Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
комп’ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Методи та засоби розробки смарт-контрактів на базі блокчейна Cardano

Виконав: студент 6 курсу, групи Сім-61
спеціальності 123 Комп’ютерна інженерія
(шифр і назва спеціальності)

(підпис) Соленко С.В. (прізвище та ініціали)

Керівник (підпис) Жаровський Р.О. (прізвище та ініціали)

Нормоконтроль (підпис) Тиш Є.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)

Рецензент (підпис) Приймак М.В (прізвище та ініціали)

Тернопіль 2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп’ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Осухівська Г.М.
(підпис) (прізвище та ініціали)
« 1 » листопада 2021 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 123 Комп’ютерна інженерія
(шифр і назва спеціальності)

студенту Соленку Сергію Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи та засоби розробки смарт-контрактів на базі блокчейна Cardano

Керівник роботи Жаровський Руслан Олегович, канд. техн. наук
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *28* » *жовтня* 2021 року № *4/7-916*

2. Термін подання студентом завершеної роботи 22.12.2021 р

3. Вихідні дані до роботи мова програмування Haskell , платформа написання смарт-контрактів Plutus Playground, блокчейн Cardano

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналіз існуючих рішень та обґрунтування теми магістерської роботи
2. Технологія блокчейн. Застосування платформи Cardano для смарт-контрактів
3. Практична реалізація смарт-контракту
4. Охорона праці та безпека в надзвичайних ситуаціях
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)
1. Тема, мета, задачі, об’єкт, предмет дослідження.
2. Актуальність.
3. Сфери застосування смарт-контрактів.
4. Порівняльний аналіз технологій.
5. Архітектура Cardano.
6. Архітектура Plutus.
7. Симуляція роботи смарт-контракту в Plutus Playground.
8. Результати симуляції смарт-контракту.
9. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., зав. каф. КС		
Безпека в надзвичайних	Стадник І.Я., проф. каф. ОХ		
ситуаціях			

7. Дата видачі завдання 1 листопада 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.11.2021 р	Виконано
2	Аналіз завдання	11.11.2021 р.	Виконано
3	Виконання розділу 1	14.11.2021 р.	Виконано
4	Виконання розділу 2	17.11.2021 р.	Виконано
5	Виконання розділу 3	21.11.2021 р.	Виконано
6	Виконання розділу 4	25.11.2021 р.	Виконано
7	Оформлення пояснювальної записки	07.12.2021 р.	Виконано
8	Оформлення графічного та презентаційного матеріалу	10.12.2021 р.	Виконано
9	Перевірка роботи на антиплагіат	10.12.2021 р.	Виконано
10	Попередній захист	15.12.2021 р.	Виконано
11	Захист	22.12.2021 р.	Виконано

Студент _____ Соленко С.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Жаровський Р.О
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Методи та засоби розробки смарт-контрактів на базі блокчейна Cardano // Кваліфікаційна робота // Соленко Сергій Віталійович // ТНТУ, комп'ютерна інженерія, група СІм-61 // Тернопіль, 2021 // с. – 72, рис. – 32, табл. – 1, аркушів А1 – 8, додат. – 1 , бібліогр. – 25.

Ключові слова: blockchain, Cardano, смарт-контракт, Plutus, децентралізація, база даних, криптосистема

Мета роботи полягає у вивченні основних принципів роботи блокчейну. Дослідження архітектури та технологій блокчейну Cardano, можливостей смарт-контрактів. Методів та засобів реалізації смарт-контрактів на блокчейні Cardano.

Технологія Blockchain дуже стрімко розвивається. Вона надає можливість створення нових бізнес-моделей в різних галузях. Блокчейн уже використовується для того, щоб позбутися посередників.

Смарт-контракти — це програми, що зберігаються в блокчейні, які запускаються при дотриманні заздалегідь визначених умов. Зазвичай вони використовуються для автоматизації виконання угоди, щоб усі учасники могли бути негайно впевнені в результаті без участі посередника чи втрати часу. Вони також можуть автоматизувати робочий процес, запускаючи наступну дію, коли виконуються умови.

В процесі виконання кваліфікаційної роботи було досліджено:

- технологію Blockchain, структуру моделі та принцип роботи системи;
- переваги блокчейну Cardano;
- технологію смарт-контрактів;
- модель програмування Plutus.

Також було розроблено смарт-контракт для громадського фінансування (краудфандінгу).

ANNOTATION

Methods and tools for developing smart contracts based on the Cardano blockchain //Qualification work // Solenko Serhii Vitaliyovych / TNTU, computer engineering, group CIm -61 // Ternopil, 2021 // p p. - 72, fig. – 32, table. –1, Sheets A1 – 8, Add – 1, Ref.– 25.

Keywords: blockchain, Cardano, smart contract, Plutus, decentralization, database, cryptosystem.

The purpose of the work is to study the basic principles of blockchain. Research of Cardano blockchain architecture and technologies, possibilities of smart contracts. Methods and means of implementing smart contracts on the Cardano blockchain.

Blockchain technology is evolving very rapidly. It provides an opportunity to create new business models in various industries. Blockchain is already being used to get rid of middlemen.

Smart contracts are programs stored in a blockchain that run under predefined conditions. They are usually used to automate the execution of the transaction so that all participants can be immediately sure of the result without the involvement of a mediator or a waste of time. They can also automate the workflow by running the next action when conditions are met.

In the process of performing the qualification work were investigated:

− Blockchain technology, model structure and principle of system operation;

− Cardano blockchain benefits

− technology of smart contracts;

− Plutus programming model.

Also, a smart contract for public funding (crowdfunding) was developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

PoS (Proof of Stake) – метод захисту в криптовалютах, заснований на необхідності доказу зберігання певної кількості коштів на рахунку.

PoW (Proof of Work) – доказ виконаної роботи.

UTXO (Unspent Transaction Output) - вихід невитраченої транзакції.

Datum - поле даних на виходах сценарію в моделі Extended UTXO.

Redeemer - аргумент сценарію перевірки.

DApps (англ. decentralized applications) – децентралізовані сервер-клієнтські додатки.

Ada – криптовалюта блокчейну Cardano.

Lovelace - базова грошова одиниця криптовалюти Ada і становить 0,000001 Ada.

Стейблкойн (від англ. stablecoin) — загальна назва криптовалют, які прив'язані до запасів звичайних валют або фізичних товарів (золота, нафти).

ВСТУП

Актуальність теми. Впровадження технології блокчейн призвело до розвитку біткоіну та інших криптовалют. Але, цифрові валюти - це не єдине благо від такої інновації. Велику користь може отримати кожен, використовуючи смарт-контракти.

Технологію блокчейн можна застосовувати у багатьох сферах, наприклад: ідентифікація особистості, електронне голосування, громадське фінансування, покупка нерухомості, керування та юреспруденція.

Смарт-контракти вважаються однією з найбільш перспективних технологій, яка повинна змінити звичні способи ведення бізнесу. Практичний ефект від їх використання полягає в автоматизації фінансових і юридичних відносин між двома і більше суб'єктами.

В кваліфікаційній роботі досліджуються:

- архітектура та технології блокчейну Cardano;
- основні можливості смарт-контракту як способу обміну цифровими цінностями (даними) в технології блокчейн;
- можливості смарт-контрактів;
- методи та засоби реалізації смарт-контрактів на блокчейні Cardano.

Оновлення Alonzo яке відбулося в 2021 надало підтримку смарт-контрактів на блокчейні Cardano. Даній тематиці присвячено ряд робіт [1-5]. В даних роботах було досліджено основні принципи роботи смарт-контрактів на блокчейні Cardano. Однак на даний момент не реалізовані засоби для їх повсякденного використання в електронній комерції. Смарт-контракти на блокчейні Cardano мають велику перспективу стати одною з головних платформ для фінансових операцій та договорів. Тому актуальним є розробка засобів електроної комерції на базі Cardano смарт-контрактів.

Метою роботи є дослідження методів та засобів розробки смарт-контрактів на базі блокчейна Cardano. Використання смарт-контрактів в електроній комерції.

Для досягнення вказаної мети, в роботі були поставлені такі задачі:

- Дослідження технології блокчейн.
- Дослідження можливостей смарт-контрактів.
- Аналіз переваг та недоліків блокчейна Cardano.
- Перевірка на практиці, створення смарт-контракту для громадського фінансування.

Об’єкт дослідження: процес розробки смарт-контрактів на базі блокчейна Cardano.

Предмет дослідження: Методи та засоби розробки смарт-контрактів на базі блокчейна Cardano.

Методи дослідження. Для вирішення поставлених задач використано наступні методи: аналіз та узагальнення – при проведенні аналізу існуючих методів та технологій написання смарт-контрактів, аналітичний і порівняльний методи, табличний і графічний методи.

Наукова новизна одержаних результатів: Вперше було запропоновано використати технології блокчейна Cardano в електронній комерції, що дало можливість замінити існуючу практику паперових договорів водночас з забезпеченням безпеки та надійності децентралізованої мережі.

Удосконалено підхід до регулювання відносин між користувачами в децентралізованій платформі, шляхом використання найбільш ефективних методів смарт-контрактів на блокчейн-платформі Cardano.

