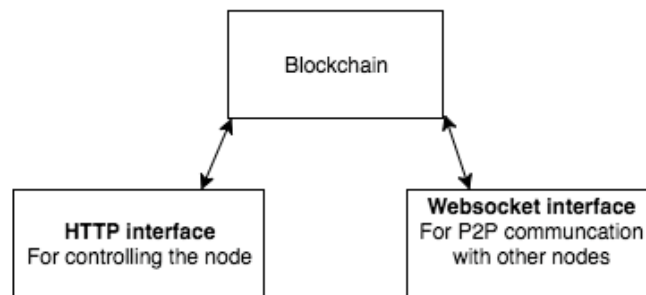


Рисунок 2.10 - Взаємодії між блокчейном з вузлами та веб-сокетом

Таким чином, можна сказати, що основу технології блокчейн становлять кілька основних принципів:

- 1) Блокчейн є розподіленим реєстром і функціонуватиме до останнього активного мережевого вузла;
- 2) У мережі блокчейн відсутня ієрархія, тобто серед усієї множини мережевих вузлів немає головного;

3) За своєю природою блокчейн здатний забезпечити унікальне поєднання відкритості та захищеності даних користувача. Висока міра надійності досягається за рахунок прогресивних методів шифрування;

4) Дані мережі блокчейн неможливо видалити або замінити, оскільки вони підтверджуються безліччю мережевих вузлів;

5) Мережа блокчейн є «довірчою» системою, оскільки транзакції здійснюються безпосередньо між її учасниками, автоматично перевіряються та підтверджуються безліччю вузлів мережі та не вимагають посередників, що повністю виключає недовіру до однієї організації-посередника.

2.2 Моделі бізнес процесів

Моделювання бізнес-процесів (англ. Business process modeling) — формалізований і виконаний за певними правилами опис послідовності дій фахівців або інших зовнішніх систем у формі логічних блок-схем, що визначають вибір подальших дій.

Слід зазначити, що блокчейн є лише децентралізованим сховищем даних, і основна бізнес логіка виконується на підставі вимог користувача. Наприклад лікар може вносити до системи історію хворого і прописувати ліки для свого клієнта. Інтерпретацією цих даних займається стороння система, лікар чи сам хворий, а не система токенизації.

Діаграма бізнес процесів показано на рисунку 2.11.

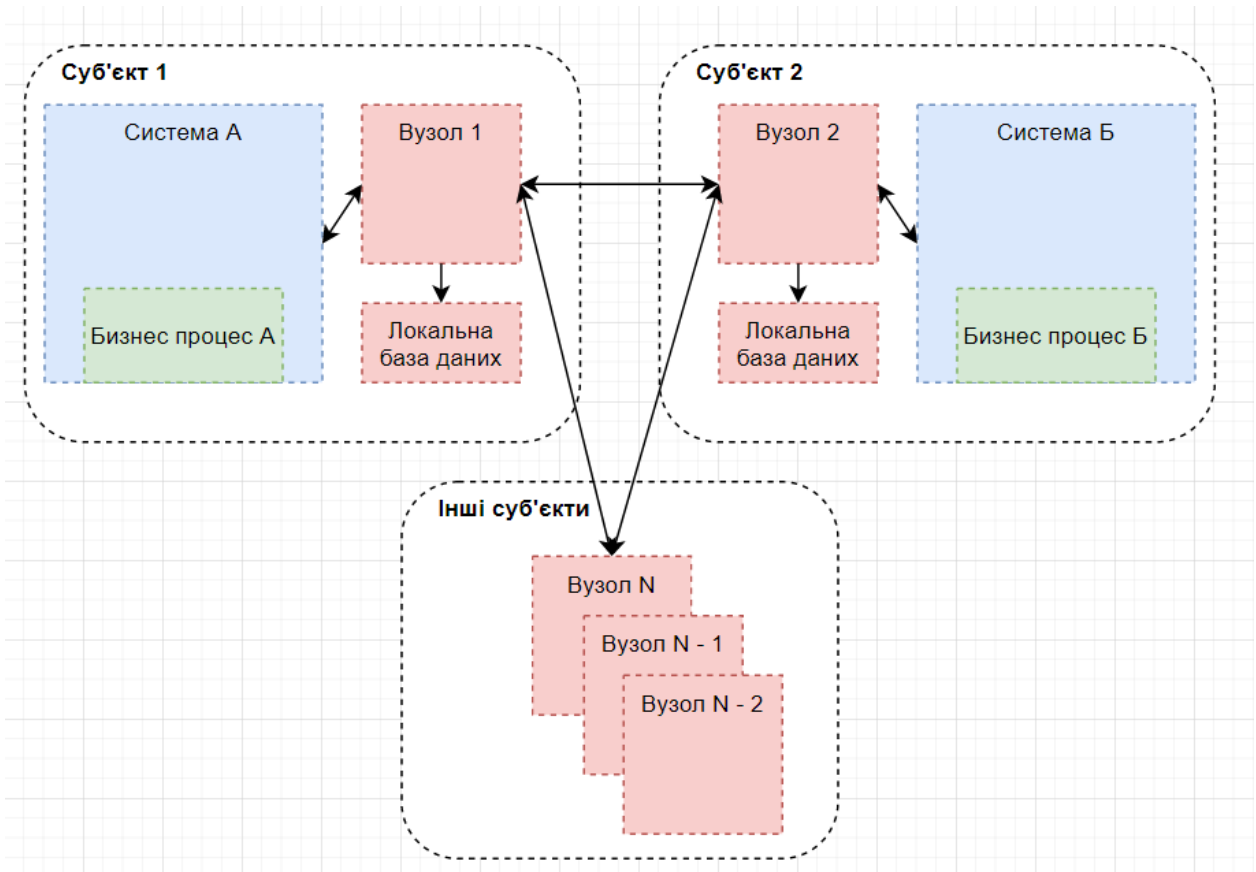


Рисунок 2.11 - Діаграма бізнес процесів

2.3 Вимоги до програмної системи

На рисунку 2.12 показано діаграма вимог до програмного забезпечення.

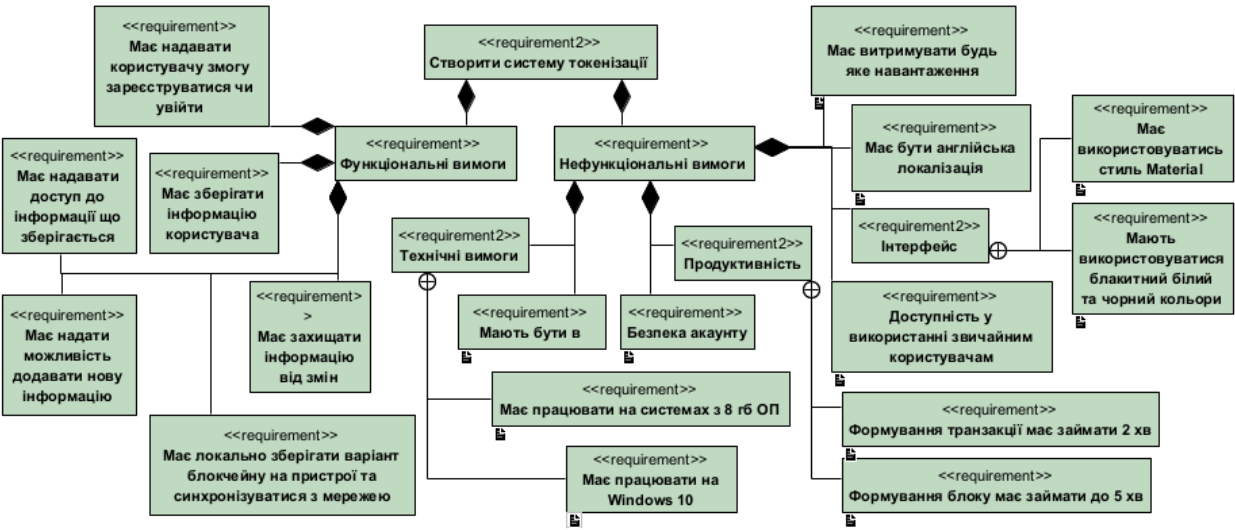


Рисунок 2.12 – Діаграма вимог

2.3.1 Функціональні вимоги

- 1) Система має надавати користувачу змогу зареєструватися чи увійти;
- 2) Система має зберігати інформацію користувачів;
- 3) Система має надати можливість користувачу додавати нову інформацію до блокчейну;
- 4) Система має надавати користувачу повний доступ до інформації, що зберігається в блокчейні;
- 5) Система має захищати інформацію в блокчейні від змін;
- 6) Система має локально зберігати варіант блокчейну на пристрої користувача та синхронізуватися з мережею.

