КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Економічний факультет
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА «КЕРУВАННЯ ФІЗИЧНИМИ АКТИВАМИ ЗА ДОПОМОГОЮ БЛОКЧЕЙНУ»

студента 4 курсу
спеціальності 051 «Економіка»
ОПП «Економічна кібернетика»
денної форми навчання
Анікушина Данила Романовича

Науковий керівник:
кандидат економічних наук,
доцент
Шпирко Віктор Васильович

Засвідчую, що у цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань

Студент_____
(підпис)

Роботу допущено до захисту перед ЕК
рішенням кафедри економічної кібернетики
від 9 червня 2022 року, протокол № 15
Завідувач кафедри:
доктор економічних наук, професор
Ляшенко Олена Ігорівна _____
(підпис)

КИЇВ – 2022

РЕФЕРАТ

Кваліфікаційна робота бакалавра містить: 40 ст., 2 рис., 1 табл., 20 - джерело.

Ключові слова: смарт-контракт, блокчейн, рікардіанський контракт, фізичні активи, машина з кінченою кількістю станів.

Об'єкт дослідження: дизайн смарт-контракту для керування фізичними активами.

Мета дослідження: визначення оптимальних стратегій керування фізичними активами за допомогою технології блокчейну.

Методи дослідження: метод порівняння, метод методи аналізу та синтезу, метод абстрагування, метод моделювання.

Наукова новизна, теоретична значимість дослідження: здійснено комплексний аналіз підходів щодо керування фізичними активами в мережі блокчейн, розроблено смарт-контракт для керування нерухомістю на основі теоретичного підходу з використанням машини з кінченою кількістю станів.

Практична цінність: можливість використання результатів роботи для створення смарт-контрактів для керування фізичними активами, зокрема в керуванню нерухомістю.

RESUME

Taras Shevchenko National University of Kyiv,

Faculty of Economics, Department of Economic Cybernetics

Key words: smart contract, blockchain, ricardian contract, real world assets, finite state machine.

The graduation research of student on managing real world assets on blockchain.

The work is interesting for using the results of work to model and implement solutions to problems of recording and managing real world assets via smart-contract and its applications.

Pages – 40, tables – 1, bibliog. – 20.

Зміст

ВСТУП

Актуальність теми. В останні роки інтерес до технології блокчейну зростає все більше. Блокчейн – це новітня технологія, яка ґрунтується на криптографії та функціонує як розподілена база даних. Про загальний інтерес може свідчити факт того, що у 2017 році ринкова капіталізація Біткоіну складала 70 мільярдів доларів, а у 2022 році вона досягла позначки в 10 разів більше — 700 мільярдів. Вперше про застосування цієї технології як основного компоненту для здійснення транзакцій цифрової валюти було згадано в 2008 році так званим Сатоші Накамото в його праці “Bitcoin: A Peer-to-Peer Electronic Cash System” [1].

З приходом блокчейну в світ сучасних технологій, поштовху отримала не тільки галузь транзакцій цифрової валюти, а багато інших галузей, таких як фінанси, аудит, торгівля в маркетплейсах, геймінг. Також за допомогою технології блокчейн можливо відслідковувати та керувати не тільки фінансовими активами, або активами які можуть знаходитися тільки у віртуальному просторі, а й фізичними активами, такими як реальні (фізичні) витвори мистецтва, приватна власність, і навіть звичайними товарами, відстежуючи їх на блокчейні, за допомогою технологій інтернету речей (IoT).

Все це стало можливо за допомогою смарт-контрактів — правил валідацій транзакцій, які можна запрограмувати і розмістити на блокчейн-платформі.

В останні пару років набув популярності концепт крипто-міст, а тобто перенесення багатьох процесів локального управління містом на блокчейн-платформу, задля досягнення таких корисних ефектів як пришвидшення бюрократичних процесів та більшої прозорості в процесах розподілу коштів та права власності, зокрема на фізичні активи. Саме тому ця робота є актуальною, адже вищеназвані властивості даної технології мають великий потенціал покращити вже існуючі моделі керування фізичними активами.

Мета і задачі дослідження. Метою даного дослідження є визначення оптимальних стратегій керування фізичними активами за допомогою технології блокчейну.

Завданнями дослідження є:

- аналіз сутності блокчейн технологій;
- дослідження існуючих методів керування фізичними активами;
- запропонувати методологію дизайну смарт-контрактів щодо керування фізичними активами;
- розробка смарт-контракту для керування нерухомістю.

Об’єктом дослідження є дизайн смарт-контракту для керування фізичними активами.

Предметом дослідження є методичне забезпечення та практичний інструментарій щодо розробки смарт-контрактів для керування фізичними активами.

Методи дослідження:

- Метод порівняння. Порівняння надасть можливість вибрати оптимальну альтернативу щодо застосування того чи іншого методу для впровадження додаткової логіки до блокчейну, яка буде розроблена в рамках даної роботи.
- Метод абстрагування. Даний метод надасть можливість уникнути несуттєвих властивостей і зв’язків в моделі керуванні фізичними активами.
- Метод аналізу та синтезу. Аналіз надасть можливість розібрати процес керування фізичними активами на складові дійові особи та функції. Синтез допоможе поєднати всі складові в одну цілісну машину з кінченою кількістю станів, якою буде наша кінцева логіка для імплементації на існуючій блокчейн-платформі.
- Метод моделювання. Моделювання дозволить нам дослідити поведінку реалізованої логіки керування та протестувати її на наявність логічних помилок.

Результати роботи можна використовувати на практиці при створенні систем керування фізичними активами.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТЕХНОЛОГІЇ БЛОКЧЕЙНУ ТА ЇХ ЗАСТОСУВАННЯ В ЗАДАЧІ КЕРУВАННЯ ФІЗИЧНИМИ АКТИВАМИ

1.1. Сутність технології блокчейну, термінологія

Блокчейн — це розподілена та незмінна база даних, яка надає можливість записування транзакцій та відслідковування активів в мережі [14].

Актив може бути матеріальним та нематеріальним.

Інформація є ключовою складовою економіки. Чим швидший та прозоріший доступ до інформації тим краще для економічного агента. Блокчейн є ідеальним засобом для розповсюдження інформації. Він надає миттєвий та повністю прозорий доступ до інформації, розміщеної в розподіленій базі даних.

Під розподільністю мається на увазі розміщення бази даних на багатьох комп’ютерах, що за допомогою механізму консенсусу дозволяє мати одну єдину версію бази даних, яка є коректною і визнається більшістю користувачів мережі блокчейн.

Кожна транзакція записується в блок транзакцій. Транзакція може містити будь-яку інформацію. Але чим більша транзакція, тим більше вона займає місця на фізичних машинах користувачів блокчейну, які утримують всю історію транзакцій на фізичній машині, що життєво необхідно для блокчейну. Отже, вартість здійснення такої транзакції зростає.

Також під час здійснення транзакції, користувач може заплатити не тільки вартість розміщення транзакції на блокчейні, а й будь-яку суму, яка йде як винагорода для валідатора поточного блоку. Чим вища ця сума, тим більш ймовірно що транзакція увійде в найближчий блок.

Кожен блок з’єднаний з попереднім та наступним блоком. Блоки формують ланцюг, кожен блок відображає послідовність транзакцій та час коли вона була здійснена. Ланцюг блоків є впорядкованим та постійно довшає.

Кожен блок містить часову позначку, хеш попереднього блоку та дані транзакцій, які подані як хеш-дерево.

Дана структура не дозволяє зловмисникам змінити інформацію, яка вже була верифікована та збережена на блокчейні. Кожен наступний блок посилює безпеку попередніх від підробки зловмисниками. Захистом від підробки та зміни даних є розміщення хешу всього блоку в наступному блоці. Відповідно внесення змін в один з блоків вимагає внесення змін у всіх наступних блоках, що є дуже затратною та складною операцією.

Валідатор — це комп'ютер, який підтримує цілісність блокчейну постійно виробляючи блоки та перевіряючи їх коректність починаючи з найпершого блоку (генезис блоку)

Алгоритм консенсусу — це процедура, через яку користувачі блокчейну досягають згоди щодо поточного стану розподіленої бази даних [15]. Таким чином мережа блокчейну створює довіру між користувачами, які не знають один одного. Кожен блок, який додається до ланцюгу блоків, існує в одному вірному екземплярі, на правдивість якого погоджується більшість користувачів.