Практичним значенням одержаних результатів є розробка смарт-контракту та програми Plutus для громадського фінансування (краудфандінг). Програма приймає кошти від учасників кампанії, дає проміжок часу власнику кампанії щоб зібрати кошти і якщо кошти не було зібрано то повертає їх учасникам кампанії.

Публікації. Результати дослідження апробовано на IX науково-технічна конференція «Інформаційні моделі, системи та технології» (8 – 9 грудня 2021 р.) Тернопільського національного технічного університету імені Івана Пулюя у вигляді 2 тез конференцій.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, 4 розділів, висновків, переліку посилань та додатків. Обсяг роботи: пояснювальна записка – 60 арк. формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ МАГІСТЕРСЬКОЇ РОБОТИ

1.1. Аналіз актуальності задачі

Смарт-контракт - це комп'ютерний аналог звичайних договорів, спеціальна програма (алгоритм) , що виконуються самостійно, умови угоди між покупцем і продавцем записуються безпосередньо в рядки коду. Смарт-контракт дозволяють здійснювати довірені транзакції та угоди між різними анонімними сторонами без необхідності центрального органу, правової системи або зовнішнього механізму забезпечення виконання.

Предметом договору може бути тільки об'єкт, що знаходиться всередині середовища існування смарт-контракту, або ж повинен забезпечуватися безперешкодний, прямий доступ розумного контракту до предмету договору без участі людини.

Умови смарт-контракту повинні мати повний математичний опис, який можливо запрограмувати в середовищі існування розумного контракту. Саме в умовах описується логіка виконання пунктів предмета договору.

Подібно до традиційних контрактів, смарт-контракти визначають правила та штрафи навколо угоди та автоматично забезпечують виконання цих зобов’язань. Хоча вони можуть працювати незалежно, багато розумних контрактів також можна реалізувати разом.

Смарт-контракти є одним із ключових компонентів багатьох екосистем на основі блокчейну та особливо важливим елементом багатьох блокчейнів, орієнтованих на застосування, таких як Ethereum та Cardano. Ці цифрові контракти є надійними, автономними, децентралізованими та прозорими — і зазвичай незворотні й незмінні після розгортання.

Переваги смарт-контракту включають зменшення або навіть усунення потреби в посередниках та забезпеченні виконання контракту в угоді чи

транзакції. Це тому, що в смарт-контракті код визначає механізми транзакції і є остаточним арбітром умов. З цієї причини смарт-контракти стали будівельними блоками цілої екосистеми децентралізованих додатків (dApps) і є основним центром розробки блокчейну в цілому.

Прообразом смарт-контрактів є звичайні паперові договори. Після складання такі договори зазвичай вручну підписуються, після чого учасники особисто виконують всі їх умови.

У табл.1.1 наведено порівняння смарт-контрактів зі звичайними, «паперовими» договорами, які використовуються повсюдно:

Таблиця 1.1

Порівняння смарт-контрактів зі стандартними договорами

Характеристика	Смарт-контракт	Стандартний паперовий договір
Носій інформації	комп'ютерний алгоритм на платформі блокчейн	папір
На чому базується документ	код програми	норми права
Можливість змінити умови	не можливо змінити умови чинного контракту	можна переписати, змінити.
Складність в складанні контракту	висока, часто потрібен розробник	середня, іноді потрібен юрист
Виконання умов договору	виконуються автоматично усіма учасниками	можуть бути не виконані сторонами
Застосування покарання	автоматично при настанні певних умов	зазвичай вирішуються в суді
Наявність посередників	угоди проводяться без посередників	зазвичай потрібна допомога юриста
Місцезнаходження сторін:	зазвичай без особистої пристутності осіб	зазвичай потрібна фізична пристуність представників сторін
Ризик шахрайських операцій	практично виключений	невеликий

На рис. 1.1 зображено графічне пояснення смарт-контракту.

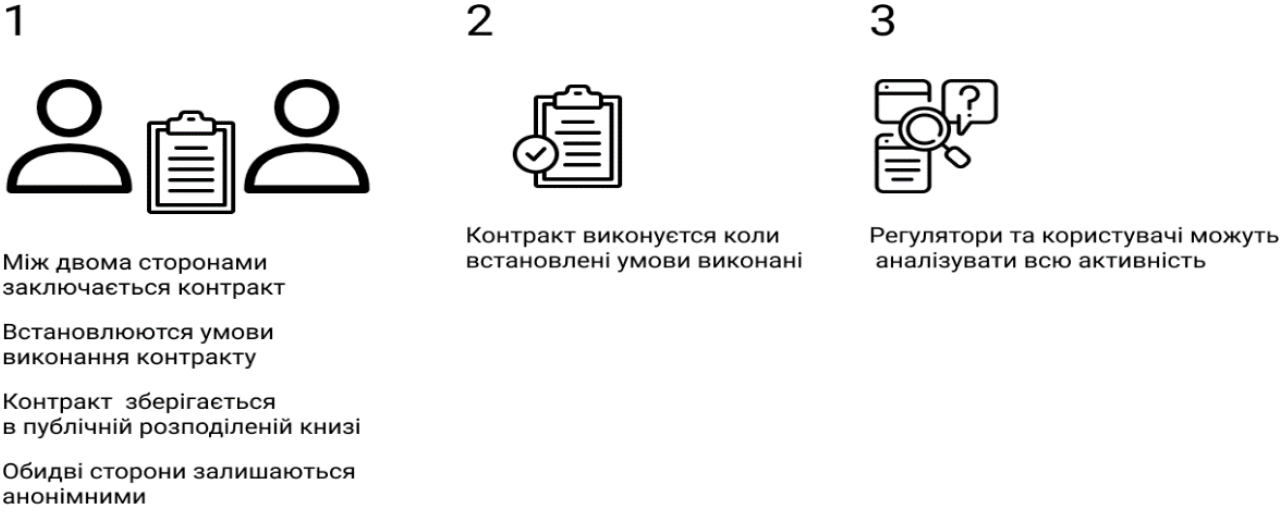

Рис. 1.1. Графічне пояснення смарт-контракту

Найбільш очевидним розвитком цих процесів могли б стати технології, які б автоматизувати ці процеси. Контракт, в такому випадку, почав би виконуватися автоматично, без фізичної присутності учасників. Чим більше блокчейн технології розвиваються тим краще смарт-контракти підходять для вирішення цих задач.

Приклади використання смарт-контрактів зображено на рис 1.2.

Рис. 1.2. Приклади використання смарт-контрактів

Одним з випадків використання смарт-контрактів є цифрова ідентифікація. Індивідуальна ідентичність є одним із найбільших активів для цієї особи. Він містить репутацію, дані та цифрові активи. Цифрова ідентичність, якщо її правильно використовувати, може дати людині нові

можливості. Крім того, цифрова ідентифікація може також допомогти захистити особистість від контрагентів і дозволити йому поділитися нею з компаніями якими вона бажає.

Розумні контракти також можуть допомогти покращити фінансові послуги, зокрема іпотеку та позики. Для цього він може з'єднати сторони та забезпечити завершення всього процесу без перебоїв. Крім того, це також забезпечує безпомилковий процес. Наприклад, смарт-контракт, створений для обробки іпотеки, може керувати нею, відстежуючи платежі та звільняючи майно, коли весь кредит буде погашено.

Управління ланцюгом поставок — це чудовий варіант використання блокчейн-розумних контрактів. Використовуючи розумні контракти, ланцюжок поставок можна багаторазово покращити. Наприклад, його можна використовувати для відстеження товарів у ланцюжку поставок з повною видимістю та прозорістю. Підприємство може використовувати ланцюжки поставок на основі смарт-контрактів і покращити відстеження запасів до детального рівня.

Урядова система голосування. Розумні контракти забезпечують безпечне середовище, що робить систему голосування менш сприйнятливою до маніпуляцій. Голосування за допомогою смарт-контрактів буде захищено реєстром, який надзвичайно важко розшифрувати.

Використання смарт-контрактів у фінансах. Додатки для децентралізованих фінансів (DeFi) представляють собою чудову альтернативу традиційним фінансовим послугам, і їх популярність зростає завдяки надійним, незмінним і прозорим характеристикам блокчейну та технології смарт-контрактів. DeFi dApps надають паралельні послуги індустрії банківських та фінансових послуг — наприклад, кредитування, позики, торгівля та низку інших фінансових послуг — разом із абсолютно новими типами продуктів і децентралізованими бізнес-моделями, які можуть запропонувати значну користь і корисність для користувачів. Завдяки підвищеній прозорості, що забезпечується смарт-контрактами (разом з

функціоналом 24/7 та зниженою вартістю), DApps мають потенціал для зниження бар’єрів для входу на арену фінансових послуг для людей у всьому світі.

Смарт-контракти та блокчейн в юридичній галузі. Можливо, одним із найперспективніших прикладів використання смарт-контрактів у реальному світі є їх потенціал функціонувати як юридично обов’язкові контракти, які визначають більшість сучасних бізнес-замовлень. Технології стимулюють інновації в юридичній галузі, нещодавно з появою електронних підписів для обов’язкових юридичних угод. Смарт-контракти є ще однією новою подією в цьому просторі, і незабаром вони можуть стати варіантом для сторін юридичних угод, що потенційно знизить витрати, пов’язані з використанням юристів та інших посередників.