2.3.2 Нефункціональні вимоги

- 1) Продуктивність - формування нового блоку до 30 секунд;
- 2) Доступність у використанні - додаток буде націленим на зручність в користуванні, інтерфейс простий та зрозумілий, для рядового користувача освоїтись повинно займати до 1 робочого дня;
- 3) Надійність – збоїв в роботі додатку можуть виникати лиш у випадку завантаження великої кількості блоків з великим об'ємом даних. Збої у роботі мережі можуть виникати лиш у випадку зникнення з'єднання як такого;
- 4) Безпека – можливість підробити дані має бути вкрай низькою, для цього зломисникам потрібно мати більше 50% апаратної потужності мережі, в іншому випадку зміна має бути неможливою. Приватний ключ має бути унікальним для кожного акаунта;
- 5) Локалізація – система має бути локалізована на англійську мову;
- 6) Технічні вимоги – система має працювати на операційній системі Windows 10. Система повинна працювати з використанням не більш ніж 8 гб оперативної пам'яті;
- 7) Інтерфейс має бути мінімалістичним. Кольори що мають використовуватись: білий, блакитний та чорний. Шрифт у всьому додатку має бути однотипним, а саме: Microsoft Sans Serif.

3 АРХІТЕКТУРА І ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Побудова загальної архітектури програмної системи

3.1.1 Бізнес процеси

Слід зазначити, що блокчейн є лише децентралізованим сховищем даних, і основна бізнес логіка виконується на підставі вимог користувача. Наприклад лікар може вносити до системи історію хворого і прописувати ліки для свого клієнта. Інтерпретацією цих даних займається стороння система, лікар чи сам хворий, а не система токенизації.

Основні БП:

- створення користувача;
- перегляду конкретного токена;
- створення токена;
- перегляду усіх токенів користувача.

Бізнес процеси зображені на рисунках 3.1-3.4 у вигляді діаграм IDEF0 першого рівня деталізації.

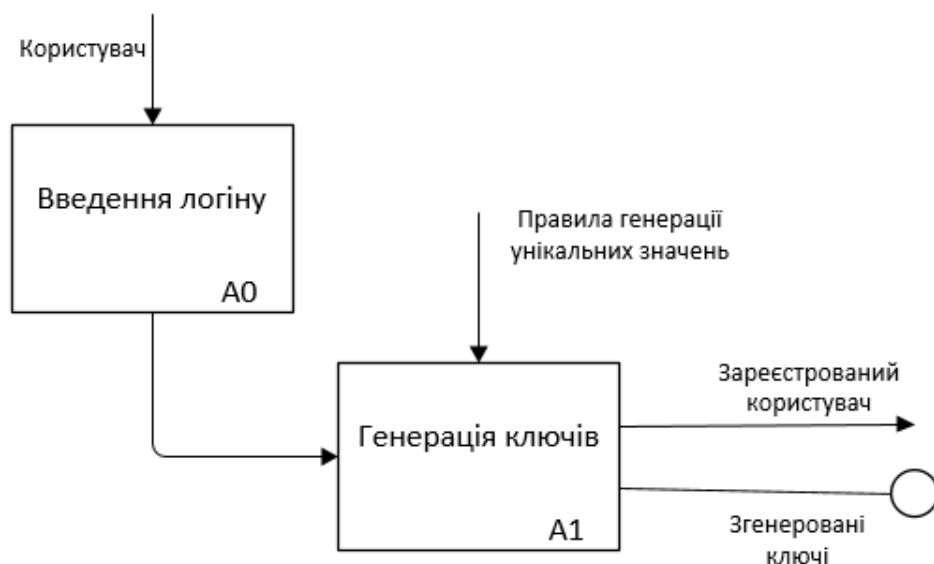


Рисунок 3.1 – Перший рівень деталізації БП реєстрації

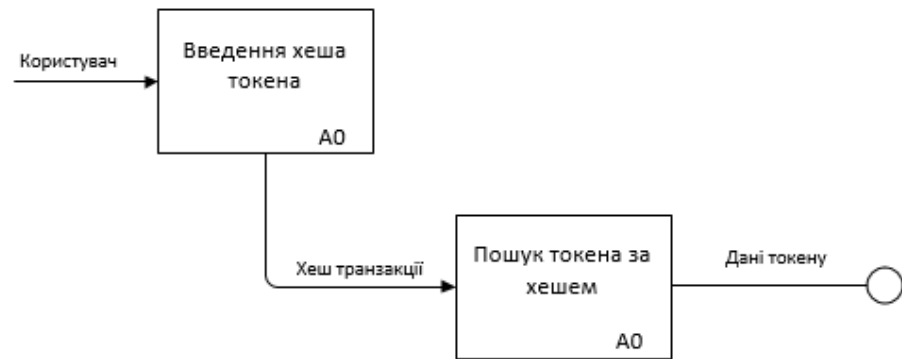


Рисунок 3.2 – Перший рівень деталізації БП пошук конкретного токена

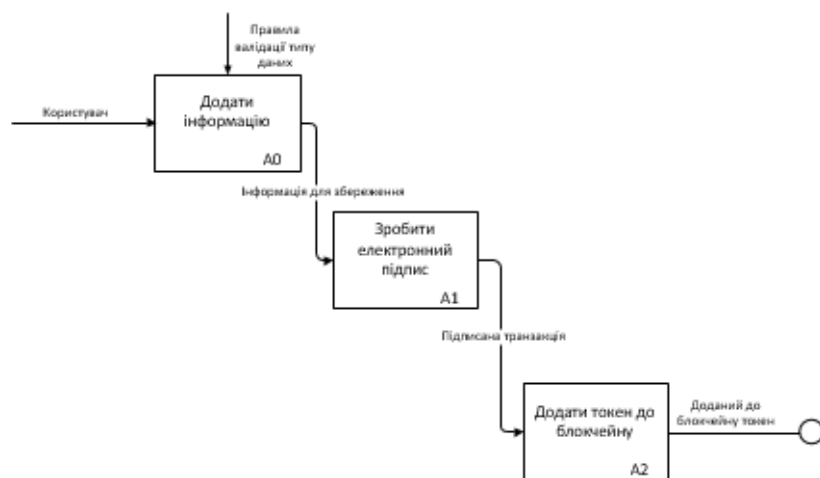


Рисунок 3.3 – Перший рівень деталізації БП створення токена

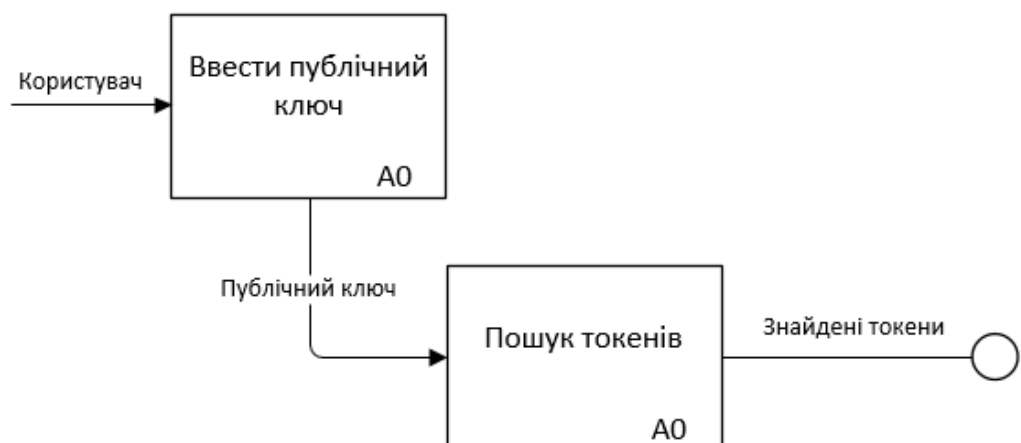


Рисунок 3.4 - Перший рівень деталізації БП пошуку токенів конкретного користувача

3.1.2 Архітектурних рішення

Проаналізувавши вимоги можна зробити певні, фундаментальні рішення, що до загальної архітектури проекту. Патерни, що необхідно буде використовувати та стратегії розробки котрим варто слідувати.