Існує багато алгоритмів досягнення консенсусу. Найпоширеніші серед них:

- Доказ виконаної роботи (Proof of Work, PoW). Даний алгоритм обирає майнера для генерації наступного блоку. Кожен майнер бере участь в вирішенні математичного пазлу. Для цього необхідно задіяти великі розрахункові потужності. Хто вирішить пазл той генерує наступний блок. Біткойн використовує даний алгоритм консенсусу.
- Підтвердження частки (Proof of Stake, PoS). В цьому алгоритмі валідатори інвестують в частку активів системи блокуючи свої активи. Після цього валідатори голосують за блок який необхідно додавати до блокчейну. Якщо блок, за який вони проголосували, виграв це голосування, то він додається до блокчейну і всі, хто голосував за нього, отримують нагороду пропорційну до їх заблокованих активів. Всі, хто голосував за іншу версію блока, штрафуються. Ethereum перейшов з PoW на PoS. Цей алгоритм консенсусу є більш економічно доцільним, адже

майнерам не треба інвестувати в дороге апаратне забезпечення та велику кількість електроенергії.

- Підтвердження об’єму (Proof of Capacity, PoC). Валідатори інвестують в місце на жорсткому диску. В кого більше об’єму, той має більший шанс на створення наступного блоку.

Кожен користувач має в мережі блокчейн власну адресу, що є його публічним ключем (що є частиною асиметричного шифрування, публічний та приватний ключ необхідні для здійснення електронних підписів).

Всі транзакції в мережі блокчейн вільно відслідковуються (щонайменше, у загальному випадку). Це є основою для безпеки всієї системи. Знаючи, які дані йдуть від кожного користувача мережі, зловмисні дії швидко ідентифікуються та караються. Але через це транзакції відбуваються не повністю приватно. Більшість блокчейн-систем псевдо анонимні — вони не потребують детальної інформації про користувачів. Але всі транзакції відслідковуються, будь-хто може подивитися всі транзакції які походять з певної адреси в мережі.

Криптовалюта — це цифрова валюта, емісія та облік якої реалізується децентралізованою платіжною системою. Блокчейн-платформа може бути децентралізованою платіжною системою

Токен — це запис у блокчейні, сенс якого полягає у відображенні балансу в деяких активах, як матеріальних так і нематеріальних. Наприклад, деякі краудфандінгові системи видають токени в обмін на пожертвування. Також токени використовуються для авторизації та ідентифікації користувачів в мережі. Управління токеном відбувається за допомогою смарт-контракту, який містить в собі рахунки всіх власників даного токену та уможливлює передачу токенів з рахунку на рахунок змінюючи значення закріплене за певним користувачем. Наразі існує дуже багато стандартів, за якими створюються токени. Наприклад, ERC-20 є стандартом для базового функціоналу взаємозамінних токенів. ERC-721 є стандартом для невзаємозамінних токенів (Non-Fungible Token, або NFT) [13].

1.2. Смарт-контракти

Смарт-контракт — це програма, яка в мережі блокчейн має свою унікальну адресу. Ця програма складається з функцій та змінних поточного стану. Під час виконання функції, стан змінних, а значить і всього контракту, змінюється залежно від закладеної в функцію логіки.

Вперше цей термін було вжито Ніком Сабо у 1996 році, задовго до реалізації першої успішної децентралізованої електронної платіжної системи “Біткойн”. Під цим поняттям тоді малося на увазі покращені методи використання договірного права в електронній торгівлі [2].

Дослідимо переваги смарт-контрактів над звичайними контрактами у реальному житті. Наприклад, людині A необхідно обміняти n токенів на m токенів людини B. Для реалізації цієї транзакції в реальному житті Людині A треба довіряти людині B, що ніякого обману з її сторони не буде. Тому, щоб прибрати цю “довіру”, вона звертається до посередника, якому пересилаються n токенів людини A та m токенів людини B. Після цього посередник переправляє токени по місцю призначення, вирішуючи будь-які конфлікти інтересів, які можуть виникнути між людиною A і B. Але які гарантії, що посередник буде виконувати свої обов’язки перед сторонами A і B коректно? В цьому випадку нам вже буде необхідно довіряти репутації посередника, або якщо зайти ще далі — то репутації страхових компаній. Також в разі конфліктних ситуацій результат буде залежати від того, який саме посередник і на основі яких правил буде відбуватися вирішення конфліктного питання. Швидкість проведення операцій також має важливе значення. Зазвичай функціонування посередників має під собою дуже багато паперової роботи, та вирішення питань залежить від багатьох людей в цій організації, що породжує бюрократичність.

Отже, проблеми з традиційними контрактами наступні:

- необхідно довіряти іншій стороні контракту або посереднику;
- відсутність детермінізму у вирішенні конфліктних ситуацій;
- повільність.

Смарт-контракт не має таких недоліків. Він повністю автоматичний, детермінований та не залежить від людського фактору після розміщення в мережі блокчейн.

Смарт-контракти пишуться на високорівневих мовах програмування, наприклад Solidity для Ethereum, ink! для Polkadot і т.д. Після цього код смарт-контракту компілюється в байт-код і, коли контракт компілюється без помилок, завантажується на блокчейн. Користувачі блокчейну можуть викликати функцію , надсилаючи транзакцію до смарт-контракту. Після цього код відповідної функції виконується на кожному вузлі блокчейну, відповідно змінюючи внутрішній стан як і смарт-контракту, так і блокчейну в цілому.

Смарт-контракти можуть тримати в собі власні токени та цифрову валюту та спілкуватися з іншими смарт-контрактами.

Одним з головним недоліків смарт-контрактів є те, що вони дуже схильні до багів та вразливостей безпеки. Наприклад, дослідження проведене в 2016 році показало, що з 19336 смарт-контрактів на публічному блокчейні Ethereum 8333 мало щонайменше одну проблему з безпекою [16]. Такі проблеми потенційно можуть призвести до того, що зловмисники можуть викрасти з них цифрові активи. Наприклад, вразливість децентралізованої автономної організації “The DAO” у 2016 році призвела до того, що 3.6 мільйонів ефіру (а це 50 мільйонів доларів на той час) були викрадені, що призвело до hard fork (розколу) в Ethereum спільноті та розгалуження Ethereum на Ethereum та Ethereum Classic [17]. Також зловмисники можуть не викрадаючи цифрових активів зруйнувати логіку смарт-контрактів, що може призвести, наприклад, до того, що більше не можна буде вивести цифрові активи з смарт-контракту.

Більш того, після переміщення коду смарт-контракту на блокчейн транзакцією, його неможливо змінити та, відповідно, виправити (крім hard fork), що створює додаткові виклики для розробників смарт-контрактів.

Смарт-контракти як машини зі скінченною кількістю станів

Машина зі скінченною кількістю станів, або скінченний автомат — це абстракція, за допомогою якої описуються всі шляхи зміни стану об’єкту залежно від поточного стану та інформації ззовні, що викликає зміну стану.

Як можна побачити, смарт-контракт є машиною зі скінченною кількістю станів, де в нас є стан смарт-контракту (його змінні стану) та метод доставки інформації з зовнішнього світу для зміни внутрішнього стану (його функції).

Теоретичний підхід до розгляду смарт-контрактів як машин зі скінченною кількістю станів було запропоновано в роботі Anastasia Mavridou, Aron Laszka - “Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach” [3].

Як приклад був взятий смарт-контракт, який реалізував сліпий аукціон. В ньому учасник аукціону надсилає захешовану ставку та повинен внести депозит до смарт-контракту для того, щоб запобігти відмову учасника надсилати гроші після закінчення аукціону.

Сліпий аукціон має 4 фази:

1. Прийняття ставок.
2. Розшифрування значень ставок. Учасники надсилають не захешовані значення ставок і якщо хеш від них дорівнює хешу надісланої ставки на етапі прийняття ставок та був надісланий депозит то ставка остаточно приймається.
3. Кінцева фаза. Найвища ставка виграє аукціон, учасники можуть забрати свої депозити, окрім переможця. Він забирає різницю між депозитом та ставкою.
4. Фаза, коли відзиваються всі ставки.

Отже, в даному підході смарт-контракт має чотири стани, які відповідають вище описаним фазам. Також вони мають функції, що дозволяють контракту та учасникам аукціону змінювати стан контракту.

Смарт-контракт як машина з кінцевою кількістю станів має множину станів та множину переходів між цими станами. Перехід змушує контракт змінювати свій стан на основі певних умов.