Важливо відзначити, що для смарт-контракту необхідна наявність декількох обов’язкових елементів:

− для укладення смарт-контрактів необхідним є децентралізоване середовище, в яке будуть записуватися смарт-контракти;

− сам предмет договору та наявність необхідних для його виконання інструментів;

− конкретно описані умови його виконання, які учасники підтверджують одночасно.

1.2. Переваги та недоліки смарт контрактів

Смарт-контракти є самовиконувальними, надійними та не вимагають дій чи присутності третіх сторін. Код смарт-контракту зберігається і поширюється в децентралізованій мережі блокчейнів.

На рис. 1.3 зображено переваги смарт-контрактів.

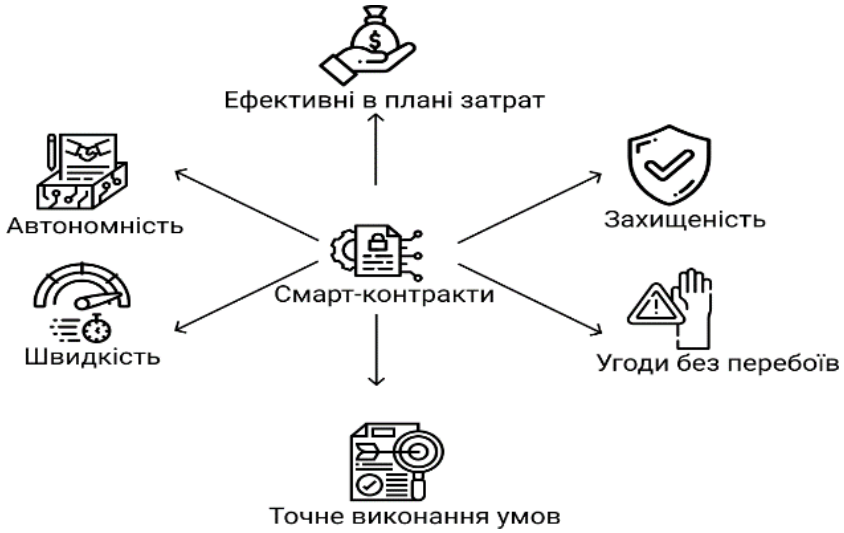

Рис. 1.3. Переваги смарт-контрактів

Переваги смарт-контрактів перед паперовими аналогами:

– Автономність. Учасники самостійно укладають усі договори – більше немає посередників для підтвердження або посвідчення договору.

– Захищеність. Технології блокчейну відповідають за збереження ваших документів. Практично неможливими взлом та підміна вашого смарт-коду завдяки децентралізації блокчейну.

– Швидкість. Робота з паперовими документами може займати багато часу. Автономні смарт-контракти виконуються набагато швидше, ніж старомодний традиційний підхід. Оскільки всі параметри вже визначені в смарт-контрактах, їм потрібно лише зіставити їх перед тим, як він почне виконуватися.

– Ефективні в плані затрат. Смарт-контракти позволяють складати договора без посередника, що заощаджує гроші та час.

– Угоди без перебоїв. Блокчейн забезпечує безперебійну роботу в цифровій мережі.

– Точне виконання умов. Автоматизовані контракти допомагають уникати помилок, які виникають при ручному заповненні форм документів і виключають людський фактор при проведенні транзакцій за договором.

Недоліки смарт-контрактів**:**

- можуть бути помилки і вразливі місця в програмному коді смарт-контракту;
- складність побудови алгоритму коду;
- є ймовірність втрати ключів доступу до смарт-контракту;
- якщо були непередбачені якісь умови змінити угоду неможливо;
- використання смарт-контрактів немає законодавчої бази.

1.3. Аналіз існуючих рішень

Ethereum (ETH) — це блокчейн-платформа для смарт-контрактів, створена складом криптографів у 2015 році. Сьогодні Ethereum в основному використовується для створення та розміщення децентралізованих фінансових програм та стейблкоінів[8].

Найзначніший внесок, який Ethereum зробив у блокчейн, — це винахід розумних контрактів.

Переваги Ethereum:

- найбільша блокчейн-платформа для смарт-контрактів у бізнесі;
- Ethereum є захищеним рішенням DeFi і NFT;
- першопроходець в смарт-контракт технології;
- майбутнє оновлення ETH 2.0 підвищить продуктивність мережі;

Недоліки Ethereum:

- поточні обмеження швидкості призводять до частого перевантаження мережі
- комісія Ethereum часто є занадто високою для розробників;
- високі комісії роблять транзакції DEX і DeFi дорогими;
- розвиток Ethereum часто відбувається неорганізовано і повільно;
- ETH 1.0 використовує PoW і сприяє зміні клімату;
- одноріневий протокол Ethereum, який виконує одночасно обчислення (розумні контракти) і розрахунки (передача токенів).

Cardano — це інноваційна блокчейн-мережа з доказом частки (PoS), яка розвивається в платформу розробки децентралізованих додатків (DApp) з реєстром багатьох активів і розумними контрактами, які можна перевірити. Побудований із суворістю формальних методів розробки з високою надійністю, Cardano прагне досягти масштабованості, сумісності та стійкості, необхідних для реальних додатків. Cardano розроблено, щоб бути платформою вибору для великомасштабних, критично важливих DApps, які ляжуть в основі економіки майбутнього[9].

На основі рецензованих наукових досліджень, Cardano має дух відкритості та прозорості. Усі дослідження та технічні специфікації, що лежать в основі Cardano, опубліковані для всіх, а вся діяльність з розробки Cardano доступна для громадськості. Cardano розроблено глобальною командою експертів, які є лідерами в різних дисциплінах, від розподілених систем до мов програмування та теорії ігор, і спільно розроблено IOHK та партнерами. IOHK розробляє технологію, Cardano Foundation відповідає за нагляд за розробкою та просуванням Cardano, а Emurgo забезпечує комерційне впровадження[9]. Коли мережа буде повністю децентралізована, вона буде належати спільноті, і саме спільнота вирішуватиме її майбутнє за допомогою розширених функцій управління.

Академічні дослідження — формальні методи, такі як математичні специфікації, тести на основі властивостей і докази, є найкращим способом забезпечення високоякісних програмних систем і надати користувачам впевненості в управлінні цифровими фондами. Cardano було побудовано з використанням формальних методів для досягнення надійних гарантій функціональної коректності основних компонентів системи. Усі дослідження та технічні специфікації, що лежать в основі Cardano, є загальнодоступними, а вся діяльність з розробки Cardano публікується в Інтернеті.

Дизайн системи — Cardano написаний на Haskell, безпечній функціональній мові програмування, яка заохочує будувати систему з використанням чистих функцій, що призводить до дизайну, де компоненти

зручно тестувати окремо. Крім того, розширені функції Haskell дозволяють нам використовувати цілий ряд потужних методів для забезпечення коректності коду, таких як базування реалізації на формальних і виконуваних специфікаціях, обширне тестування на основі властивостей і запуск тестів під час моделювання.

Безпека — Ouroboros (протокол Cardano proof-of-stake) встановлює суворі гарантії безпеки; він був представлений з кількома рецензованими доповідями, представленими на конференціях найвищого рівня та публікаціями в області кібербезпеки та криптографії.

Споживання електроенергії — Cardano — це блокчейн із доказом частки (PoS). На відміну від блокчейнів, що підтверджують роботу (PoW), Cardano вимагає набагато менше енергії та обчислювальної потужності. Мережа біткойн розвивається завдяки комп'ютерам, які виконують все більш енергоємні обчислення – доказ роботи – що є нежиттєздатним у довгостроковій перспективі. У Кембриджському університеті є онлайн-інструмент, який показує, що комп'ютери, що живлять біткойн, щороку споживають вдвічі більше енергії, ніж Швейцарія.

Безперебійне оновлення – традиційно, блокчейни оновлюються за допомогою хардфорків. При проведенні хард-форку поточний протокол припиняє роботу, впроваджуються нові правила та зміни, а ланцюжок перезапускається – зі стиранням його попередньої історії. Cardano по-різному обробляє хард-форки. Замість радикальних змін технологія комбінатора хардфорка Cardano забезпечує плавний перехід на новий протокол, зберігаючи історію попередніх блоків і не викликаючи жодних збоїв для кінцевих користувачів.

Децентралізація — Cardano підтримується майже 3000 розподіленими пулами активів, якими керує спільнота. Усі блоки та транзакції перевіряються учасниками мережі, не покладаючись на централізовані органи влади[9].

Функціональне середовище для бізнес-використання — Cardano створює основу для глобальних децентралізованих фінансів для розробки ряду

DApps, які можуть працювати з використанням функціональних і специфічних для домену смарт-контрактів, надаючи токени з кількома активами для будь-яких потреб.

Окремі рівні Cardano дозволяють розумним контрактам і децентралізованим додаткам існувати незалежно від транзакцій токенів ADA. Цей поділ має дозволити Cardano масштабуватися, залишатися недорогим для розробників і користувачів, а також взаємодіяти з іншими платформами.