При проектуванні проекту буде використовуватись каскадна стратегія розробки. Дана стратегія буде використана тому, що ми маємо чіткі вимоги до проекту, які вимагають повної реалізації певних функцій, в повній мірі, а значить використання інкрементної чи еволюційної стратегії недоцільне і лиш сповільнить темпи розробки.

Що до патернів, то доцільно використовувати singleton для реалізації взаємодій з блокчейном, тому що це унеможливить ситуації з внутрішнім конфліктом даних в об'єкті, а отже спростить його використання та зменшить кількість перевірок коректності.

При розробці мережевого модуля буде використовуватися однорангова пірингова система, і виконана вона буде з використанням патерна – стратегія. Стратегія дасть гнучкості піринговій системі, а отже можна буде швидко змінювати протоколи передачі даних в залежності від операційної системи в подальшому портуванні проекту.

Для того, щоб можна було передавати дані різного типу буде використано шаблонні методи.

Відповідно до рішень та виявлених бізнес процесів було розроблено загальні архітектурні моделі, що зображені на рисунках 3.5-3.10.

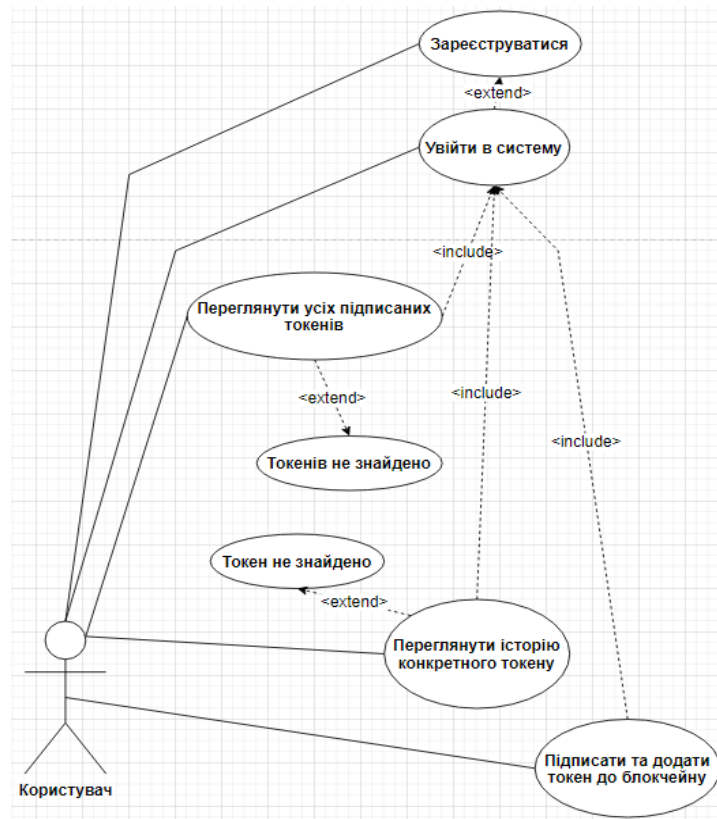


Рисунок 3.5 – Діаграма варіантів використання

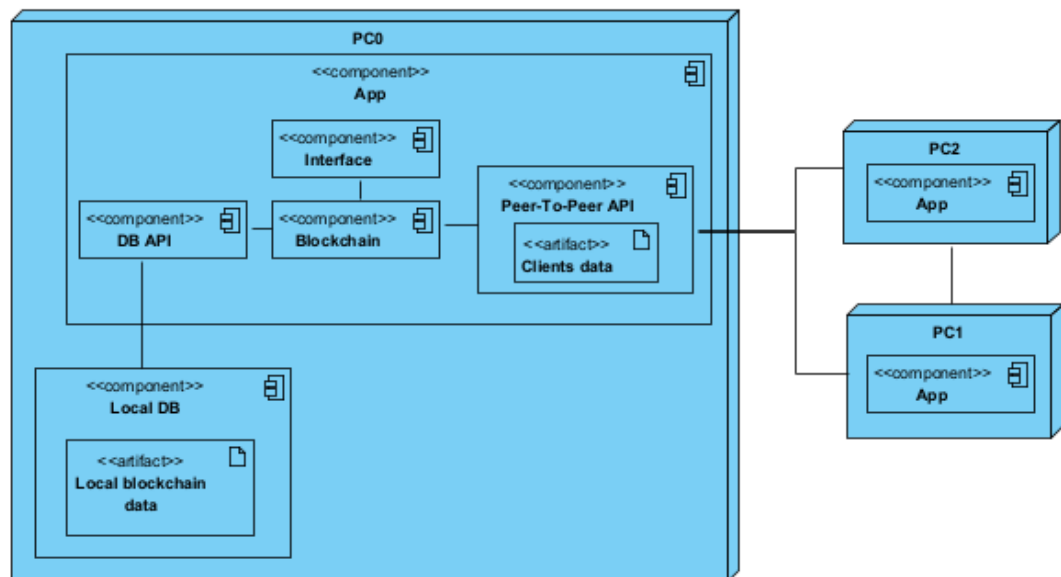


Рисунок 3.6 - Діаграма розгортання

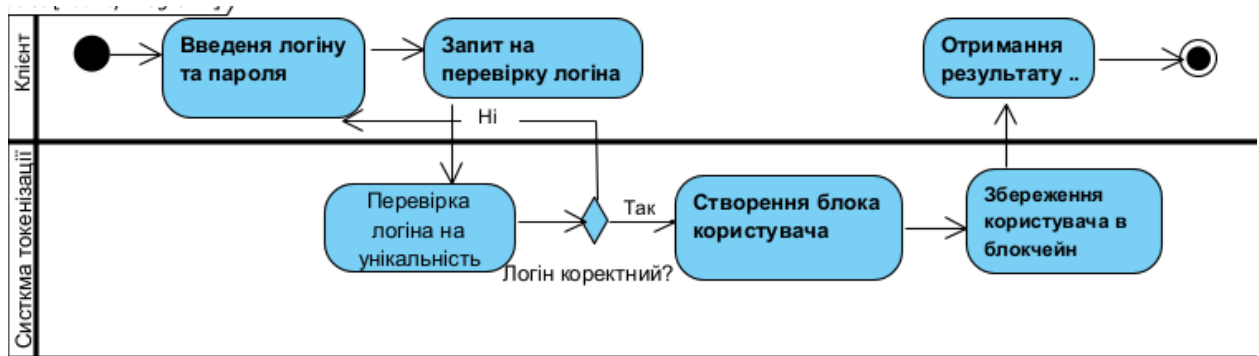


Рисунок 3.7 – Діаграма діяльності процесу реєстрації

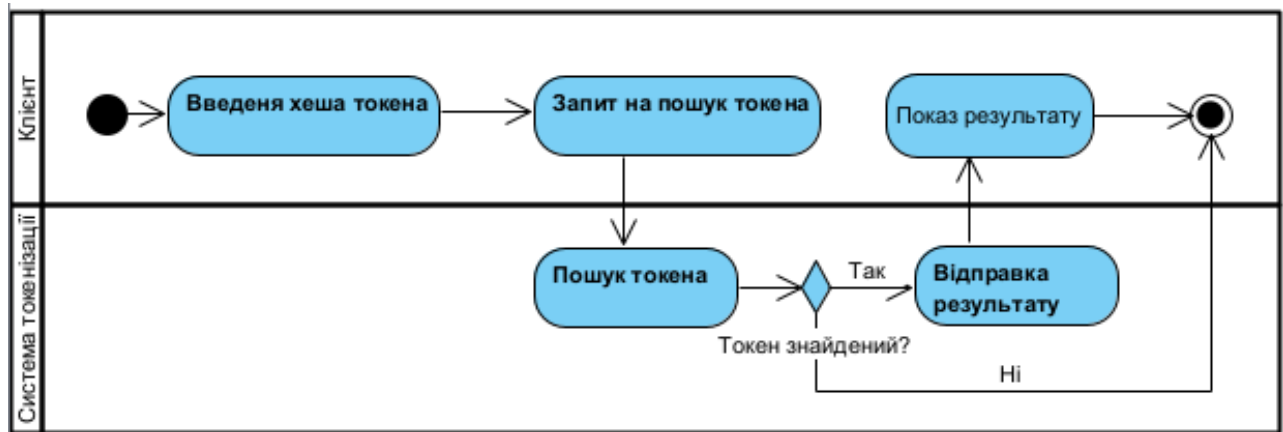


Рисунок 3.8 – Діаграма діяльності процесу пошуку конкретного токена

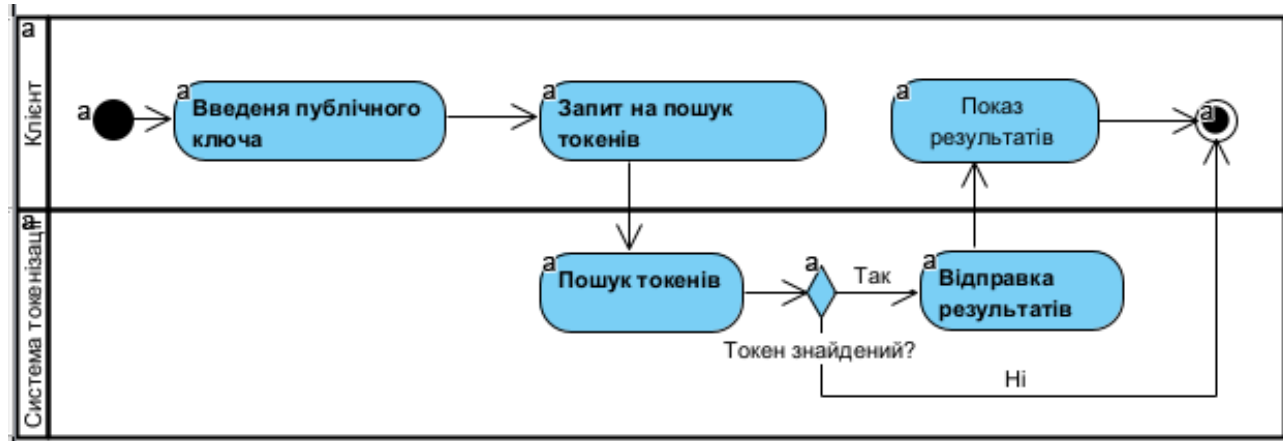


Рисунок 3.9 – Діаграма діяльності процесу пошуку токенів конкретного користувача токена

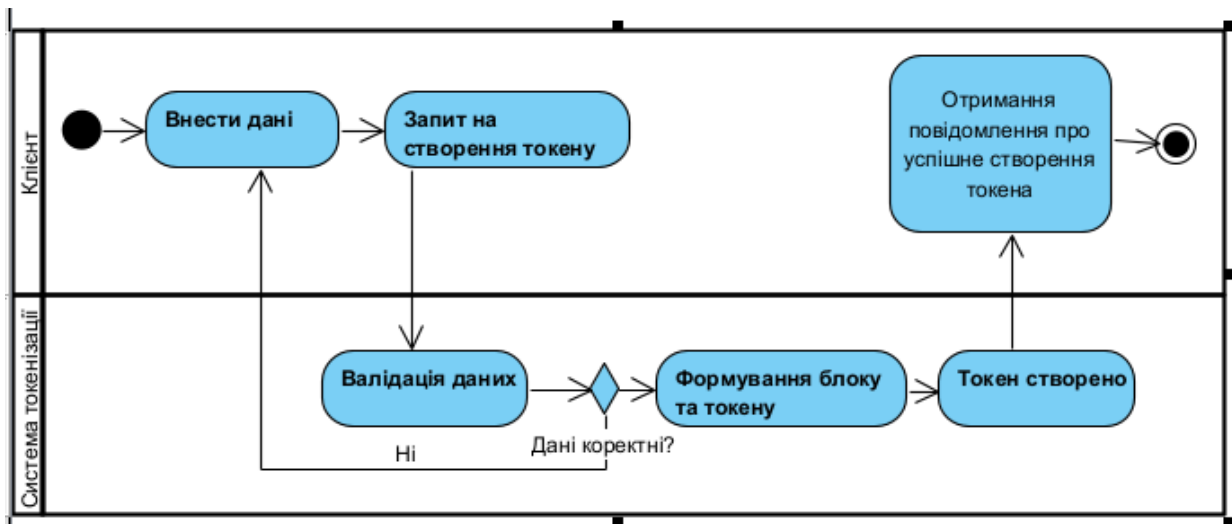


Рисунок 3.10 – Діаграма діяльності процесу додавання нового токена до блокчейну

3.2 Моделі даних блокчейну

Блокчейн складається з блоків, що зберігають різноманітні дані, тому для зберігання блокчейну потрібно лиш найпростіша база даних, що складається з однієї таблиці.

Модель даних, до описує блок показано на рисунку 3.11.

Block	
Owner	varchar(255)