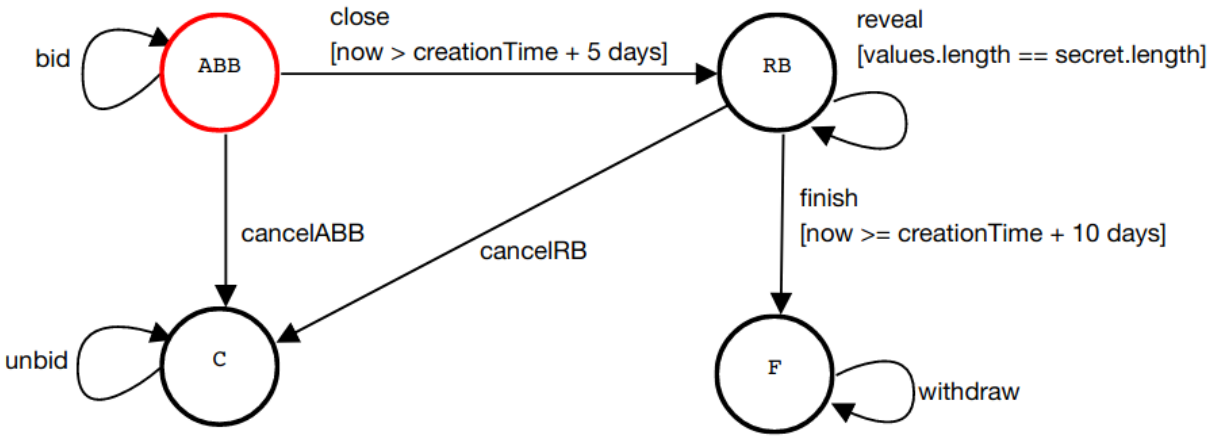

Рис. 1.1. Сліпий аукціон як машина з кінцевою кількістю станів
Рисунок взято з джерела 3

Рисунок відображає приклад сліпого аукціону як машину з кінцевою кількістю станів. Кожна транзакція (bid, reveal, cancel) відображає множину дій, які може здійснити учасник сліпого аукціону. Наприклад, учасник може здійснити bid перехід до ABB стану для того щоб зробити ставку та відправити депозит. Учасник може здійснити close перехід, але тільки якщо термін з початку торгів буде не більший ніж п’ять днів (now >= creationTime + 5 days).

Всі умови щодо переходів з одного стану в інший та можливі дії учасників ґрунтуються на множині даних, таких як:

- дані смарт-контракту (зберігаються в змінних стану контракту);
- вхідні дані (параметри функції за допомогою якої здійснюється перехід);
- вихідні дані (значення які повертають функції за допомогою яких здійснюється перехід).

C, I, O — три множини змінних (смарт-контракту, вхідних та вихідних даних).

$B[C, I]$ - множина предикатів на змінних стану смарт-контракту та вхідних даних.

$E[C, I, O]$ - множина виразів які можуть бути визначені в повному синтаксисі смарт-контракту.

$E[C, I, O]$- це повна множина всіх дій всіх переходів.

Визначення смарт-контракту як машини з кінцевою кількістю станів:

Смарт-контракт — це кортеж $(S, s_0, C, I, O, \rightarrow)$*, де*

- S *— кінцева множина станів;*
- $s_0 \in S$ *- початковий стан;*
- C*,* I*,* O *— це неперетинні кінцеві множини змінних смарт-контракту, вхідних та вихідних змінних.*
- $\rightarrow \subseteq S \times G \times F \times S$ *- перехідне відношення, де:*
  - $G = B[C, I]$*- множина умов:*
  - F *— множина дій, множина всіх підмножин* $E[C, I, O]$.

1.3. Децентралізовані автономні організації

Децентралізована автономна організація (Decentralized Autonomous Organization – DAO) – це організація, яка складається з правил у формі комп'ютерної програми, які контролюються членами даної організації і ніяк не центральною владою. Транзакції та правила децентралізованої автономної організації зберігаються на блокчейні. Через це вони автоматично отримують всі переваги мережі блокчейн [4].

Вся логіка децентралізованих автономних організацій закодована у формі одного чи декількох смарт-контрактів, які написані командою розробників даної організації та які є гарантією прозорості, яка дозволяє будь-якому учаснику або потенційному учаснику децентралізованої автономної організації повністю зрозуміти протокол на кожному його етапі.

В рамках логіки цих смарт-контрактів:

- учасники децентралізованої автономної організації мають прямий контроль над своїми активами в рамках організації;
- всі правила організації формалізовані та автоматизовані.

Після того як всі правила формально вписані в мережу блокчейн, наступним кроком є отримання фінансування. Зазвичай, це досягається завдяки випуску свого токену. Токен продається та цим самим збагачує децентралізовану автономну організацію.

Токен надає його власникові певні права в середині організації, а саме право голосу.

Після того як продаж токенів закривається, децентралізована автономна організація повністю розгортається в мережі блокчейн. На цьому етапі вже не є можливими будь-які зміни в логіці організації тільки командою розробників. Відтепер всі учасники децентралізованої автономної організації мають право голосу у вирішенні будь-яких питань всередині організації.

Історично, компанії діяли через певну особу чи групу осіб які керували організацією. Це було джерелом для таких проблем:

- люди не завжди дотримувалися правил;
- люди не завжди були згодні з правилами організації.

Децентралізована автономна організація вирішує ці проблеми таким чином:

- правил не можливо не дотримуватися, адже вся взаємодія з організацією відбувається через певні функції смарт-контракту, які мають власну незмінну логіку виконання;
- якщо люди не згодні з правилами організації – вони можуть ініціювати демократичні, не підпорядковані певній централізованій владі голосування щодо змін правил, а саме керування активами, технічні вдосконалення організації і т.д., або спокійно покинути організацію, виводячи з неї всі свої активи.

1.4. Многорівневий підхід до технологічних компонентів блокчейну

Блокчейн має багаторівневу структуру технологічних компонентів, таких як транзакції, блоки, механізм консенсусу, смарт-контракти, децентралізовані застосунки [12]. Всі ключові технологічні компоненти можуть бути поділені на 5 рівнів:

- рівень заліза;
- рівень даних;
- рівень мережі;
- рівень консенсусу;
- рівень застосунків.

Рівень заліза

Всі екземпляри певного блокчейну зберігаються на серверах по всьому світу. Клієнти можуть робити запити на певні дані на серверах під час користування інтернетом, або використовуючи різні застосунку. Це традиційно називається клієнт-серверною архітектурою.

В мережі блокчейн відсутня клієнт-серверна архітектура. Замість цього використовується peer-to-peer (P2P) мережа — велика група фізичних машин які спільно зберігають ідентичну інформацію, а саме вираховують, валідують та записують транзакцій користувачів мережі.

Вузол — це окрема фізична машина в мережі peer-to-peer.

Рівень даних

Структура даних блокчейну — це послідовний лист блоків, які містять в собі інформацію. Отже вона складається з двох основних елементів —

вказівників та зв’язаного списку. Зв’язаний список — це список блоків з даних які мають вказівники на свої попередні блоки.

Вказівники це змінні які мають в собі адресу позиції інших змінних.

В кожному блоці містяться дерева Меркла — бінарні дерева з хешами. Кожен блок містить хеш кореня дерева Меркла з хешами всіх транзакцій та таку інформацію як хеш блоку, часова мітка, номер версії блоку та іншу потрібну для конкретного блокчейну інформацію.

Для блокчейну дерево Меркла є дуже важливим, адже надає йому такі властивості як більша безпека, цілісність та неспростовність. Блокчейн побудований на деревах Меркла, криптографії, та алгоритму консенсусу.

Перший блок у зв’язному списку називається генезис-блоком, та відповідно не містить вказівника на попередній блок, бо його немає.

Для того, щоб захистити цілісність даних, транзакції підписуються електронними підписами. Приватний ключ використовується для того щоб підписати транзакції. Публічний ключ використовується для того щоб перевірити, що транзакція підписана власником приватного ключа, що йде в парі з даним публічним ключем. Також публічний ключ використовується як адреса в мережі блокчейн. Будь яка маніпуляція з транзакцією після її підпису інвалідує підпис транзакції.

Таким чином транзакції не можуть бути підробленими.

Рівень мережі

Рівень мережі, або рівень peer-to-peer відповідає за механізми комунікації між вузлами мережі. Знаходження інших вузлів, прийом транзакцій та додавання блоків — це все задачі саме цього рівня.

Також цей рівень повинен гарантувати, що всі вузли можуть взаємодіяти між собою та підтримувати мережу блокчейн у валідному стані.