1.4. Висновки до розділу

Підсумовуючи, розумні контракти є незмінними, оскільки контракт не можна змінити, вони розповсюджуються та захищені від несанкціонованого доступу, швидкі та економічно ефективні, оскільки немає посередника, що економить гроші та час, і є безпечним завдяки шифруванню.

Вважаю, що подальше використання смарт-контрактів набуде особливої актуальності, оскільки, як зазначено вище, даний елемент системи електронного документообігу має ряд переваг над традиційним, паперовим методом.

Cardano розроблений як блокчейн третього покоління з метою вирішити проблеми масштабованості першого покоління (наприклад, Bitcoin) і другого покоління (наприклад, Ethereum).

Блокчейни попереднього покоління страждають від вузьких місць, які в корені обмежують об'єм пропускної здатності, з якою вони можуть впоратися. Це робить їх неефективним рішенням для масового використання в усьому світі.

При аналізі існуючих рішень було досягнуто висновку що написання смарт-контрактів на базі блокчейна Cardano є технологічно і економічно хорошим рішенням для використання в електронній комерції.

РОЗДІЛ 2
ТЕХНОЛОГІЯ БЛОКЧЕЙН. ЗАСТОСУВАННЯ ПЛАТФОРМИ CARDANO ДЛЯ СМАРТ-КОНТРАКТІВ

2.1. Технологія блокчейн

Блокчейн – це тип бази даних або реєстру, який дублюється та розповсюджується всім учасникам мережі блокчейн. Він складається з набору взаємопов'язаних вузлів, які зберігають дані або цінні елементи в блоках. Ці блоки перевіряються транзакціями і пов'язуються один з одним у хронологічному порядку в ланцюжку. Деталі цих транзакцій постійно вписуються в блок і не можуть бути змінені.

Технологія блокчейн, інакше відома як технологія розподіленої книги (DLT), забезпечує децентралізовану та доступну структуру даних для різних записів. Такі записи можуть включати інформацію про фінансові платежі та транзакції, а також інші типи інформації – від комерції до записів Інтернету речей (IoT).

Оскільки блокчейн зберігає дані децентралізованим способом, він не залежить від централізованих, контролюючих організацій або посередників. Це забезпечує підвищену прозорість зберігання даних та управління ними. Важливою особливістю блокчейну є те, що він зберігає записи незмінно, що означає, що їх не можна змінити, підробити або видалити, оскільки це порушить ланцюжок записів.

Транзакції та блоки є головним аспектом будь-якого блокчейну. Транзакції – це атомарні операції, які змінюють стан книги. Будь-який користувач або програма може вільно приєднатися до мережі та подавати транзакції, які пізніше транслюються на інші вузли для перевірки та виконання. Ці транзакції можуть виконуватися різними способами залежно від блокчейна, який вони запускають. Наприклад, біткойн в першу чергу дозволяє здійснювати транзакції для переказу коштів між рахунками. Він підтримує

сценарії, але обмежений. З іншого боку, Ethereum та Cardano дозволяє переказувати кошти та запускати програми.

Додавання блоків до книги вимагає координації між усіма Майнерами в мережі. Що ще гірше, пам’ятайте, що Майнери також змагаються один з одним за гонорари та винагороди. Якби будь-який Майнер міг додавати блоки на свій розсуд, значення Blockchain дорівнювало б 0. У реальному житті будь-який Blockchain використовує розподілений протокол, відомий як протокол консенсусу, який керує Майнерами та визначає, як і коли вони можуть надсилати блоки в реєстр. Біткойн представив варіант протоколу під назвою доказ роботи (Proof of Work), але є багато інших. Наприклад, Ethereum був запущений з точною реалізацією, але в перспективі він перейде на доказ частки "Proof of Stake" з новим зведеним пакетом під назвою Ethereum 2.0.

Доказ роботи(Proof of Work) покладається на комп’ютерну обробку для вирішення складних математичних задач. Коли Майнер генерує блок з транзакціями, він також включає хеш, обчислений з двійкового представлення транзакцій, випадкове значення одноразового значення (набір байтів) і хеш попереднього блоку. Це робить блокчейн стійким до змін. Якщо Майнер хоче змінити транзакцію в одному блоці, він також повинен відновити хеш у всіх попередніх блоках, що робить це практично неможливим[14].

Основною проблемою протоколу Proof of Work є споживання енергії, через що багато розробників почали шукати менш дорогі альтернативи. Так з’явився доказ частки(Proof of Stake). Для початку цей протокол вводить вхідний бар’єр для майнерів. Щоб стати майнерами або валідаторами, як їх називають у цьому протоколі, вони повинні володіти певною кількістю монет. З точки зору реалізації, цей протокол дозволяє валідаторам проштовхувати блоки в реєстр на основі рейтингу, визначеного кількістю монет в валідатора та репутацією з часом. На рейтинг можуть вплинути неправильні рішення або коли вузол не перевіряє транзакції (наприклад, він переходить в автономний режим), через що валідатор втрачає монети[14].

Блокчейни не лише забезпечують незмінну та безпечну базу даних, але й діють як функціональне середовище для транзакцій коштів, створення цифрових валют та обробки складних угод за допомогою цифрових угод (смарт-контрактів).

2.2. Блокчейн Cardano

Cardano – це блокчейн-проект із відкритим вихідним кодом, який розпочався у 2015 році для вирішення існуючих проблем блокчейну в розробці та розробці криптовалют. Він спрямований на забезпечення більш збалансованої та стійкої екосистеми, яка краще враховує потреби своїх користувачів, а також інших систем, які прагнуть інтеграції.

Перше покоління блокчейнів (наприклад, біткойн) пропонувало децентралізовані реєстри для безпечного переказу криптовалюти. Однак такі блокчейни не забезпечували функціональне середовище для комплексного розрахунку угод і розробки децентралізованих додатків (DApp). У міру розвитку технології блокчейн друге покоління (наприклад, Ethereum) надало більш вдосконалені рішення для написання та виконання смарт-контрактів, розробки додатків і створення різних типів токенів. З іншого боку, друге покоління блокчейнів часто стикається з проблемами з точки зору масштабованості.

Cardano задуманий як блокчейн третього покоління, оскільки він поєднує властивості попередніх поколінь і розвивається для задоволення всіх виникаючих потреб користувачів. Порівнюючи властивості блокчейну, слід враховувати багато аспектів. Таким чином, найкраще рішення повинно забезпечувати найвищу безпеку, масштабованість (пропускна здатність транзакцій, масштаб даних, пропускна здатність мережі) і функціональність (крім обробки транзакцій, блокчейн повинен забезпечувати всі засоби для розрахунків угоди). Крім того, важливо гарантувати, що технологія блокчейн

постійно розвивається з точки зору стійкості та взаємодіє з іншими блокчейнами та фінансовими установами.

Щоб задовольнити ці потреби, Cardano зосереджується на таких основних концепціях, як:

– Масштабованість – гарантує, що книга Cardano здатна обробляти велику кількість транзакцій, не впливаючи на продуктивність мережі. Масштабованість також забезпечує більш високу пропускну здатність, що дозволяє транзакціям переносити значну кількість допоміжних даних, якими можна легко керувати в мережі. Для цих потреб Cardano реалізує різні методи (наприклад, стиснення даних) і працює над впровадженням Hydra, яка дозволить функціонувати кількома бічними ланцюгами.

– Сумісність – забезпечує найбільш багатофункціональне середовище для фінансових, ділових або комерційних операцій, дозволяючи користувачам взаємодіяти не тільки з одним типом валюти, але й з кількома валютами в різних блокчейнах. Крім того, взаємодія з централізованими банківськими установами настільки ж важлива для забезпечення легітимності та зручності використання. Cardano розробляється для підтримки міжланцюгових переказів, кількох типів токенів і широко використовуваних мов смарт-контрактів.

– Стійкість – розробка блокчейну з доказом частки означає, що дуже важливо забезпечити самодостатність системи. Щоб стимулювати зростання та зрілість у справді децентралізований спосіб, Cardano створено, щоб дозволити спільноті підтримувати свій постійний розвиток, беручи участь, пропонуючи та впроваджуючи покращення системи. Щоб забезпечити стійкість, система казначейства контролюється спільнотою і постійно поповнюється з потенційних джерел, таких як щойно викарбувані монети, які затримуються в якості фінансування, відсоток винагород пулу ставок і комісія за транзакції.

2.3. Смарт-контракти Cardano

Смарт-контракт — це автоматизована цифрова угода, написана в коді, яка відстежує, перевіряє та виконує обов'язкові транзакції контракту між різними сторонами. Транзакції контракту автоматично виконуються кодом смарт-контракту, коли виконуються заздалегідь визначені умови. По суті, смарт-контракт – це коротка програма, вхідні та вихідні дані якої є транзакціями в блокчейні.

Смарт-контракти є самовиконувальними та надійними та не вимагають дій чи присутності третіх сторін. Код смарт-контракту зберігається і поширюється в децентралізованій мережі блокчейнів, що робить його прозорим і незворотним.