OwnerHash	varchar(255)
Hash	varchar(255)
PreviousHash	varchar(255)
CreatedTime	timestamp
DataType	integer(10)
Data	varbinary(4058)

Рисунок 3.11 – Модель блоку

Сутність «Блок» - має дані про користувача котрий зберіг дані. Дані в бінарному вигляді та їх тип, для подальшої обробки даних А також хеш блоку та хеш минулого блоку, для перевірки блокчейну на предмет коректності ланцюга і для генерації наступних блоків.

3.3 Інструменти для реалізації

Відповідно до поставлених вимог можна обрати інструменти, що будуть використовуватися в ході розробки для вирішення поставлених задач.

Для локального збереження даних буде використовуватися реляційна база даних Sql та Entity Framework, що забезпечить автоматичне створення бази даних на кожному окремому клієнті та надасть зручні інструменти для взаємодії з БД прямо через код програми.

Програмний застосунок буде реалізовано для персональних комп'ютерів на операційній системі Windows, тому що блокчейн може потребувати великої кількості обчислювальної потужності, що персональний комп'ютер може гарантувати.

Для реалізації програмного застосунку та самої системи токенизації буде використовуватися .NET Framework та мова програмування C#, тому що є найпоширенішим та найоптимізованішим рішенням для платформи Windows.

Для реалізації хеш-алгоритмів буде використовуватися SHA3-KECCAK, тому що є най захищеним та досить швидким алгоритмом.

Для реалізації peer-to-peer мережі буде використано модуль System.Net.PeerToPeer, тому що є стандартним і дуже гнучким рішенням бібліотеки .NET Framework.

Інтерфейс буде розроблено з використанням Windows Presentation Foundation (WPF .New Framework) , тому що передбачає зручний, та сучасний функціонал в роботі з UI компонентами, як в окремих графічних редакторах так і прямо з коду програми.

Оскільки у проекті використовується .Net Framework, найкращим рішенням IDE буде Visual Studio. Тому що є безкоштовним та найпопулярнішим рішенням від розробників самого фреймворка, а отже найбільш заточена для роботи з ним.