Рівень консенсусу

Рівень консенсусу є найбільш важливим та критичним для будь-якого блокчейну. Саме цей рівень відповідає за те, який блок треба пропускати в мережу, а який ні, та є гарантією що всі вузли в мережі на це погодяться.

Рівень застосунків

Смарт-контракти та децентралізовані автономні організації знаходяться на цьому рівні.

Цей рівень поділяється на рівень застосунків та рівень виконання коду.

Також до цього рівня входять всі застосунки та програми, за допомогою яких користувачі блокчейну можуть взаємодіяти з мережею блокчейн — це програмний інтерфейс застосунків (API, Application Programming Interface) та всі фреймворки які дають доступ до інформації збереженої на блокчейні.

Всі інші рівні слугують бек-ендом для цих застосунків.

Хоча транзакція рухається з рівня застосунків до рівня виконання коду, вона ще проходить процес валідації на семантичному рівні.

Рівень застосунків надає інструкції для рівня виконання коду.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ І ПІДХОДІВ ДО КЕРУВАННЯ ФІЗИЧНИМИ АКТИВАМИ НА БЛОКЧЕЙНІ

2.1. Приклади керування містом та нерухомістю за допомогою мережі блокчейн

- Reno DAO — децентралізована автономна організація міста Рено. Мер міста Рено у штата Невада, США запропонувала заснувати Reno DAO [18] — мешканці міста отримують токени, які відображають частку суспільної нерухомості міста, яка винаймається. Завдяки цим токенам мешканці зможуть отримувати дохід від здачі в оренду суспільної власності та голосуванням вирішувати нагальні питання щодо керування містом.
- CityDAO — децентралізована автономна організація, яка має за мету створення цілих міст з нуля, які б керувалися учасниками цієї децентралізованої автономної організації. Наразі цей проект здобув легальний статус у штаті Вайомінг, США та його учасники вже почали купувати та переносити на блокчейн куплені території [19].

Створення міст та інфраструктури, які повністю керуються за допомогою децентралізованих автономних організацій, відкривають великий простір для розвитку демократії [5]:

- всі існуючі бюрократичні процеси переносяться у прозорий та повністю підлягаючий верифікації світ блокчейну, будуючи більшу довіру в існуючі процеси;
- виникають абсолютно нові та експериментальні форми власності.

Блокчейн системи набагато ефективніші ніж будь-які нецифрові рішення керування містом та набагато прозоріші ніж керування за допомогою паперової демократії та централізованих закритих від суспільства цифрових систем.

Наступні процеси можуть бути знатно покращені шляхом побудови більшої довіри до них:

- Чесні рандомні генератори, не розміщені на закритих від стороннього ока централізованих серверах. Це необхідно, наприклад, для лотерей.
- Різноманітні сертифікати та документи. Наприклад, право власності на певну нерухомість, свідоцтва про народження, посвідчення що людина є резидентом певного міста. За допомогою такого рішення як “Доказ існування” (Proof of existence, PoE) можливо зберігати хеші від документів на мережі блокчейн. Наприклад, коли в якійсь організації буде необхідна довідка про несудимість певної особи, це зможуть перевірити на відповідному блокчейні.
- Ведення обліку активів резидентів міста.
- Голосування.

Але не всі вищеописані можуть бути повністю децентралізованими. Наприклад, вилучення певного майна за рішенням суду не може бути реалізованим без централізованої влади. Але може набувати більшої децентралізованості, якщо в цьому процесі будуть брати участь декілька незаінтересованих сторін.

2.2. Приклади підходів до керування та відслідковування товарів у ланцюгу поставок в мережі блокчейн

Ідентифікування продуктів у ланцюгу поставок є дуже важливою задачею. Для кінцевих покупців дуже важливим є питання оригінальності продуктів, їх якості, їх походження. Для того щоб відслідковувати та ідентифікувати продукти на всіх етапах його виробництва та транспортування необхідні дві речі:

- надійні сенсори, які можуть розпізнати продукт;
- надійне сховище даних про переміщення та стан продукту.

Вирішенням першого питання, як розпізнати продукт на етапах ланцюгу поставок, є мітки RFID (Radio frequency identification — радіочастотна ідентифікація), які переносять ідентифікаційну та іншу інформацію. Хоча цей метод надає можливість відслідковувати товар, він не є безпечним, адже RFID-мітки можливо підробити.

Надійне сховище для даних щодо стану товару в ланцюгу поставок організувати нескладно, але кінцевий споживач просто не буде мати доступу до цієї інформації. В цьому питанні рішенням виступають мережі блокчейн.

Але товарів, які циркулюють в глобальній мережі ланцюгів поставок неймовірна велика кількість, а об’єм транзакцій, які можуть обробити блокчейни в рамках розподіленого сховища даних, яке повинно бути збереженим на кожному фізичному комп’ютері, який підтримує мережу, є дуже обмеженим.

Вирішення першої проблеми є “Кракелюр-відслідковування” (Craquelure-based Tracking) [6]. “Кракелюр” — це випадковий узор з тріщин, наприклад, на старій картині. Процес висихання фарби гарантує випадковий та унікальний узор тріщин. Фінальний узор може використовуватися як унікальний відбиток кожного продукту.

Рішенням другого питання є OriginStamp — система для децентралізованого зберігання часових міток [7]. За допомогою цього продукту, стани певного продукту можуть бути записані та верифіковані у часі зберігаючи SHA-256 хеш продукту на блокчейні Bitcoin. Після збереження хешу разом з часовою міткою запис практично неможливо підробити. Також по хешу, збереженомуу на блокчейні, неможливо отримати інформацію про продукт — хеш це функція яка працює тільки в одну сторону. Але, вже маючи продукт, кінцевий може подивитися всі часові мітки продукту, порахувавши його хеш та відповідно знайшовши його записи на блокчейні.

Оскільки кількість даних, яка може передаватися транзакціями на блокчейні, є дуже обмеженою, замість того щоб зберігати кожні часову мітку з хешем окремою транзакцією, всі хеші за певний період збираються, сортуються

у алфавітному порядку, конкатенуються та знову хешуються. Таким чином досягається велика економія коштів на зберігання часових міток разом з відповідними їм хешами.

2.3. Рікардіанський контракт

Рікардіанський контракт — це контракт, який збережений у цифровому вигляді, повністю визнається юридично та повністю непідробний. Отже, це смарт-контракт, який визнається юридично. Також це є звичайним контрактом у широкому розумінні цього слова з відмінністю в тому, що він зберігається в мережі блокчейн, є повністю непідробним та прозорим для усіх сторін [8].

Коли певні сторони ведуть певну активність, пов’язану з контрактом, вони просто можуть посилатися на нього використовуючи його хеш. Цей контракт є валідним і для машин і для людей — програма містить таку ж логіку як і написаний на папері документ.

На відміну від смарт-контракту, рікардіанський контракт не тільки відображає дії, а також наміри. Контракт використовує хеші для посилання на певні документи.

Переваги рікардіанського контракту:

- Контракт є валідним у правовому полі. Це означає, що він є валідним у суді при вирішенні суперечностей.
- Може бути прочитаним та використаним людьми та машинами.

У найближчому майбутньому рікардіанські контракти тісно переплітуться зі смарт-контрактами, формуючи юридичні контракти на блокчейні.

Приклади застосування рікардіанського контракту:

- Peer-to-peer комерція. Для відслідковування комерційних відносин на онлайн маркетплейсах.
- NFT. Блокчейн-протоколи використовують комбінацію смарт-контрактів, рікардіанських контрактів для створення NFT. Наприклад,

Mattereum [8] протокол використовує рікардіанські протоколи для того, щоб електронно підписати та зберегти NFT на блокчейні.

- Діджиталізація юридичних контрактів. Рікардіанські контракти можуть створити стандартний шаблон для складних юридичних контрактів, які дуже полегшують процес валідації та гарантують непідробність контрактів.

Загалом можна сказати, що рікардіанські контракти ще більше посилюють зв'язок децентралізованого віртуального світу блокчейну з фізичними активами. Тепер можна покладатися не тільки на логіку смарт-контрактів, зв'язок яких з фізичним світом є дуже обмеженим (вони можуть спілкуватися з навколишнім світом тільки через "оракли", які надають параметри до функцій смарт-контракту зі світу який існує поза блокчейну, на якому знаходиться даний смарт-контракт), а й на контракти й незалежних експертів поза блокчейном, у яких є більше доступу до світу фізичних активів.