Cardano вводить підтримку смарт-контрактів у 2021 році. Будучи багатофункціональним середовищем, Cardano підтримуватиме розробку та розгортання смарт-контрактів за допомогою таких мов програмування, як:

- Plutus – спеціально створена платформа для розробки та виконання смарт-контрактів. Контракти Plutus складаються з частин, які виконуються на блокчейні (код у мережі), і частин, які виконуються на машині користувача (код поза ланцюгом або клієнт). Plutus спирається на дослідження сучасних мов, щоб забезпечити безпечне середовище програмування з повним стеком на основі Haskell, провідної функціональної мови програмування.
- Marlowe – доменно-специфічна мова (DSL) для написання та виконання фінансових контрактів, що дозволяє будувати контракти як візуально, так і в більш традиційному коді. Фінансові установи можуть використовувати його, наприклад, для розробки та впровадження спеціальних інструментів для своїх клієнтів і клієнтів. Сама мова Marlowe тепер вбудована як в JavaScript, так і в Haskell, що пропонує вибір редакторів залежно від уподобань і навичок розробників.

− Glow – нова мова (DSL) для розробки децентралізованих додатків (DApps) на блокчейні. Завдяки Glow користувачі можуть створювати безпечні DApps, гарантуючи безпечну роботу розумних контрактів у середовищі.

2.3. Модель програмування Plutus

Ми можемо розглядати блокчейн як книгу, що складається з послідовності транзакцій, що переміщують криптовалюту. У моделі UTxO, популяризованої біткоіном, кожна транзакція складається з набору вхідних даних (споживання валюти) та набору результатів (постачання валюти). Вхідні та вихідні дані транзакцій у бухгалтерській книзі з'єднані, щоб утворити спрямований ациклічний граф, що вкорінюється в блоці генезису блокчейна. У будь-який момент звисаючі (невитрачені) виходи на межі графіка утворюють множину, яка називається невитраченими транзакційними виходами (UTxO). Цей набір UTxO повністю визначає поточний стан ланцюжка і, зокрема, розподіл коштів.

Право власності на кошти передається опосередковано завдяки можливості витрачати певний (поки що) невитрачений вихід. Це вимагає володіння приватним криптографічним ключем, який відповідає відкритому ключу, що блокує вихід. Більш конкретно, транзакція, що містить вхідні дані, що витрачають заданий результат з набору UTxO, буде дозволена лише у тому випадку, якщо вона підписана криптографічно із закритим ключем, що відповідає відкритому ключу цього виходу. Транзакція допускається до ланцюжка (тобто вона дійсна тільки), якщо її підписи дозволяють усі витрати, визначені її вхідними даними, і якщо сума значень криптовалюти на виходах менша, ніж на входах (різниця називається комісія за транзакцію і використовується для оплати інфраструктури, наданої операторами блокчейну)

2.3.1. Архітектура Plutus. Plutus забезпечує підтримку написання сценаріїв, які працюють на блокчейні Cardano.

На рис. 2.1 міститься частковий графік UTxO, позначений (A), де входи червоного кольору, а виходи чорного кольору. Вхід, що підключається до виходу, символізує, що підпис у транзакції входу відповідає відкритому ключу виводу.

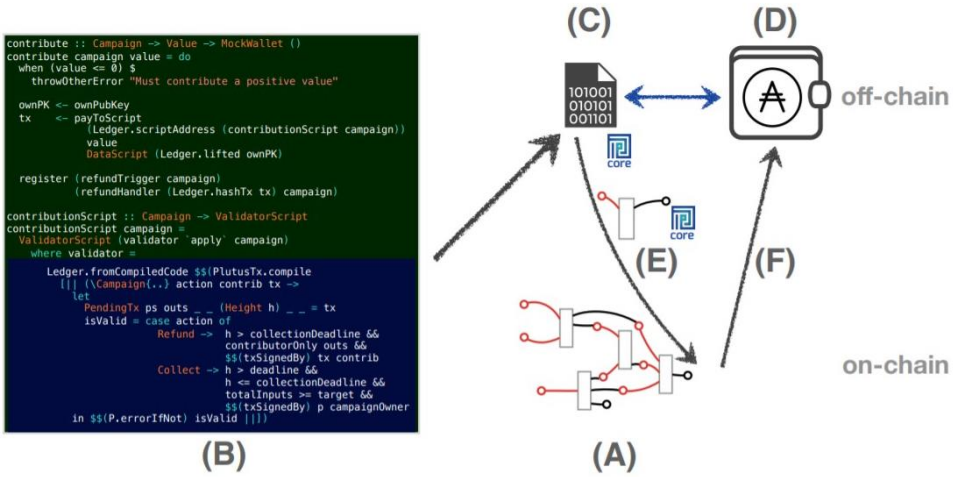

Рис. 2.1. Архітектура Plutus

Ліворуч на рис. 2.1 ми маємо Plutus контракт, позначений (B), який містить частину поза ланцюгом (зелений фон) та частину ланцюга (синій фон). Код як в ланцюзі, так і поза ланцюжком переплітаються в одній Haskell програмі, а код ланцюга вбудовується в код поза ланцюгом за допомогою шаблону Haskell.

Набір інструментів - ефективне розширення GHC за допомогою підтримки його плагінів - компілює контракт Plutus у позамережевий виконуваний файл (C), вбудовуючи скомпільований код ланцюга в нашу основну мову Plutus Core. Код поза ланцюгом виконується в контексті крипто-гаманця (D), який містить кошти для оплати транзакцій та криптографічні ключі для підписання транзакцій. Коли код контракту поза мережею подає

нову транзакцію (E) у блокчейн, він зазвичай блокує (деякі з) виходів з деяким своїм вбудованим кодом ланцюга Plutus Core[2].

Перевірка, а отже, включення цих та інших транзакцій у ланцюжок запускає події блокчейна (позначені стрілкою з міткою (F)), які спостерігаються гаманцем. Гаманець пересилає до цього коду події, які мають значення для коду контракту поза мережею, що, у свою чергу, призводить до нових транзакцій

Початкове виконання коду контракту поза мережею зазвичай ініціюється користувачем гаманця за допомогою кінцевих точок контракту (тобто функцій верхнього рівня), які стають доступними через елементи інтерфейсу гаманця. Отже, ми можемо розглядати поза-мережевий код контракту як форму плагіна для гаманців криптовалюти.

2.4. Розширена модель книги UTXO

Cardano (як і Bitcoin) використовує модель обліку невитрачених (U) транзакцій (TX), вихідних (O). У моделі UTXO транзакція має входи та виходи, де вхідні дані є невитраченими результатами попередніх транзакцій[1]. Як тільки вихідні дані використовуються як вхідні дані в транзакції, вони витрачаються і більше ніколи не можуть бути використані. Вихідні дані визначаються адресою (відкритим ключем або хешем відкритого ключа) і значенням (що складається з суми ada і необов’язкового, додаткового власного токена). Адреса виходу визначає, яким транзакціям дозволено «розблокувати» вихід і використовувати його як вхід. Транзакція повинна бути підписана власником приватного ключа, що відповідає адресі. Думайте про адресу як про «замок», який можна «розблокувати» лише за допомогою правильного «ключа» — правильного підпису.

Модель EUTXO розширює цю модель у двох напрямках:

1. Він узагальнює поняття «адреса» за допомогою аналогії із замком і ключем. Замість того, щоб обмежувати блокування відкритими ключами і

ключами для підписів, адреси в моделі EUTXO можуть містити довільну логіку у вигляді скриптів. Наприклад, коли вузол перевіряє транзакцію, вузол визначає, чи дозволено транзакції використовувати певний вихід як вхід. Транзакція шукає сценарій, наданий адресою виводу, і виконає сценарій, якщо транзакція може використовувати вихід як вхід.

2. Друга відмінність між UTXO та EUTXO полягає в тому, що виходи можуть переносити (майже) довільні дані на додаток до адреси та значення. Це робить сценарії набагато потужнішими, дозволяючи їм переносити стан.

Під час перевірки адреси сценарій отримає доступ до даних, що передаються виходом, транзакції, що перевіряється, і деяких додаткових даних, які називаються викупниками, які транзакція надає для кожного введення. Переглядаючи всю цю інформацію, сценарій має достатньо контексту, щоб дати відповідь «так» чи «ні» у дуже складних ситуаціях і випадках використання.

Підсумовуючи, EUTXO розширює модель UTXO, дозволяючи вихідним адресам містити складну логіку, щоб вирішувати, які транзакції можуть їх розблокувати, і додаючи власні дані до всіх вихідних даних.

Модель EUTXO пропонує унікальні переваги перед іншими моделями бухгалтерського обліку. Успіх або невдача перевірки транзакції залежить лише від самої транзакції та її вхідних даних, а не від чогось іншого в блокчейні. Як наслідок, дійсність транзакції може бути перевірена поза ланцюгом, перш ніж транзакція буде відправлена в блокчейн. Транзакція все ще може завершитися невдачею, якщо якась інша транзакція одночасно споживає вхідні дані, які транзакція очікує, але якщо всі вхідні дані все ще присутні, транзакція гарантовано буде успішною.

Це контрастує з моделлю на основі облікового запису (як використовується Ethereum), де транзакція може бути невдалою під час виконання сценарію. Цього ніколи не може статися в EUTXO. Крім того, витрати на виконання транзакцій можуть бути визначені поза ланцюгом перед передачею – ще одна функція, неможлива в Ethereum.