4 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Особливості програмної реалізації системи

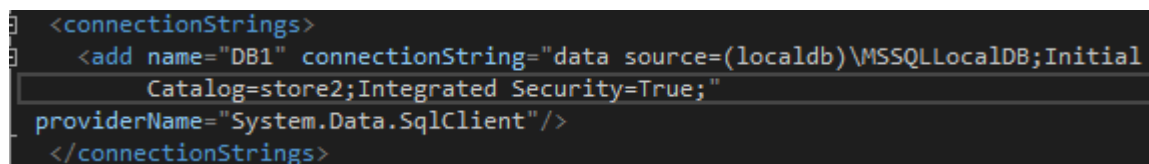
В попередньому розділі було виділено декілька окремих систем, а саме: API що реалізує взаємодію між клієнтами, API що реалізує взаємодію з локальною базою даних, система управління блокчейном, та система для UI відображення роботи блокчейну.

4.1.1 Розробка API для роботи з базою даних

Для простої та швидкої взаємодії з БД використовується Entity Framework. Він дозволяє створювати та взаємодіяти з базою даних не пишучи sql запитів.

Entity Framework — це платформа ORM з відкритим вихідним кодом для додатків .NET, яку підтримує Microsoft. Це дозволяє розробникам працювати з даними, використовуючи об'єкти специфічних для домену класів, не зосереджуючи увагу на базових таблицях і стовпцях бази даних, де ці дані зберігаються. За допомогою Entity Framework розробники можуть працювати на більш високому рівні абстракції, коли вони мають справу з даними, а також можуть створювати та підтримувати орієнтовані на дані програми з меншою кількістю коду в порівнянні з традиційними програмами. [14]

Для створення, або використання бази даних достатньо вказати назву та розташування в App.config файлі, як показано на рисунку 4.1.

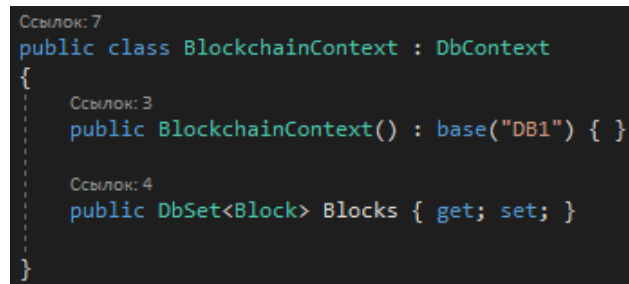


```
<connectionStrings>
  <add name="DB1" connectionString="data source=(localdb)\MSSQLLocalDB;Initial
    Catalog=store2;Integrated Security=True;"
    providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Рисунок 4.1 – Вміст App.config файлу

Для створення таблиці достатньо створити спеціальний клас — «Context», що містить в собі визначення вмісту таблиці. Об'єкт типу Block

позначений атрибутом – `DataContract`, що свідчить про можливість його парсингу в таблицю бази даних. Контекст блоку показано на рисунку 4.2.



```

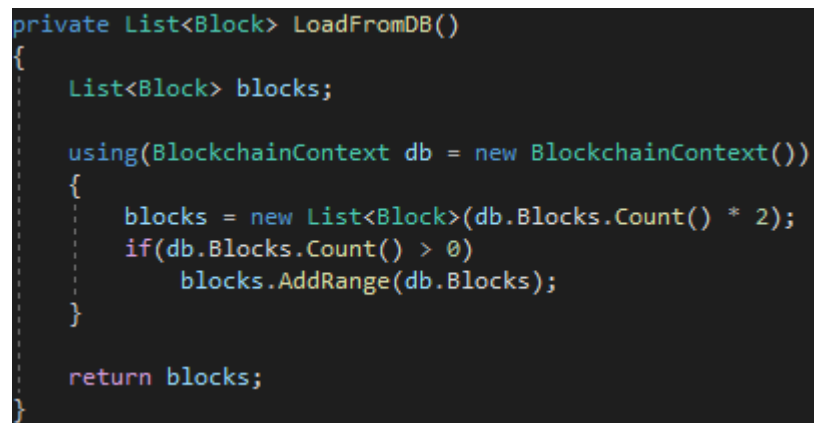
Ссылка: 7
public class BlockchainContext : DbContext
{
    Ссылка: 3
    public BlockchainContext() : base("DB1") { }

    Ссылка: 4
    public DbSet<Block> Blocks { get; set; }
}

```

Рисунок 4.2 – Контекст блоку

Для доступу до даних достатньо створити об'єкт класу контексту та взаємодіяти з ним як зі звичайним списком блоків, як показано на рисунку 4.3.



```

private List<Block> LoadFromDB()
{
    List<Block> blocks;

    using(BlockchainContext db = new BlockchainContext())
    {
        blocks = new List<Block>(db.Blocks.Count() * 2);
        if(db.Blocks.Count() > 0)
            blocks.AddRange(db.Blocks);
    }

    return blocks;
}

```

Рисунок 4.3 – Метод отримання списку блоків з БД

Для внесення змін необхідно напрямку про взаємодіяти з списком об'єктів і після змін, виконати метод - `SaveChanges`.

Розглянуті опції у використанні, повністю задовольняють потреби в локальній базі даних. Розглянувши їх реалізацію за допомогою Entity Framework можна сміливо констатувати, що вибір в фреймворці виправданий і є досить швидким, простим та зручним.

4.1.2 Розробка P2P API

Для побудови P2P з'єднання буде використовуватися бібліотека `System.Net.PeerToPeer`. Ця бібліотека надає можливість для створення

унікальних однорангових клієнтів (клас `PeerName`), реєстрації цих клієнтів в мережі (клас `PeerNameRegistration`), вирішення конфліктів між портами клієнтів (`PeerNameResolver`) та передачі даних між клієнтах.

Для передачі даних потрібно створюючи пір створити в ньому канал, через який буде здійснюватися передача даних. В цей канал слід додати клас в якому буде описано механізми передачі та обробки даних (сервіси) позначивши його атрибутами `ServiceContract` та для методів `OperationContract`. Після цього викликавши метод контракту цей метод буде викликано у всіх користувачів мережі. Приклад сервісу, що описує взаємодію з каналом пірингової мережі показано на рисунку 4.4.

```
[ServiceContract(CallbackContract = typeof(IPingService))]
//Ссылка: 12
public interface IPingService
{
    [OperationContract(IsOneWay = true)]
    void Ping(HostInfo hostInfo);

    [OperationContract(IsOneWay = true)]
    void GetBlocks(HostInfo sender, HostInfo reciver);

    [OperationContract(IsOneWay = true)]
    void SendBlocks(HostInfo sender, HostInfo reciver, Block[] blocks);

    [OperationContract(IsOneWay = true)]
    void GetChainInfo(HostInfo sender);

    [OperationContract(IsOneWay = true)]
    void SendChainInfo(HostInfo sender, HostInfo reciver, int chainLength, bool chainStatus);