2.4. Оракли як спосіб поєднання даних поза блокчейнів з програмною логікою на блокчейнах

Оракли — це програмні компоненти які дозволяють мережі блокчейн мати доступ до даних, розташованих на зовнішніх системах, дозволяючи смарт-контрактам мати вхідні дані до своїх функцій зі світу поза мережею блокчейн.

Оракли надають для децентралізованих мереж блокчейну доступ до вже існуючих джерел даних, застосунків та обчислювальних потужностей, які не доступні в обмеженому розподіленому середовищі.

Децентралізовані мережі ораклів надають можливості щодо створення так званих гібридних контрактів [9] — коли інфраструктура децентралізованого світу та світу поза ним об'єднуються для підтримки покращених децентралізованих застосунків, які можуть реагувати на події в реальному світі та взаємодіяти з традиційними системами.

Дуже велика кількість галузей можуть використовувати гібридні смарт-контракти та вирішувати різноманітні задачі в децентралізованому світі блокчейну, такі як визначення вартості активів для фінансових застосунків, інформація про погодні умови для застосунків у сфері страхування, джерела генерування випадкових чисел для ігор на блокчейні, сенсори інтернету речей для ланцюгів поставок, верифікація осіб для державних застосунків та багато інших.

Механізм ораклів повинен бути добре спроектований, адже просте рішення таке як постачання даних з одного централізованого джерела руйнує всю ідею децентралізованих застосунків. Бо одне джерело даних означає єдине джерело в якому може статися помилка, наприклад джерело перестане бути доступним онлайн, або хтось буде постачати невірні дані. Це також веде до того, що помилкові дані, які були передані з ораклу, записуються на мережу і після цього не є можливим це виправити.

Рішенням цієї проблеми є децентралізована система ораклів, яка поєднує певну множину джерел даних для зменшення впливу окремих ораклів.

Існують такі види ораклів:

- Оракли для вхідних даних. Найбільш поширені оракли. Вони дістають дані з джерел навколишнього світу та доставляють їх як параметри в смарт-контракти.
- Оракли для вихідних даних. Дозволяють смарт-контрактам надсилати свої вихідні дані до навколишнього світу. Наприклад для того, щоб проінформувати фінансову установу, що на мережі блокчейн відбулася певна транзакція.
- Міжланцюговий оракл. Даний оракл може передавати інформацію та активи між різними блокчейнами.
- Оракл для складних обчислювальних операцій. Мережі блокчейну не пристосовані до важких обчислень, бо це було б дуже дорого та неефективно. Саме тому були створені оракли, які б поєднували багато джерел обчислень для смарт-контракту поза мережею.

Дана технологія є надзвичайно важливою для керування фізичними активами, адже поєднує інформації з реального фізичного світу з мережею блокчейн.

РОЗДІЛ 3. РОЗРОБКА СМАРТ-КОНТРАКТУ ДЛЯ КЕРУВАННЯ ФІЗИЧНИМИ АКТИВАМИ

Процес розробки смарт-контракту було вирішено поділити на три фази:

1. Фаза аналізування
2. Фаза дизайну
3. Фаза імплементації

3.1. Фаза аналізування

Вибір мови програмування для написання смарт-контракту

Наразі найпопулярнішою мовою програмування для написання смарт-контрактів є Solidity. Вона була розроблена в рамках проекту Ethereum для трансляції в байт-код віртуальної машини Ethereum.

В останні роки дуже широкого розповсюдження набула технологія веб-ассемблеру (WebAssembly) [11] — універсальний низькорівневий та високоефективний проміжний код для виконання в браузері. По швидкості він дуже наближений до машинного коду, але при цьому на відміну від машинного коду є платформо незалежним, а отже може однаково детерміністично виконуватися в будь-якому браузері який підтримує дану технологію. Він компілюється в дуже компактні бінарні файли, що є дуже важливим для зберігання та виконання на блокчейні, де кожен байт та кожна машинна інструкція є дуже коштовною, адже код повинен зберігатися та виконуватися зразу на всіх фізичних машинах в мережі. Також в нього може компілюватися код будь-якої мови програмування.

Мова програмування Rust є мовою з найкращою підтримкою компіляції в веб-ассемблер.

Для розробки смарт-контракту було обрано вбудовану предметно-орієнтовано мову програмування (embedded domain-specific language, eDSL) ink!, основою для якої є мова програмування Rust [10]. Вона використовується

для блокчейнів створених з допомогою фреймворка для створення блокчейнів Substrate.

Предметно-орієнтована мова програмування (domain-specific language, DSL) — це мова програмування, яка створена для вирішення задач конкретної предметної галузі. Застосування предметно-орієнтованої мови програмування уможливлює опис та вирішення специфічних завдань таких, як, наприклад, в нашому випадку написання смарт-контрактів. “Вбудована” означає що вона є підмножиною мови програмування загального призначення (в нашому випадку предметно-орієнтована мова програмування ink! є підмножиною мови програмування загального призначення Rust).

Також ряд інших переваг обумовлює вибір підмножини мови Rust:

- мова має сувору типізацію та забезпечує безпечну роботу з пам’яттю;
- при компіляції генерує дуже маленькі бінарні файли, наприклад, Rust переставляє поля структури для того щоб вона займала мінімальну кількість пам’яті ;
- має мінімальний час виконання;
- має дуже велику екосистему доступних бібліотек та інструментів, які значно полегшують розробку застосунків.

Таблиця 3.1

Порівняльна таблиця мов програмування для написання смарт-контрактів

	ink!	Solidity
Віртуальна машина	Будь-яка з підтримкою веб-ассемблера	EVM (віртуальна машина Ethereum)
Компілюється у	Веб-ассемблер	Байт-код EVM
Мова програмування	Rust	Solidity
Захист від програмного переповнення	Увімкнений за замовчуванням	Відсутній

Продовження таблиці 3.1

	ink!	Solidity
Кількість конструкторів	Один або більше	Один
Інструментарій	Всі що підходять до мови програмування Rust	Власний (тільки той що підходить до Solidity)
Версіонування	Семантичне	Семантичне
Містить метадані?	Так	Так
Проект може містити багато файлів	Ні	Так
Об’єм слотів в пам’яті під змінні	Варіюється	Тільки 256-бітні
Має інтерфейси?	Так (трейти Rust)	Так

Процес розгортання контракту на мережі блокчейн

Процес розгортання контракту на мережі блокчейн поділяється на такі етапи:

- Написання смарт-контракту на вбудованій предметно-орієнтованій мові програмування ink!;
- компіляція коду смарт-контракту в веб-ассемблер за допомогою утиліти cargo-contract, як результат цього етапу маємо код на веб-асемблері та метадані, які включають в себе двійковий програмний інтерфейс (ABI — Application Binary Interface) та файл в форматі .contract, який містить в собі два вищезазначених файли. Саме файл у форматі .contract розгортається на блокчейні. Метадані включають в себе такі секції як опис всіх функцій які є в коді смарт-контракту, опис типів даних, які використовуються в програмі, та опис всіх даних, які зберігаються в смарт-контракті;
- безпосередня розгортка смарт-контракту на мережі блокчейн. Цей етап поділяється в Substrate на два етапи: загрузка коду на веб-асемблері на блокчейн і потім створення екземпляру контракту. Цей етап дуже

різниться від інших подібних рішень в створенні смарт-контрактів. В них загрузка коду на створення самого смарт-контракту — це єдиний процес і кожен раз, коли необхідно створити смарт-контракт, який лише відрізняється параметрами при конструюванні, необхідно завантажувати код, коли код в Substrate завантажується один раз і потім може бути створений з різними параметрами стільки скільки потрібно;

- при створенні нового екземпляру смарт-контракту, Substrate створює нову адресу в мережі блокчейн для того щоб зберігати всі активи смарт-контракту та для того щоб інші користувачі блокчейну могли взаємодіяти зі смарт-контрактом.

Опис смарт-контракту

Під час фази аналізування, необхідно провести збір вимог до розробленого децентралізованого застосунку з точки зору різних категорій користувачів застосунку.

В даній роботі буде розроблений смарт-контракт “Land”, створений для відображення права власності на певну нерухомість в мережі блокчейн та реалізації можливості винаймати нерухомість в мережі блокчейн.