Нарешті, через «локальну» природу перевірки транзакцій можливий високий ступінь паралельності: вузол, в принципі, міг би перевіряти транзакції паралельно, якщо ці транзакції не намагаються споживати одні й ті самі вхідні дані. Це чудово як для ефективності, так і для міркування, спрощує аналіз можливих результатів і доводить, що «нічого поганого» не може статися.

2.4.1. Транзакції. Перш ніж зануритися у сценарії, ми почнемо з того, що розглянемо детальну структуру транзакцій, з яких складається книга блокчейну.

На рис. 2.2 зображено основний тип даних транзакції.

```
data Tx = Tx {
txInputs :: Set.Set TxIn,
— входи для цієї транзакції
txOutputs :: [TxOut],
—  виходи цієї транзакції
txForge :: Ada,
— створена  валюта цієї транзакції
txFee :: Ada,
— комісія за цю  транзакцію
txValidRange :: (Slot, Slot),
— інтервал дії цієї транзакції
txSignatures :: Map PubKey Signature
— підписи цієї транзакції
}
```

Рис. 2.2. Основний тип даних транзакції

Входи та виходи є зв'язками з попередніми та наступними транзакціями. Створена валюта - це нова валюта, введена в книгу обліку, тоді як комісія - це частина загальної вартості транзакції, яка виплачується лідеру слота, який інтегрує транзакцію в блок на блокчейні. Поточний лідер слотів у кожному слоті створює один новий блок блокчейна (містить багато транзакцій), фіксований часовий інтервал, визначений протоколом блокчейна "Уроборос". Поточна кількість слотів надає уявлення про час, що пройшов з моменту створення блокчейна. Інтервал дійсності визначає діапазон слотів, у якому

поточна транзакція може бути перевірена: поза цим інтервалом транзакція є недійсною.

Кожна транзакція асоціюється з унікальним ідентифікатором typeTxID, який є просто криптографічним хешем серіалізованого представлення транзакції без підписів. Фактично, підписи підписують хеш транзакції, а не всю транзакцію. Виходами транзакції, txOutputs, є список(див.рис.2.3).

```
data TxOut = TxOut {
txOutAddress :: TxID,
— ідентифікатор цілі платежу
txOutValue :: Ada,
— значення виходу
txOutType :: TxOutType
— тип виходу
}
```

Рис. 2.3. Список виходів транзакції txOutputs

Виходи завжди виплачують фіксовану вартість криптовалюти на адресу. У найпростішому випадку ця адреса ідентифікує криптографічну пару ключів, що міститься у криптографічному гаманці. Це, як правило, називається платіж за допомогою публічного використання та використовується для прямих платежів між двома користувачами блокчейну.

Результати однієї транзакції споживаються вхідними матеріалами наступних транзакцій. З цією метою кожна вхідна специфікація TxIn містить вихідну посилання типу TxOutRef(див.рис.2.4).

```
data TxIn = TxIn {
txInRef :: TxOutRef,
txInType :: TxInType
}
data TxOutRef = TxOutRef {
id :: TxID,
index :: Int
}
```

Рис. 2.4. Посилання типу TxOutRef

Посилання на вихідну інформацію типу TxOutRef однозначно ідентифікує вихідні дані в існуючій книзі за допомогою комбінації ідентифікатора транзакції, що містить вихідні дані та індекс, до списку результатів цієї транзакції. Більше того, вхідні дані транзакцій також містять компонент txInType(див.рис.2.5), який має узгоджуватися з TxOutType споживаного виходу.

```
data TxInType
= ConsumePublicKeyAddress PubKey
| ConsumeScriptAddress ValidatorScript
RedeemerScript
```

Рис. 2.5. Компонент txInType

Коли вхід транзакції txIn :: TxIn посилається у своєму txInRef на вихід транзакції txOut :: TxOut, вказуючи ідентифікатор ідентифікатора транзакції цього виводу та індекс txOut у списку результатів ідентифікатора, ми кажемо, що txIn витрачає txOut. Враховуючи блокчейн як список транзакцій, ми можемо визначити набір усіх результатів, які з’являються у транзакції, але не витрачаються на введення будь -якої іншої транзакції.

Цей набір результатів називається набором невитрачених транзакцій (UTxO). Він визначає всю вартість (кошти), які ще можуть бути витрачені, і,

отже, є первинною структурою даних, що представляє поточний стан книги UTxO.

2.5. Мова сценаріїв Plutus Core

Для реалізації моделі EUTXO необхідно чітко визначити терміни скрипт і дані. Скрипти вимагають певної, чітко визначеної мови сценаріїв, і також важливо визначити тип даних, які додаються до вихідних даних і використовуються як сповільнювачі.

Тут на допомогу приходить Plutus Core. Plutus Core — це мова сценаріїв, яку використовує Cardano. Це проста функціональна мова, подібна до Haskell, і велика підмножина Haskell може використовуватися для написання скриптів Plutus Core[2]. Як автор контракту, ви не пишете жодного Plutus Core. Усі програми Plutus Core генеруються плагіном компілятора Haskell.

Ці скрипти виконуватимуться вузлами під час перевірки транзакції «живо» в ланцюжку. Вони або блокують UTXO у вигляді сценаріїв валідації, або як політики карбування, які контролюють карбування та запис нативних монет.

Дані Redeemer — це простий (алгебраїчний) тип даних, який можна легко визначити в Haskell, що є ще однією причиною, чому Haskell є хорошим варіантом для написання скриптів Plutus Core. На практиці розробник смарт-контракту писатиме сценарії перевірки на Haskell, які потім автоматично компілюються в Plutus Core.

Datum - поле даних на виходах сценарію в моделі Extended UTXO.

Відповідні бібліотеки Haskell спрощують написання такої логіки перевірки, надаючи основні типи даних для перевірки транзакцій під час перевірки, а також пропонуючи безліч допоміжних функцій і абстракцій вищого рівня, дозволяючи авторам контракту зосередитися на бізнес-логіці і не турбуватися про занадто багато детале низького рівня.

2.6. Plutus Application Framework (PAF)

Plutus Application Framework –це частина платформи Plutus, є основою для розробки розподілених додатків за допомогою блокчейну Cardano.

Стан ланцюга сценаріїв перевірки може бути змінений лише транзакціями, які витрачають і виробляють вихід сценарію. При написанні програми Plutus нам потрібно враховувати не тільки внутрішню частину програми (скрипти Plutus Core), але й частину поза ланцюгом, яка створює та передає транзакції.

Код поза ланцюгом написаний на Haskell, так само, як і внутрішній код. Таким чином, нам потрібно лише один раз написати бізнес-логіку. Потім ми можемо використовувати його в сценарії перевірки та в коді, який створює транзакції, які запускають сценарій перевірки.

Багатьом додаткам потрібно стежити за набором UTXO для змін до певних адрес, тому, якщо ми пишемо наш контракт як кінцевий автомат, нам потрібно відстежувати невитрачений результат, який представляє поточний стан машини, і оновлювати наш локальний стан, коли в ланцюжку зміни стану. Аналогічно, багатьом додаткам потрібно зв’язуватися з бекендом гаманця, щоб отримати доступ до криптовалюти, яку вони використовують для транзакцій.

PAF забезпечує легкий доступ до служб, які зазвичай використовуються додатками Plutus. Додатки, розгорнуті за допомогою бібліотек фреймворка, можна запускати на серверній частині програми Plutus, яка забезпечує підтримку під час виконання для доступу до блокчейну та інших проблем, таких як збереження, ведення журналів і моніторинг.

Програми, написані поверх PAF, автоматично забезпечують інтерфейс HTTP та WebSocket, який можна використовувати для взаємодії з програмою з веб-браузера.

2.7. Поетапне програмування

Як і веб-додатки, децентралізовані програми (dapps), побудовані на блокчейнах, складаються з двох окремих компонентів, які розгортаються в окремих середовищах виконання:

1. Код на ланцюгу, що зберігається в ланцюжку блоків і виконується під час включення нових транзакцій у нові блоки, які додаються до ланцюжка (подібно до серверного компонента веб-додатка).

2. Код поза ланцюгом, який зазвичай розгортається через веб-сайт і виконується на клієнтській машині користувача блокчейну з доступом до криптографічного гаманця користувача, так само як клієнтська частина веб-програми, що працює у веб-браузері користувача; Фактично, код поза ланцюгом dapp часто також виконується у веб-браузері. Чому потрібне це розкладання? Код у мережі містить договірні компоненти dapp. Необхідно забезпечити, щоб лише транзакції, які відповідають договірним зобов'язанням, були успішно підтверджені та додані до ланцюжка. Іншими словами, цілісність смарт-контракту залежить від цілісності коду в ланцюжку: таким чином, нам потрібно зберігати його в криптографічно незмінному блокчейні, щоб запобігти фальсифікації. Більше того, лідери слотів (сервери, які додають нові блоки в ланцюжок у протоколі proof-of-stake Ouroboros, що відповідають майнерам протоколу proof-of-work, який використовується в біткойні) повинні виконувати код у ланцюжку, зокрема, сценарії перевірки — щоб гарантувати, що в ланцюжок приймаються лише транзакції, які відповідають усім відповідним договірним зобов'язанням[2].