    void SendBlock(HostInfo sender, Block block);
}
```

Рисунок 4.4 – сервіс взаємодії з мережею P2P

4.1.3 Розробка системи управління блокчейном

Блокчейн система складається з декількох основних класів, це клас `Block`, та клас `Chain`.

Клас `Block` містить у собі всі ,вище зазначені у моделі блоку, поля. Методі хешування та перевірки хешу. І методи серелізації та десерелізації об'єкта.

Клас Chain містить у собі список блоків, посилання на P2P сервіс і посилання на самого себе (реалізує сінглтон). Також містить методи завантаження даних з локального сховища, порівняння з даними інших користувачів. І методи завантаження, відправки та синхронізації з глобальною мережею.

4.1.4 Розробка інтерфейсу

При розробці інтерфейсу користувача використовувалися стандартні форми WPF, зі стандартними UI елементами.

Windows Presentation Foundation (WPF) — це структура інтерфейсу користувача, яка створює настільні клієнтські програми. Платформа розробки WPF підтримує широкий набір функцій розробки додатків, включаючи модель програми, ресурси, елементи керування, графіку, макет, прив'язку даних, документи та безпеку [15].

WPF є частиною .NET, тому, якщо ви раніше створювали програми з .NET за допомогою ASP.NET або Windows Forms, досвід програмування повинен бути знайомим. WPF використовує Extensible Application Markup Language (XAML) для забезпечення декларативної моделі для програмування програм.

Основним в даній програмній системі є саме блокчейн, а зробивши невелику кількість маніпуляцій з системою управління блокчейну та з UI інтерфейсом можна, повністю змінити функціонал додатку інкапсулюючи дані блоків.

Для демонстрації було створено додаток, що містить чотири форми: логін, реєстрація, основна форма, та форма для перегляду результатів пошуку.

Сторінка реєстрації дозволяє зберігати новоствореного користувача в блокчейні, в блоці спеціального типу – User.

Сторінка логіну дає можливість зайти під ім'ям користувача, який вже міститься в блокчейні, або вирушити до сторінки реєстрації, щоб додати нового користувача.

Основна сторінка зверху відображає інформацію про користувача (логін, порт та uri end point). Майже всю область цієї сторінки займає список з блоків, що вже додані до блокчейну. Нажавши на будь який елемент списку можна зробити 4 операції з цим блоком :

- скопіювати хеш блоку;
- скопіювати ім'я користувача, що додав блок;
- переглянути вміст блоку (якщо це просте повідомлення, то воно відобразиться в вікні повідомлень, а якщо це якийсь файл, то він тимчасово завантажиться на комп'ютер та відкриється);
- скачати контент блоку на комп'ютер (якщо у блоці вказано тип файлу, то файл завантажиться з даним типом).

В лівому нижньому куті знаходиться функціонал для додавання нових блоків, або з звичайним строковим повідомленням, або передавши будь який файл з комп'ютера.

Також по середині міститься пошукова стрічка в яку можна вводити хеш блоку, або ім'я користувача і відповідно отримувати результат на свій запит в формі отримання результатів.

Дані форми показано на рисунку 4.5 – 4.8.

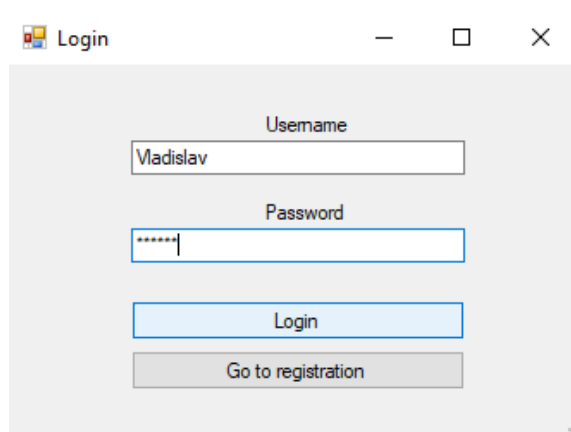
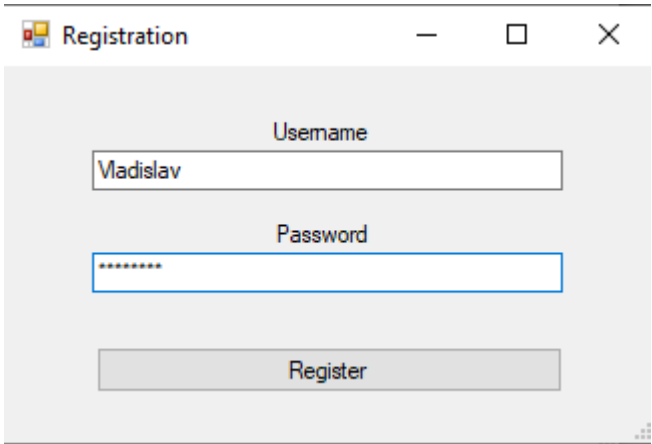
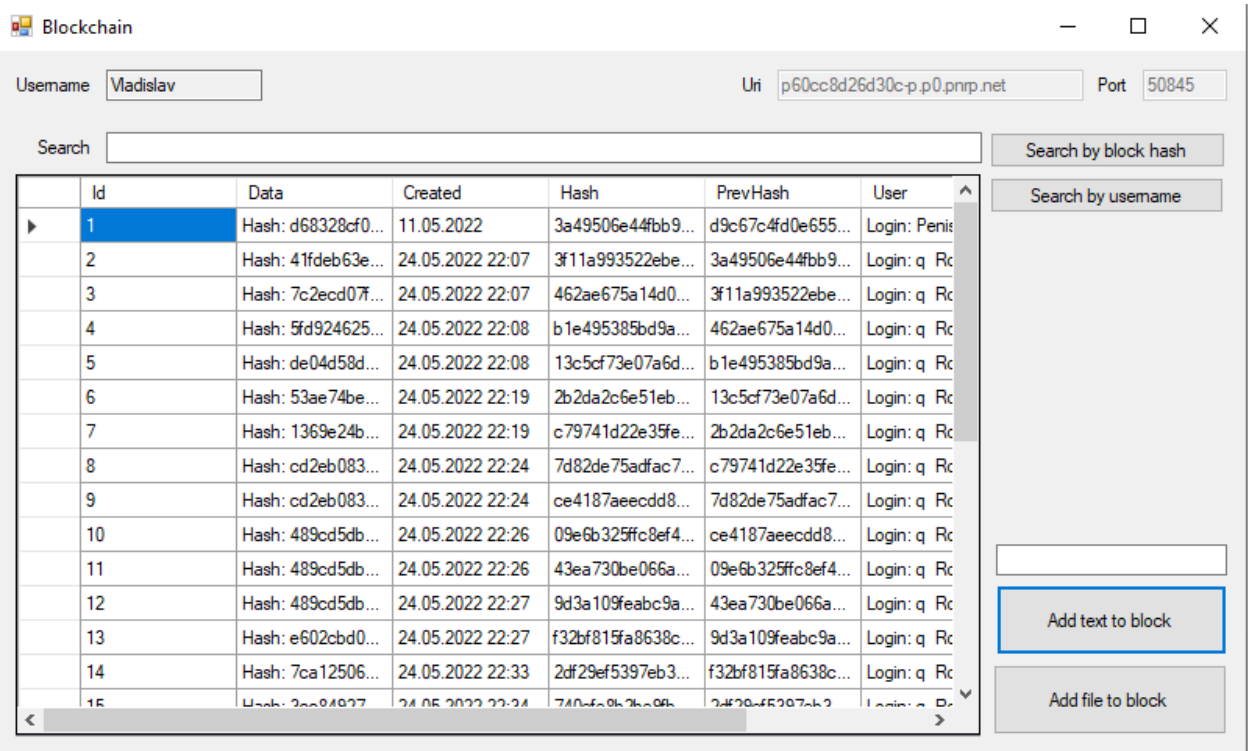


Рисунок 4.5 - Форма логіну