Ролі в смарт-контракті:

- Власник смарт-контракту. Має контроль над додаванням (а також вилученням) власності закріпленої за певним власником до сховища смарт-контракту. За свої послуги отримує певний відсоток з вартості оренди.
- Власник нерухомості. Має контроль над встановленням ціни оренди за місяць, підбору орендаря та його вилучення зі статусу орендаря своєї власності. Також може видалити свою власність зі сховища смарт-контракту.
- Орендар. Реєструється власником нерухомості, повинен платити за оренду щоб повністю вступити в статус орендаря. Може продовжити

термін оренди. Може бути вилученим власником нерухомості. Якщо вилучення відбувається не через несплату оренди, а з інших причин — дискусія щодо цього переходить у простір поза блокчейном. Головна ідея полягає в тому, що факт спочатку прийняття орендодавцем потенційного як орендаря та сплати орендарем ренти зберігається на блокчейні і не можу бути підробленим та видаленим, як це може статися, наприклад, з реальними фізичними документами, або документами, які розташовані на сервері, або групі серверів, які контролюються однієї організацією та закриті від звичайних людей.

Як ми бачимо, в даному смарт-контракті є центральна влада, а саме власник-смарт-контракту, який може додавати нерухомість до внутрішнього сховища смарт-контракту, а також видаляти її.

З одного боку, це суперечить одній з основних ідей децентралізованих застосунків — відсутність центрального актора, який міг би впливати на те що відбувається на блокчейні. Але в даному випадку центральна влада все ж таки повинна бути, адже хтось повинен проводити моніторинг зі сторони на, як мінімум, існування нерухомості та оспорювати всілякі суперечності між власником нерухомості та орендарем.

Головна ідея цього застосунку полягає тому, що переважаюча більшість всіх відносин (реєстрування нерухомості, реєстрування особи як орендаря, сплата оренди, відслідковування термінів оренди) відбувається на мережі блокчейн, а отже виносить основну частку всіх взаємодій між власниками нерухомості та орендарями на децентралізовану, прозору для всіх спостережників мережу блокчейн. Лише невелика частка взаємодій, яка включає в себе вирішення всіляких суперечностей коли один, або обидва акторів із множини власник-орендар поводять себе нечесно. Але навіть в такому випадку частка їх можливих доказів в цьому процесі вже записана на блокчейні, а ці записи підробити практично майже неможливо.

Згодом у роботі буде описано метод, згідно з яким можна буде уникнути центральної влади у нашому смарт-контракті.

Основні процеси смарт-контракту

В нашому випадку, процеси відбувається з участю трьох учасників.

Основних процесів смарт-контракту 3:

1) Підписування контракту про оренду. В цьому процесі власник нерухомості відповідає на заявку орендаря про оренду нерухомості. Цей процес ділиться на дві стадії:

- Перша стадія. Власник отримує id аккаунта орендаря та вносить його у внутрішнє сховище смарт-контракту, яке містить список орендарів закріплених за певною нерухомістю.
- Друга стадія цього підписання — це сплата вартості оренди орендарем, а отже це означає повне укладання контракту та право орендаря користуватися нерухомістю. Якщо друга стадія не відбувається (орендар не сплачує ренти), власник нерухомості ініціює третій основний процес смарт-контракту — процес видалення орендаря зі списку орендарів власником нерухомості.

2) Отримання коштів власником смарт-контракту (відсоток за послуги, та/або налог, залежно від того хто є власником смарт-контракту та в яких юридичних рамках відбуваються його процеси) та власником нерухомості. Цей процес завершує перший процес (другу стадію).

3) Процес вилучення орендаря з реєстрів смарт-контракту. Цей процес є неоднозначним, адже практично власник смарт-контракту чи власник нерухомості може вилучити орендаря у будь-який момент. Несправедливі рішення можуть бути оскаржені в суді. При цьому докази укладення контракту та сплати ренти будуть збережені в мережі блокчейн у будь-якому випадку.

В такому випадку даний контракт буде рікардіанським і повинен мати ще додаткові документи, які б прикріплювали його юридичне значення.

Інше вирішення цього питання — це використання залогів [20] на блокчейні в рамках цього контракту які і власником нерухомості, так і орендарем. Всі суперечки між ними вирішувалися б за допомогою системи ораклів та, можливо IoT сенсорів. Друге рішення є більш бажаним, адже вилучає потребу в рішеннях доволі централізованої інституції як суд. Але воно є більш складним, щонайменше в рамках даної роботи. Тому було прийняте рішення обмежитися реалізацією частини рікардіанського контракту, яка знаходиться на мережі блокчейн у рамках смарт-контракту.

3.2. Фаза дизайну

Майже всі взаємодії в рамках смарт-контракту відбуваються між власниками нерухомості та орендарями.

Ціль контракту — це забезпечити прозоре підтвердження того, що договір про оренду підписаний та рента сплачена вчасно.

Згідно з підходом розглянутим в розділі 1.3, а саме представлення смарт-контракту як машини зі скінченною кількістю станів, було розроблено представлення нашого смарт-контракту у вигляді машини зі скінченною кількістю станів.

Оскільки наш смарт-контракт може відображати стан багатьох нерухомостей, то розглянемо стан однієї такої нерухомості в рамках смарт-контракту.

Початковий стан — це “Нерухомість не розташована на блокчейні” (“Property not on chain”). Після того як власник смарт-контракту зареєструє певну власність за певним власником на смарт-контракті (функція approve_property), нерухомість буде зареєстрована на блокчейні. Нерухомість переходить в стан “Нерухомість зареєстрована на блокчейні, але не орендується” (“Property not borrowed”). Після цього власнику слід виставити ціну оренди в місяць за допомогою методу set_price. Після цього власнику

можна буде ініціювати метод approve_tenant (звісно якщо такий орендар знайдеться) і після цього орендар повинен сплатити ренту. В залежності від того скільки він заплатить, на стільки місяців і орендує. Термін оренди та початкова дата вноситься у внутрішнє сховище смарт-контракту. Нерухомість переходить в стан “Нерухомість зареєстрована на блокчейні та орендується”.