І навпаки, код поза ланцюгом, який подає нову транзакцію лідерам слотів для перевірки та включення в ланцюжок, обов'язково повинен працювати в тісному зв'язку з криптовалютним гаманцем користувача контракту. Зрештою, кожна транзакція має бути оплачена шляхом включення невеликої комісії за транзакцію, а криптографічний гаманець — єдине місце,

де зберігаються необхідні криптографічні облікові дані (все інше може поставити під загрозу безпеку коштів).

2.8. Висновки до розділу

Було проведено аналіз архітектури та технології блокчейну Cardano. Описано модель програмування Plutus та основні компоненти смарт-контрактів блокчейну Cardano.

Plutus забезпечує новий інтегрований підхід до розробки смарт-контрактів і розподілених додатків, який є зручнішим і безпечнішим, ніж попередні альтернативи. І внутрішній, і позаланцюжний код засновані на одній мові. Розробники використовують єдину кодову базу, яку ланцюжок інструментів Plutus потім автоматично розділяє на внутрішній і позаланцюжний код і пакети для розгортання.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ СМАРТ-КОНТРАКТУ В PLUTUS PLAYGROUND

3.1. Платформа для написання додатків Plutus

Платформа Plutus – це платформа для написання додатків, які взаємодіють із розподіленим реєстром із можливостями написання сценаріїв, зокрема блокчейном Cardano.

Контракти Plutus складаються з частин, які виконуються на блокчейні (код у мережі), і частин, які виконуються на машині користувача (код поза ланцюгом або клієнт). Як внутрішнє, так і поза ланцюгом код написані на Haskell, а розумні контракти Plutus фактично є програмами на Haskell. Позаланцюжний код можна написати за допомогою PAF, і цей код потім компілюється компілятором GHC (Glasgow Haskell Compiler), в той час як внутрішній код (написаний за допомогою Plutus Core) компілюється компілятором Plutus.

3.2. Середовище Plutus Playground

Plutus Playground забезпечує середовище для написання та тестування смарт-контрактів перед їх випуском на блокчейні Cardano. Plutus Core, яка є мовою смарт-контрактів, вбудована в реєстр, заснована на формальних принципах програмування Haskell і дозволяє розробникам писати програми з високою надійністю, які взаємодіють з Cardano. Haskell був обраний як основа для платформи Plutus, оскільки він виділяється серед інших мов програмування тим, що пропонує можливість писати більш безпечний код. Використання Haskell для розгортання смарт-контрактів гарантує, що контракти запрограмовані на те, що від них очікується, і їх можна перевірити на точність перед впровадженням.

Plutus Playground призначений для людей, які створюють децентралізовані програми (DApps), і програмістів розумних контрактів, які хочуть працювати з Cardano. Plutus стане платформою для створення DApps для ланцюгів поставок, відстеження та відстеження, медичних записів, голосування особи, реєстрації власності, P2P-платежів та фінансових систем.

Plutus Playground складається з редактора коду та симулятора (див.рис.3.1-2).

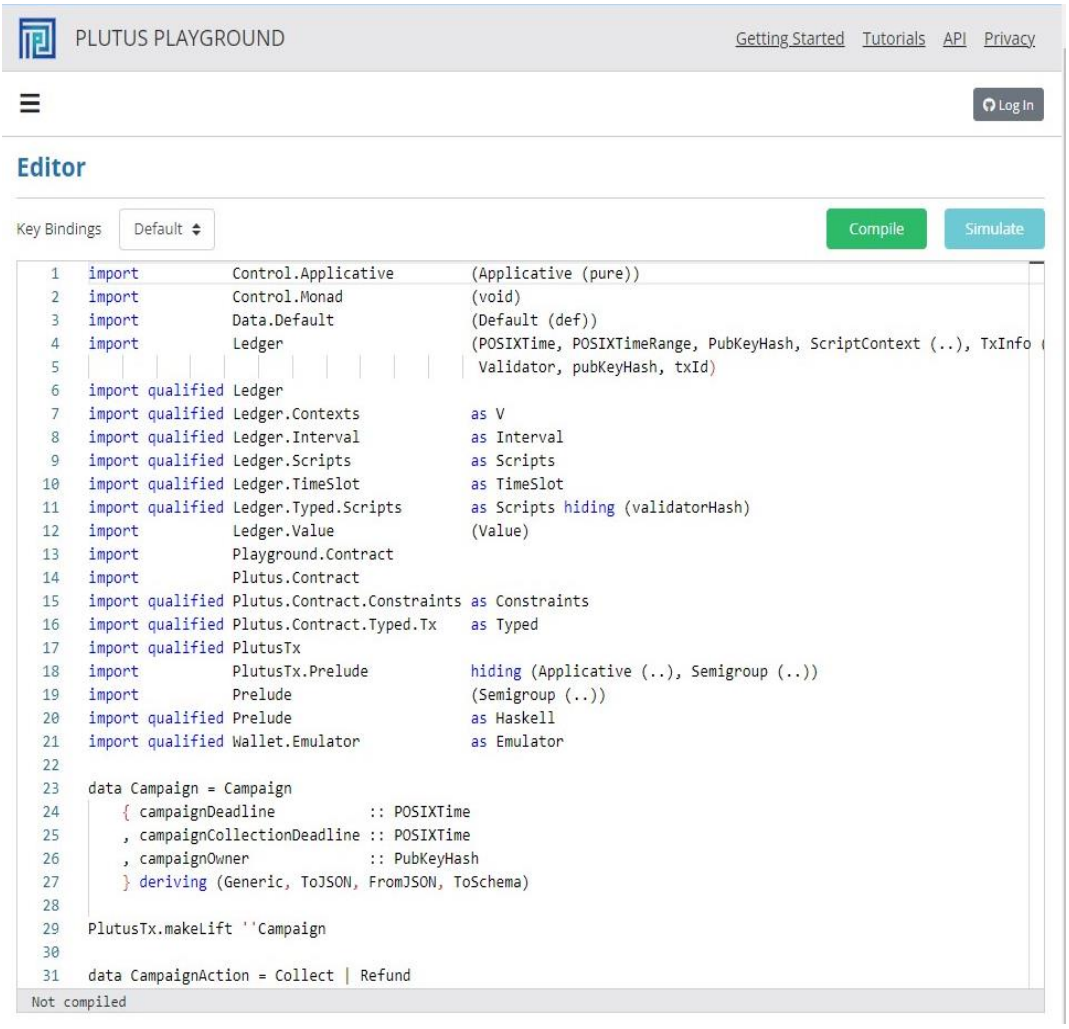

Рис. 3.1. Plutus Playground (редактор коду)

Коли програму скомпільовано, ми можемо запускати її в змодельованому середовищі.

Натиснувши кнопку Simulate. Редактор коду замінюється симулятором.

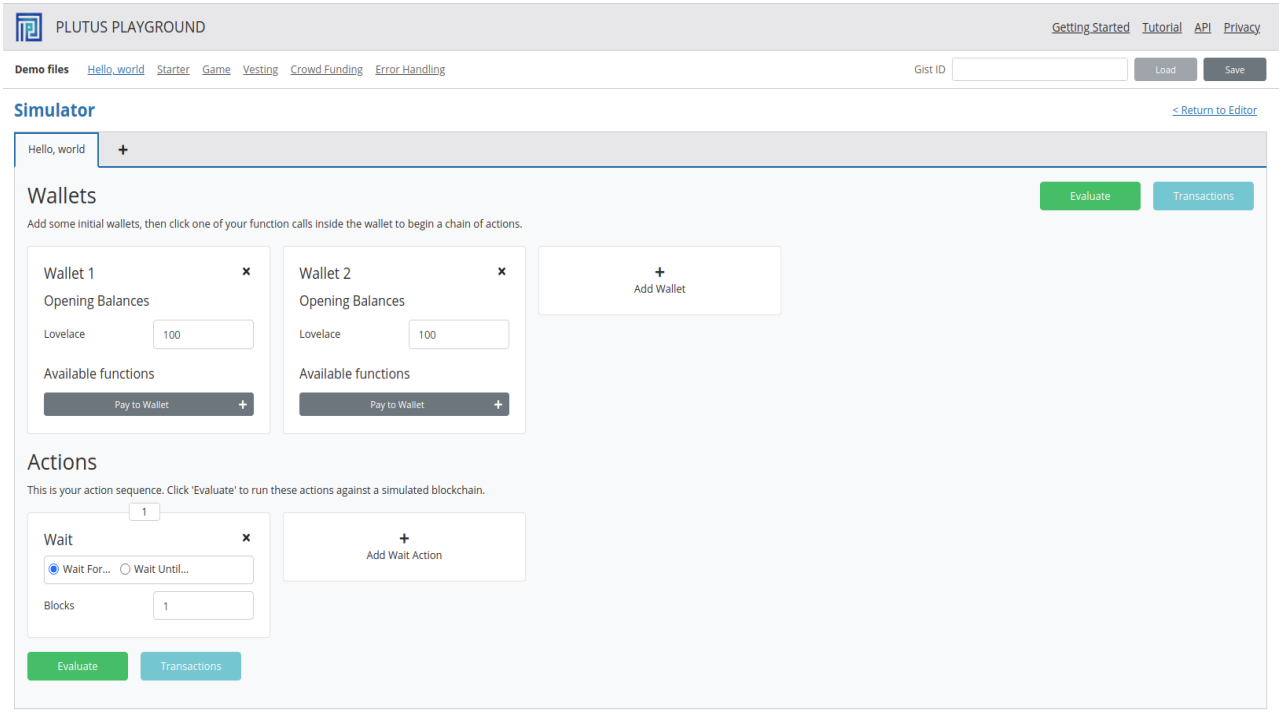

Рис. 3.2. Симулятор Plutus Playground

Ми можемо використовувати симулятор для визначення складних сценаріїв із кількома агентами, які торгують і спілкуються через Cardano.