A screenshot of a 'Registration' window. It contains two input fields: 'Username' with the text 'Vladislav' and 'Password' with masked characters '*****'. Below these fields is a 'Register' button.

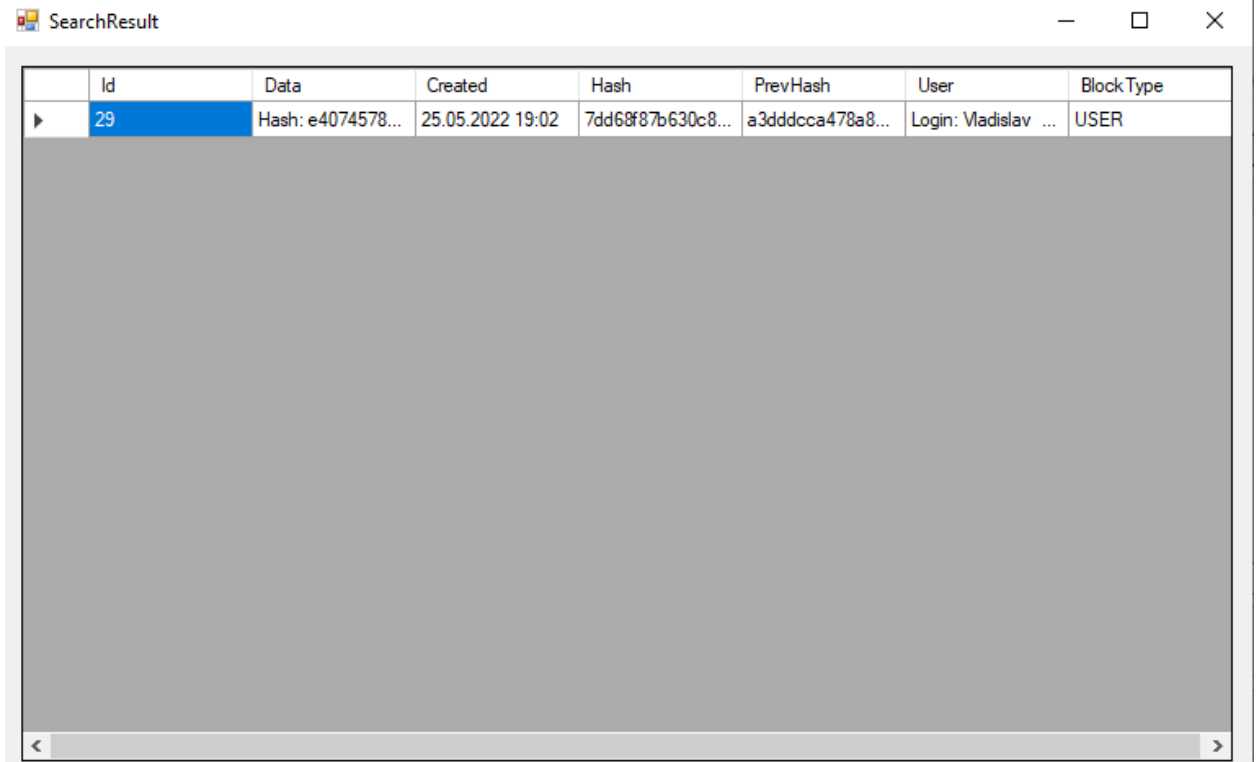
Рисунок 4.6 – Форма реєстрації



A screenshot of a 'Blockchain' application window. At the top, there are fields for 'Username' (Vladislav), 'Uri' (p60cc8d26d30c-p.p0.pnnp.net), and 'Port' (50845). Below these is a 'Search' bar. The main area contains a table with 15 rows of transaction data. To the right of the table are buttons for 'Search by block hash' and 'Search by username'. At the bottom right, there are two buttons: 'Add text to block' and 'Add file to block'.

	Id	Data	Created	Hash	PrevHash	User
▶	1	Hash: d68328cf0...	11.05.2022	3a49506e44fbb9...	d9c67c4fd0e655...	Login: Penit
	2	Hash: 41fdeb63e...	24.05.2022 22:07	3f11a993522ebe...	3a49506e44fbb9...	Login: q Rc
	3	Hash: 7c2ecd07f...	24.05.2022 22:07	462ae675a14d0...	3f11a993522ebe...	Login: q Rc
	4	Hash: 5fd924625...	24.05.2022 22:08	b1e495385bd9a...	462ae675a14d0...	Login: q Rc
	5	Hash: de04d58d...	24.05.2022 22:08	13c5cf73e07a6d...	b1e495385bd9a...	Login: q Rc
	6	Hash: 53ae74be...	24.05.2022 22:19	2b2da2c6e51eb...	13c5cf73e07a6d...	Login: q Rc
	7	Hash: 1369e24b...	24.05.2022 22:19	c79741d22e35fe...	2b2da2c6e51eb...	Login: q Rc
	8	Hash: cd2eb083...	24.05.2022 22:24	7d82de75adfac7...	c79741d22e35fe...	Login: q Rc
	9	Hash: cd2eb083...	24.05.2022 22:24	ce4187aeecdd8...	7d82de75adfac7...	Login: q Rc
	10	Hash: 489cd5db...	24.05.2022 22:26	09e6b325ffc8ef4...	ce4187aeecdd8...	Login: q Rc
	11	Hash: 489cd5db...	24.05.2022 22:26	43ea730be066a...	09e6b325ffc8ef4...	Login: q Rc
	12	Hash: 489cd5db...	24.05.2022 22:27	9d3a109feabc9a...	43ea730be066a...	Login: q Rc
	13	Hash: e602cbd0...	24.05.2022 22:27	f32bf815fa8638c...	9d3a109feabc9a...	Login: q Rc
	14	Hash: 7ca12506...	24.05.2022 22:33	2df29ef5397eb3...	f32bf815fa8638c...	Login: q Rc
	15	Hash: 2a84927...	24.05.2022 22:34	740f68b2b09b...	2df29ef5397eb3...	Login: q Rc

Рисунок 4.7 – Основна форма взаємодії з блокчейном



	Id	Data	Created	Hash	PrevHash	User	Block Type
▶	29	Hash: e4074578...	25.05.2022 19:02	7dd68f87b630c8...	a3ddcca478a8...	Login: Vladislav ...	USER

Рисунок 4.8 – Форма з результатами пошуку

4.2 Процес тестування

Тестування це перевірка відповідності між реальною поведінкою програми та її очікуваним поведінкою на кінцевому наборі тестів, обраному певним чином.

Тестування має на меті виявлення дефектів, підвищення впевненості в рівні якості, запобігання дефектів [16].

Для тестування програмної системи на етапі розробки, кожного разу при написанні якогось модуля, проводилось Unit тестування для цього модуля.

Модульний тест (Unit testing) — це спосіб тестування одиниці найменшої частини коду, яку можна логічно виділити в системі. У більшості мов програмування це функція, підпрограма, метод або властивість. Ізольована частина визначення має важливе значення. [17].

Також на при завершенні розробки, було проведено ряд тестування на відповідність функціональним та нефункціональним вимогам (функціональне тестування та нефункціональні тестування).

4.2.1 Модульне тестування

Обрана середовище розробки Visual studio дає можливість в автоматичній генерації шаблонів для модульного тестування. Приклад такого шаблону показано на рисунку 4.9.