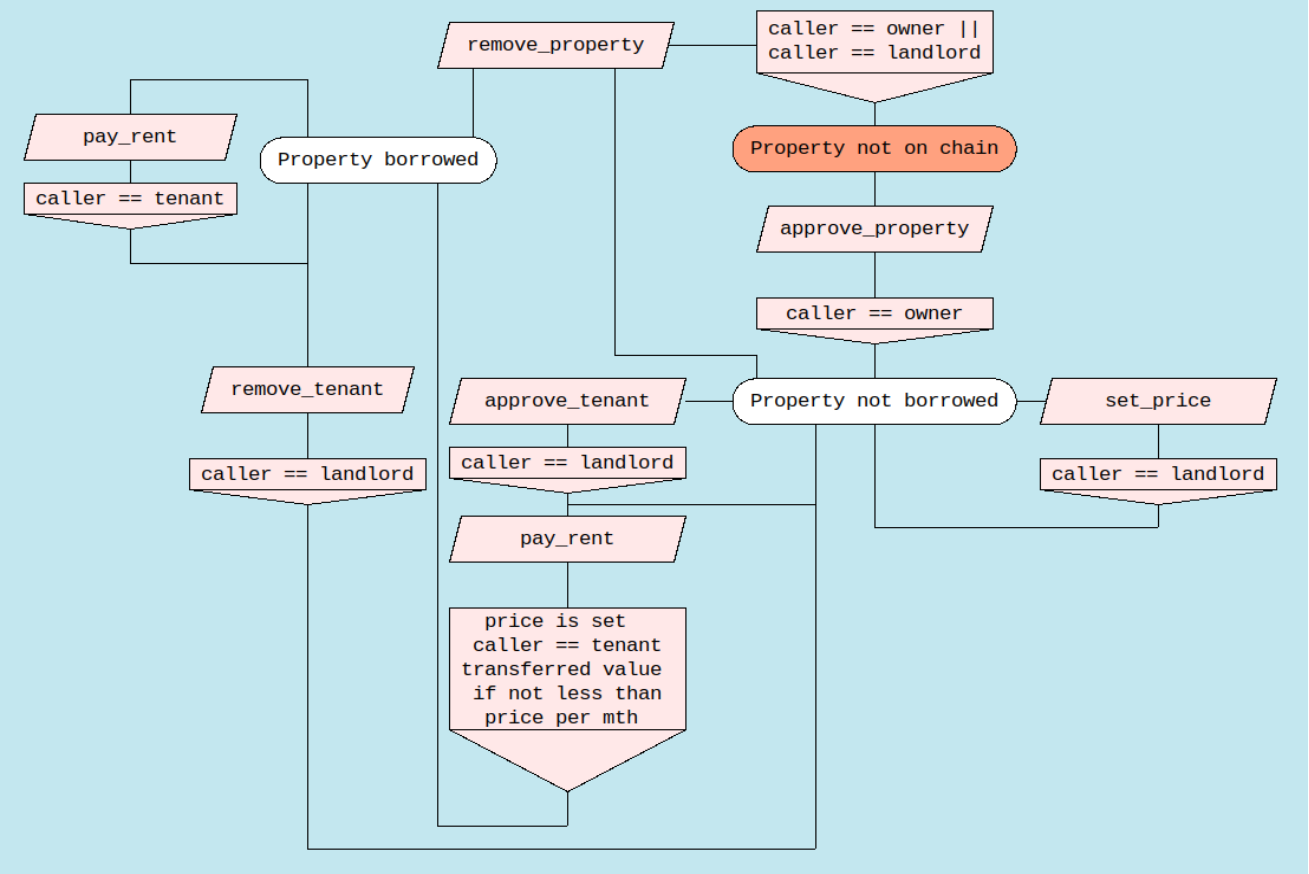

Рис. 3.1. Смарт-контракт у вигляді машини зі скінченною кількістю станів

Джерело: рисунок автора на основі джерела 3

Після цього орендар може або платити ренту і продовжувати свій термін оренди, або він може бути видалений зі статусу орендар власником нерухомості. Тоді нерухомість переводиться у стан “Нерухомість зареєстрована на блокчейні, але не орендується” (“Property not borrowed”). Також у будь-який момент власність може бути вилучена з блокчейну (або власником нерухомості, або власником смарт-контракту). Таким чином нерухомість буде повернута в

початковий стан “Нерухомість не розташована на блокчейні” (“Property not on chain”).

3.3. Фаза імплементації

На основі міркувань в розділах 3.1 та 3.5 було розпочато роботу щодо імплементації смарт-контракту на вбудованій предметно-орієнтованій мові ink!, спроектованого у формі машини з кінченою кількістю станів.

Весь процес був поділений на етапи:

1. Проектування типів даних, які будуть використовуватися смарт-контрактом.
2. Проектування подій, які публікуються в мережі блокчейн
3. Написання інтерфейсу смарт-контракту.
4. Визначення структур даних, необхідних для відображення внутрішнього стану смарт-контракту.
5. Проектування можливих помилок
6. За допомогою методу керованої тестами розробки описати поведінку функцій смарт-контракту. Це робиться шляхом написання тестів та перевірки поведінки написаних функцій.
7. Імплементація функцій.
8. Запуск локального тестового блокчейну.
9. Розгортування смарт-контракту на локальному блокчейні за допомогою застосунку https://paritytech.github.io/contracts-ui.
10. Мануальне тестування смарт-контракту.

Проектування типів даних

До вже існуючих типів даних, таких як AccountId (для відображення адреси користувача мережі блокчейн) та Balance (для відображення кількості коштів на балансі користувача) було додано такі типи даних:

- PropId — для відображення id нерухомості;

- PricePerMth — для відображення ціни оренди нерухомості за місяць, виставленої власником нерухомості;
- Duration — для відображення терміну оренди.

Проектування подій

Смарт-контракт має можливість публікувати події, які трапилися в рамках смарт-контракту, в мережі блокчейн для інформування користувачів, що певна подія відбулася та постачати додаткову інформацію про ці події.

Було спроектовано такі події:

- PropertyApproved. Ця подія збуджується функцією approve_property, якщо власник смарт-контракту заносить власність до реєстру. Подія містить id нерухомості та id власника як додаткову інформацію.
- TenantApproved. Ця подія збуджується функцією approve_tenant, якщо власник нерухомості погоджується здавати свою власність в оренду певному орендарю. Подія містить id нерухомості та id орендаря як додаткову інформацію.
- PriceSet. Ця подія збуджується функцією set_price, коли власник нерухомості виставляє ціну за місяць за оренду своєї певної нерухомості. Подія містить id нерухомості та ціну за місяць оренди як додаткову інформацію.
- RentPaid. Ця подія збуджується функцією pay_rent, коли вже раніше погоджений власником нерухомості орендар сплачує ренту. Подія містить id нерухомості, id орендаря, ціну за місяць яку було сплачено, початкову часову мітку оренди та довжину оренди як додаткову інформацію.
- PropertyRemoved. Ця подія збуджується функцією remove_property, якщо власник смарт-контракту або власник нерухомості вирішує видалити нерухомість зі смарт-контракту. Подія містить id нерухомості та id власника як додаткову інформацію.

Написання інтерфейсу смарт-контракту

Інтерфейс смарт-контракту складається з наступних сигнатур функцій:

- new() - функція для створення та ініціалізації смарт-контракту;
- get_landlord(property) — функція для отримання id аккаунта власника нерухомості за id нерухомості;
- get_price(property) — функція для отримання ціни оренди за місяць за id нерухомості;
- get_tenant(property) — функція для отримання id орендаря за id нерухомості;
- get_timespan(property, tenant) — функція для отримання моменту оплати оренди та терміну оренди за id нерухомості та id орендаря даної нерухомості;
- approve_property(landlord) — функції для реєстрації певної власності. Для реалізації даної функції для реального проекту також необхідно зберігати, наприклад, хеш документу про володіння даною нерухомістю даним власником. Оскільки реєстрація власності відбувається через власника смарт-контракту, це є його обов’язком перевірити достовірність існування даної нерухомості у власності цього власника.
- remove_property(property) — вилучення власності з реєстрів смарт-контракту.
- set_price(property, price) — встановлення вартості оренди нерухомості за місяць.
- approve_tenant(property, tenant) — фактично функція підписування смарт-контракту зі сторони власника нерухомості. Орендар зі своєї сторони це повністю затверджує сплачуючи ренту.
- pay_rent(property) — сплачування ренти орендарем.

Визначення структур даних, необхідних для відображення внутрішнього стану смарт-контракту

Для опису внутрішнього стану смарт-контракту було використано наступні структури даних:

- Прості змінні:
  - Змінна owner використовується для зберігання id власника смарт-контракту для подальшої його ідентифікування в функціях.
  - Змінна last_property_id використовується для відслідковування останнього присвоєного id нерухомості задля збереження унікальності кожного id нерухомості.
- Структура даних відображення (Mappin):
  - landlords: Mapping<PropId, AccountId> - відображення нерухомостей на їх власників. Заповнюється власником смарт-контракту при виклику функції approve_property.
  - tenants: Mapping<PropId, AccountId> - відображення нерухомостей на їх орендарів. Заповнюється власником нерухомості при виклику функції approve_tenant.
  - prices: Mapping<PropId, PricePerMth> - відображення нерухомостей на ціну їх оренди в місяць. Заповнюється власником нерухомості при виклику функції set_price.
  - timespans: Mapping<(PropId, AccountId), (Timestamp, Duration)> - відображення id нерухомості та id орендаря на часову мітку початку оренди та тривалість оренди. Заповнюється орендарем під час виклику функції pay_rent.

Проектування помилок

В разі виникнення виключних ситуацій, таких як недостатній рівень доступу користувача для виконання певної ідеї, або недостатньої суми для сплати ренти, функціями смарт-контракту будуть повернені помилки, а всі зміни стану смарт-контракту, зроблені під час цієї функції, будуть відкликані.

Смарт-контрактом можуть бути повернені такі помилки:

- NotEnoughRights — недостатньо прав для виконання операції. Може бути повернута функціями:
  - approve_property, якщо акаунт, який викликає функцію, не є акаунтом власника смарт-контракту;
  - remove_property, якщо акаунт, який викликає функцію, не є акаунтом власника смарт-контракту або власника нерухомості;
  - set_price, якщо акаунт, який викликає функцію, не є акаунтом власника нерухомості, ціна за яку виставляється;
  - approve_tenant, якщо акаунт, який викликає функцію, не є акаунтом власника нерухомості, яку орендує орендар.
- PropertyDoesntExist — нерухомості з таким id не існує. Може бути повернута функціями:
  - get_landlord;
  - remove_property;
  - set_price;
  - approve_tenant;
  - pay_rent.
- UnsufficientRent — сума передана до смарт-контракту недостатня для сплати ренти. Може бути повернута функцією pay_rent.
- NotApprovedTenant - користувач не є підтвердженим орендарем даної нерухомості, і, відповідно, не може сплачувати ренту. Може бути повернута функцією pay_rent.
- NoApprovedTenant — дана нерухомість не має підтверджених орендарів. Може бути повернута функцією pay_rent та get_tenant.
- PriceIsnSet — ціна для даної нерухомості не встановлена, відповідно орендар не може оплатити ренту. Може бути повернута функцією pay_rent та get_price.
- FailedTransferFunds — помилка у переслані криптовалюти. Може бути повернута функцією pay_rent.

- TimespanDoesntExist - часова мітка не існує. Може бути повернута функцією get_timespan.

Опис поведінки функції за допомогою керованої тестами розробки та імплементація функцій

Керована тестами розробка — це метод розробки програмного забезпечення, при якій розробка відбувається короткими ітераціями шляхом попереднього написання тестів, які визначають поведінку функцій. Мета — розробити імплементацію цих функцій, яка пройде тести.

Отже, для вищеописаного інтерфейсу були написані тести та після цього була написана імплементація функцій, яка б задовольняла дані тести.

Процес був організований таким чином, що спочатку пишеться один тест для однієї окремої функції і після цього пишеться імплементація функції. Після цього процес повторюється необхідну кількість разів.

Був протестований конструктор смарт-контракту (функція new()) та функції, які змінюють внутрішній стан смарт-контракту, тобто функції-мутатори.

Були протестовані наступні функції:

- new
- approve_property
- approve_tenant
- set_price
- pay_rent

Розгортання смарт-контракту на локальному блокчейні

Для того щоб розгорнути смарт-контракт на локальному блокчейні, необхідно:

- скомпілювати код смарт-контракту в веб-ассемблер та отримати файл веб-ассемблера з метаданими (з файловим розширенням .contract);

- скомпілювати та розгорнути локальний блокчейн substrate-contract-node, взятий із офіційного репозиторія: https://github.com/paritytech/substrate-contracts-node;
- підключитися з графічного браузерного інтерфейсу для тестування смарт-контрактів https://paritytech.github.io/contracts-ui до свого локального блокчейну;
- загрузити код смарт-контракту (скомпільований на першому етапі в файл з розширенням .contract);
- створити екземпляр смарт-контракту на блокчейні.

Мануальне тестування смарт-контракту

Після розгортки смарт-контракту на блокчейні, він був протестований ручною перевіркою програмного забезпечення. Для цього на тестовому блокчейні є тестові акаунти, з яких можна викликати функції та імітувати виклики функцій смарт-контракту.

Повний код розробленого смарт-контракту за посиланням: https://github.com/DanylAnikushyn/land_smart_contract/blob/master/lib.rs

ВИСНОВКИ

Технологія блокчейн в останні роки досягла свого розквіту. З розвитком смарт-контрактів, децентралізованих застосунків та розширенням своєї пропускної здатності він увійшов в багато перспективних галузей - від фінансів до геймінгу. Разом з тим, з розвитком таких технологій та нововведень, як інтернет речей (IoT), оракли та рікардіанські контракти, стало можливим доєднувати до блокчейну, до цього світу суто децентралізованого та віртуального, інформацію з реального, фізичного світу.

Таким чином стала можливою відслідковування та передача права власності активів фізичного світу, від товарів в ланцюгу поставок до нерухомості.

В рамках даної роботи було досліджено можливість надання нерухомості в оренду за допомогою смарт-контрактів та розроблено смарт-контракт, який виступав би таким сервісом. Смарт-контракт було розроблено використовуючи предметно-орієнтовану мову програмування ink! та інфраструктуру з документацією Parity Technologies. Як теоретичне підґрунтя було взято модель смарт-контракту як машини зі скінченною кількістю станів. Дана модель була використана для проектування смарт-контракту перед його розробкою. Було виділено два можливих рішення проблеми захисту даного смарт-контракту від зловмисних дій його учасників та реалізоване рішення використовування рікардіанського контракту.

Результати даної роботи можна використовувати на практиці при створенні подібних застосунків для керування фізичними активами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Satoshi Nakamoto Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.
2. Nick Szabo Formalizing and Securing Relationships on Public Networks. *First Monday*, *2*. 1998. №. 9. doi: https://doi.org/10.5210/fm.v2i9.548
3. Anastasia Mavridou, Aron Laszka Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach. 2017. doi: https://doi.org/10.48550/arXiv.1711.09327
4. Christoph Jentzsch, Decentralized Autonomous Organization To Automate Governance . 2017.
5. Vitalik Buterin, Crypto Cities. URL: https://vitalik.ca/general/2021/10/31/cities.html
6. Thomas Hepp, Patrick Wortner, Alexander Schönhals, Bela Gipp Securing Physical Assets on the Blockchain. *CryBlock'18: Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*. 2018. P. 60-65. doi: https://doi.org/10.1145/3211933.3211944
7. Thomas Hepp, Alexander Schoenhals, Christopher Gondek, Bela Gipp, OriginStamp, A blockchain-backed system for decentralized trusted timestamping. 2018. doi: https://doi.org /10.1515/itit-2018-0020
8. Mattereum, Smart Contracts. Real Property. 2020. URL: https://mattereum.com/wp-content/uploads/2020/02/mattereum_workingpaper.pdf
9. Chainlink. What is Blockchain Oracle? URL: https://chain.link/education/blockchain-oracles
10. ink! URL: https://paritytech.github.io/ink
11. Wasm URL: https://wiki.polkadot.network/docs/learn-wasm
12. Cointelegraph. A Beginners Guide to Understand the Layers of Blockchain Technology URL: https://cointelegraph.com/blockchain-for-beginners/a-beginners-guide-to-understanding-the-layers-of-blockchain-technology
13. Ethereum. Tokens URL: https://ethereum.org/en/developers/docs/standards/tokens/

14. Glossary of Blockchain Terms URL: https://blockchaintrainingalliance.com/pages/glossary-of-blockchain-terms
15. Consensus Algorithms in Blockchain URL: https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/
16. Luu, L., Chu, D.H., Olickel, H., Saxena, P., Hobor, A., Making smart contracts smarter. *In: Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2016. P 254-269.doi: https://doi.org/10.1145/2976749.2978309
17. Cryptopedia. What was the DAO? URL: https://www.gemini.com/cryptopedia/the-dao-hack-makerdao
18. TezTalks Live №23 – Hillary Shieve. URL: https://www.youtube.com/watch?v=uqXW0Kt-RzU
19. CityDAO. URL: https://www.citydao.io/
20. DeFi Rate. Collateralized Loans in DeFi. https://defirate.com/collateralized-loan/

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

студента 4 курсу спеціальності 051 «Економіка»

освітньої програми «Економічна кібернетика»

Анікушина Данила Романовича

1. Тема роботи: «Керування фізичними активами за допомогою блокчейну».

2. Термін завершення роботи: до 20 травня 2021 року.

3. Об’єктом дослідження є дизайн смарт-контракту для керування фізичними активами.

4. Предмет дослідження: методичне забезпечення та практичний інструментарій щодо розробки смарт-контрактів для керування фізичними активами..

5. Мета роботи полягає в визначенні оптимальних стратегій керування фізичними активами за допомогою технології блокчейну.

6. Завдання дослідження:

6.1. аналіз сутності блокчейн технологій;

6.2. дослідження існуючих методів керування фізичними активами;

6.3. запропонувати методологію дизайну смарт-контрактів щодо керування фізичними активами;

6.4. розробка смарт-контракту для керування нерухомістю.

Науковий керівник: к. е. н., доцент. Шпирко Віктор Васильович

_____ Шпирко В.В.

Студент: Анікушин Данило Романович

_____ Анікушин Д.Р.

Затверджено на засіданні кафедри економічної кібернетики

протокол № від р.

Календарний план виконання кваліфікаційної роботи бакалавра

№	Етапи роботи	Термін виконання	Відмітка керівника про виконання
1	Вибір теми кваліфікаційної роботи бакалавра	30.11.2021	
2	Розробка та затвердження завдання кваліфікаційної роботи бакалавра	31.01.2022	
3	Написання першого розділу кваліфікаційної роботи бакалавра	31.03.2022	
4	Написання другого розділу кваліфікаційної роботи бакалавра	20.04.2022	
5	Написання третього розділу кваліфікаційної роботи бакалавра	20.05.2022	
6	Подання роботи до попереднього захисту	21.05.2022	

Науковий керівник: к. е. н., доцент. Шпирко Віктор Васильович

_____ Шпирко В.В.

Студент: Анікушин Данило Романович

______ Анікушин Д.Р.