```
[TestClass()]
Ссылка: 0
public class BlockTests
{
    [TestMethod()]
    Ссылка: 0
    public void GetSummaryDataTest()
    {
        Assert.Fail();
    }

    [TestMethod()]
    Ссылка: 0
    public void SerializeTest()
    {
        Assert.Fail();
    }

    [TestMethod()]
    Ссылка: 0
    public void DeserializeTest()
    {
        Assert.Fail();
    }

    [TestMethod()]
    Ссылка: 0
    public void ToStringTest()
    {
        Assert.Fail();
    }
}
```

Рисунок 4.9 – Згенерований шаблон модульного тестування для класу блок.

В такий шаблон слід записати логіку тестування, та викликати спеціальний метод – `Assert.AreEqual()` для того, щоб порівняти очікуваний результат виконання метода з тим, що справді виходить в результаті виконання метода. Приклад написаного модульного тесту, для копіювання логіна користувача в буфер обміну, показано на рисунку 4.10.


```
[TestMethod()]
Ссылка: 0
public void CopyLoginToBufferTest()
{
    Block genBlock = new Block();

    genBlock.CopyLoginToBuffer();

    Assert.AreEqual(genBlock.User.Login, Clipboard.GetText());
}
```

повністю відповідає цій вимозі. При передачі великих файлі таких як ,наприклад, картинка система додає блок за менш ніж одну секунду.

Доступність у використанні – відповідно до вимог інтерфейс має бути зрозумілим, для рядового користувача освоїтись повинно займати до 1 робочого дня. Система має дуже простий інтерфейс роботи з блокчейном, яка надає невелику кількість можливостей, але ці можливості можна використовувати задовольняючи будь які подальші надбудови над системою, наприклад інтеграцію аналогів смарт контрактів.

Надійність – система має стабільно працювати з мінімальною кількістю зависань при умові середнього навантаження на неї. Вході тестування на стресостійкість система гарно себе показала. При завантаженні 40 блоків з яких близько половини були файли середнім розміром у 500 кб, час очікування повністю новоствореного аканта на завантаження склало менш ніж 1 секунда на одному пристрої. А отже швидкість завантаження на різних пристроях може залежати в більшій мірі лиш від швидкості інтернета конкретного користувача.

Безпека – можливість підробити дані має бути можлива лиш в випадку переваги більш ніж 50% у апаратних можливостях конкретного користувача. Цей пункт також повністю задовільнений. При тестуванні двох програм, що використовують один апаратний потенціал, але звертаються до різних локальних сховищ, не вдалося обманути систему зламавши ланцюжок даних. Некоректні дані одразу затираються.

Локалізація – система повністю локалізована на англійській мові, відповідно до вимог.

Технічні вимоги – система працює на операційній системі Windows 10 та потребує менш ніж 8 гб оперативної пам'яті, згідно до раніше поставлених вимог.

Інтерфейс – повністю відповідає стилістичним вимог, що були поставлені раніше. Приклад інтерфейсу було представлено на рисунках 4.5-4.8.

4.3 Приклади використання

Розроблену систему можна використовувати наприклад в системах, що мають на меті медичний облік історій хвороб пацієнтів. Або звичайне збереження документів, наприклад про закінчення вищої освіти чи проходження якогось курсу. Також, можна зберігати невеликі програми, що можна використовувати як спрощену альтернативу смарт-контрактів Ethereum.

Усе це можна досягти не змінюючи саму систему, а лиш змінюючи найвищий шар взаємодії з системою та інтерфейс користувача, що є дуже вагомим плюсом системи.

В рамках демонстрації цих можливостей, було створено невелику програму для обліку історій хвороб та курсу лікування, що призначається хворому. Це, наприклад, дозволить притягнути некомпетентного лікаря до відповідальності у разі призначення ним некоректного курсу лікування чи некоректно поставленого діагнозу. Дана програма зображена на рисунках 4.12-4.13.

PatientName	PatientSurName	CurrentDate	Date
Иван	Полубокос	28.05.2022 0:32	28.05.2022 0:32
Владислав	Роксоланов	28.05.2022 0:34	28.05.2022 0:34

Patient Name: Владислав
 Patient Surname:
 Diagnosis: Рак
 Comment: Рак на начальном уровне развития, с шансом на излечение
 Analyzes: Опухоль лобной доли мозга
 Treatment: Химиотерапия 2 недели
 Date: 29 мая 2022 г.
 Save new

Рисунок 4.12 – Головна сторінка додатку

The screenshot shows a web application window titled "SelectViewForm". It contains a form with the following fields and values:

- Patient Name:** Ivan
- Patient Surname:** Полубоков
- Diagnos:** Спід
- Coment:** Жити лишилося трохи більш ніж 2 тижні
- Analyzes:** (Empty text area)
- Treatment:** Немає
- Date:** 28 мая 2022 г.
- Date of arise:** 28 мая 2022 г.

Рисунок 4.13 – Форма перегляду конкретної історії хворого

4.4 Можливі вдосконалення системи

При розробці програмного продукту були виявлені недоліки в архітектурі взаємодій між системою токенизації та системою Peer-To-Peer передачі даних. Ці проблеми слід виправити для більшої надійності, та покращенню можливостей подальших модифікацій.

Також слід провести оптимізацію збережень даних при ввімкненому застосунку, тому що, на разі усі блоки мають бути повністю завантажитися в оперативну пам'ять перед локальним збереженням, що є не оптимальним рішенням. А також слід створити механізми які будуть вирішувати питання постійного пере-записування локальної бази даних при вході в систему, тому що при подальшому масштабуванні, вхід в додаток може тривати вагомий відрізок часу.

ВИСНОВКИ

Для досягнення поставленої мети були проведені наступні роботи.

Був проведений аналіз предметної області. Виявлено основні особливості, та недоліки сучасних систем токенизації на основі технології блокчейн. Були досліджена проблема не достовірності інформації та гіпотетичні рішення цієї проблеми. Поставлені задачі, що слід вирішити для усунення сформованих проблем.

Було проведено дослідження різних алгоритмів, що використовуються в аналогічних системах. Виявлено переваги та недоліки даних алгоритмів. Були обрані найоптимальніші методики для розробки системи токенизації. Поставлені задачі, сформовані функціональні та нефункціональні вимоги, а також бізнес процеси, що мають виконуватися в системі.

Під час розробки були створенні архітектурні рішення для побудови майбутньої системи з урахуванням особливості блокчейну. Також було сформовано вектор розробки і узгоджені періодичні модульні тести для підтримки розробки проекту. Було розроблено систему токенизації на основі технології блокчейн, P2P сервіс, що надає доступ до системи, та додаток, що призначений надати візуальне відображення роботи системи токенизації і надати зручний та простий доступ до неї.

Підчас більш детального тестування були проведені функціональні та нефункціональні тести для того, щоб впевнитися в відповідності готового продукту раніше сформованим вимогам.

Загалом було спроектовано та розроблене програмне забезпечення, що зберігає будь яку інформацію, чи документ користувачів для того, щоб таким чином підтверджувати достовірність цих записів і внаслідок цього вирішити проблему підробок та фальсифікацій документів та інформації в цілому.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Про зареєстровані кримінальні правопорушення та результати їх досудового розслідування [Електронний ресурс] — Режим доступу до ресурсу: <http://www.gp.gov.ua/ua/stat.html> (дата звернення: 25.04.2021).
2. Архів Вінницького НДЕКЦ МВС: звітність Вінницького НДЕКЦ МВС // форма 1-ЕС, 2012-2018.
3. Архів Державного НДЕКЦ МВС: звітність ДНДЕКЦ МВС // форма 1-ЕС, 2012-2018.
4. Як придбати освіту [Електронний ресурс] — Режим доступу до ресурсу: <https://studway.com.ua/kupiti-diplom-nedorogo/> (дата звернення: 28.04.2021).
5. Огляд блокчейн [Електронний ресурс] — Режим доступу до ресурсу: <https://www.ibm.com/topics/what-is-blockchain> (дата звернення: 28.04.2021).
6. Що таке Ethereum [Електронний ресурс] — Режим доступу до ресурсу: <https://vc.ru/crypto/304025-chto-takoe-efirium-prostymi-slovami-i-chem-on-otlichaetsya-ot-bitkoina> (дата звернення 28.04.2021).
7. Що таке смарт-контракт [Електронний ресурс] — Режим доступу до ресурсу: <https://shalaginov.com/2021/08/05/what-is-smart-contract/> (дата звернення: 30.04.2021).
8. Bitcoin [Електронний ресурс] — Режим доступу до ресурсу: <https://bitcoin.org/> (дата звернення: 30.04.2021).
9. Блокчейн [Електронний ресурс] — Режим доступу до ресурсу: <https://habr.com/ru/post/569514/> (дата звернення: 30.04.2021).
10. Однорангові сіті [Електронний ресурс] — Режим доступу до ресурсу: <https://academy.binance.com/ru/articles/peer-to-peer-networks-explained> (дата звернення: 30.04.2021).

11. Кудь О. Цифрові активи: цифрові активи і їх економічно-правове регулювання в світлі розвитку технології блокчейн / О. Кудь, М. Кучерявенко, Є. Смичок., 2019. – 322 с.

13. Блокчейн в 200 строчок коду [Електронний ресурс] — Режим доступу до ресурсу: <https://habr.com/ru/post/323586> (дата звернення: 01.05.2021).

14. What is Entity Framework [Електронний ресурс] — Режим доступу до ресурсу: <https://www.entityframeworktutorial.net/what-is-entityframework.aspx> (дата звернення: 26.05.2022).

15. Developer tools, technical documentation and coding examples. [Електронний ресурс] — Режим доступу до ресурсу: <https://docs.microsoft.com/uk-ua/visualstudio/designers/getting-started-with-wpf?view=vs-2022> (дата звернення: 26.05.2022).

16. Навчальний ресурс з тестування програмного забезпечення [Електронний ресурс] — Режим доступу до ресурсу: https://qalearning.com.ua/theory/about_qa/shpargalka-z-testuvannya (дата звернення: 26.05.2022).

17. What Is Unit Testing [Електронний ресурс] — Режим доступу до ресурсу: <https://smartbear.com/learn/automated-testing/what-is-unit-testing> (дата звернення: 26.05.2022).