Симулятор показує, як буде вести себе контракт на блокчейні Cardano. Важливим аспектом цього є те, що він може діяти як інструмент тестування, оскільки демонструє принципи роботи. Користувачі можуть визначати та змінювати гаманці, які взаємодіють з контрактом, а також дії, що впливають на результат. Потім результати можна оцінити, щоб побачити, що відбувається в блокчейні та як відбуваються транзакції (див.рис.3.3).

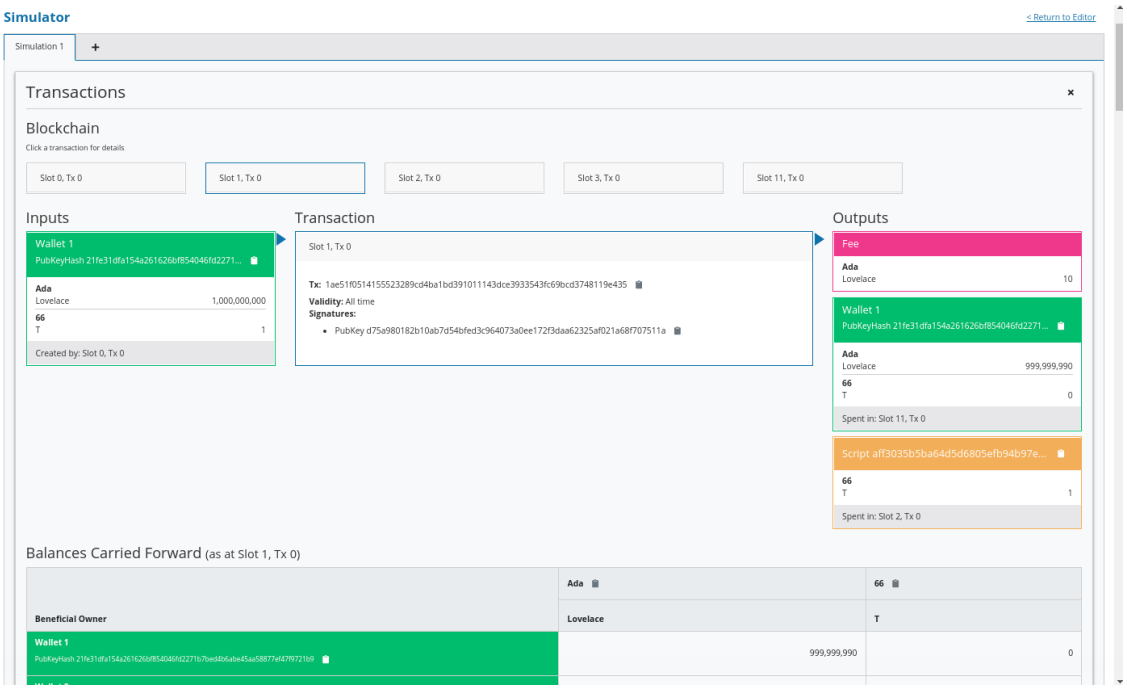

Рис. 3.3. Симулятор Plutus Playground (транзакції смарт-контракту)

3.3. Написання додатка Plutus на Plutus Playground

Програми Plutus – це програми, які працюють поза мережею та керують активними екземплярами контрактів. Вони стежать за блокчейном, запитують введення користувачів і надсилають транзакції в блокчейн.

У цьому смарт-контракті ми збираємося написати програму Plutus для громадського фінансування (краудфандінг). Програма приймає кошти від учасників кампанії, дає проміжок часу власнику кампанії щоб зібрати кошти і якщо кошти не було зібрано то повертає їх учасникам кампанії.

На рис. 3.4. зображено лістинг імпорту потрібних бібліотек.

```
1   import          Control.Applicative         (Applicative (pure))
2   import          Control.Monad               (void)
3   import          Data.Default                (Default (def))
4   import          Ledger                      (POSIXTime, POSIXTimeRange, PubKeyHash, ScriptContext (..), TxInfo (..),
5                                                Validator, pubKeyHash, txId)
6   import qualified Ledger
7   import qualified Ledger.Contexts            as V
8   import qualified Ledger.Interval            as Interval
9   import qualified Ledger.Scripts             as Scripts
10  import qualified Ledger.TimeSlot            as TimeSlot
11  import qualified Ledger.Typed.Scripts       as Scripts hiding (validatorHash)
12  import          Ledger.Value                (Value)
13  import          Playground.Contract
14  import          Plutus.Contract
15  import qualified Plutus.Contract.Constraints as Constraints
16  import qualified Plutus.Contract.Typed.Tx    as Typed
17  import qualified PlutusTx
18  import          PlutusTx.Prelude            hiding (Applicative (..), Semigroup (..))
19  import          Prelude                     (Semigroup (..))
20  import qualified Prelude                     as Haskell
21  import qualified Wallet.Emulator             as Emulator
```

Рис. 3.4. Лістинг імпорту бібліотек

3.3.1 Визначення типів. Ми починаємо з визначення деяких типів даних(див.рис.3.5-6)., які нам знадобляться для програми громадського фінансування (crowdfunding campaign).

```
23  -- | Крауфандінг кампанія.
24  data Campaign = Campaign
25      { campaignDeadline           :: POSIXTime
26      -- ^ дата по яку в кампанії кошти можуть бути надані.
27      , campaignCollectionDeadline :: POSIXTime
28      -- ^ дата по яку власник кампанії має право для збору коштів
29      , campaignOwner              :: PubKeyHash
30      -- ^ Відкритий ключ з власника кампанії.
31      --Цей ключ має право на отримання коштів , якщо кампанія є успішною.
32      } deriving (Generic, ToJSON, FromJSON, ToSchema)
33
34  PlutusTx.makeLift ''Campaign
```

Рис. 3.5. Лістинг визначення типу кампанії

Ми використовуємо тип "PubKeyHash" для ідентифікації одержувачів. Під час здійснення платежу ми зможемо використовувати хеші для створення двох виходів транзакцій відкритих ключів

Дії, які можуть бути прийняті учасниками цього договору. значення з `CampaignAction` це представлено в якості на redeemer. Скрипт валідатора потім перевіряє чи виконуються умови для виконання цих дій.

```
38  data CampaignAction = Collect | Refund
39
40  PlutusTx.unstableMakeIsData ''CampaignAction
41  PlutusTx.makeLift ''CampaignAction
42
43  type CrowdfundingSchema =
44          Endpoint "schedule collection" ()
45          .\/ Endpoint "contribute" Contribution
46
47  newtype Contribution = Contribution
48          { contribValue :: Value
49          -- ^ скільки внести
50          } deriving stock (Haskell.Eq, Show, Generic)
51            deriving anyclass (ToJSON, FromJSON, ToSchema, ToArgument)
52
53  --Створення значення "Campaign" з параметрів кампанії, використовуючи відкритий ключ гаманця.
54  mkCampaign :: POSIXTime -> POSIXTime -> Wallet -> Campaign
55  mkCampaign ddl collectionDdl ownerWallet =
56      Campaign
57          { campaignDeadline = ddl
58          , campaignCollectionDeadline = collectionDdl
59          , campaignOwner = pubKeyHash $ Emulator.walletPubKey ownerWallet
60          }
61
62  -- |'POSIXTimeRange' період часу, протягом якого кошти можуть бути зібрані
63  collectionRange :: Campaign -> POSIXTimeRange
64  collectionRange cmp =
65      Interval.interval (campaignDeadline cmp) (campaignCollectionDeadline cmp - 1)
66
67  -- | The 'POSIXTimeRange' період часу, протягом якого можна получити відшкодування
68  refundRange :: Campaign -> POSIXTimeRange
69  refundRange cmp =
70      Interval.from (campaignCollectionDeadline cmp)
```

Рис. 3.6. Лістинг визначення типів

3.3.2. Екземпляри для типів даних. Тип "Campaign" має екземпляри для ряду класів типів. Ці екземпляри дозволяють серіалізувати "Campaign" в різні формати. ToJSON і FromJSON необхідні для серіалізації JSON. Об’єкти JSON передаються між інтерфейсом (наприклад, Playground) і екземпляром програми. PlutusTx.FromData і PlutusTx.ToData використовуються для значень, які додаються до транзакцій, наприклад, як <redeemer> виходу

сценарію. Цей клас використовується програмою Plutus під час виконання для створення значень "Data" . Нарешті, PlutusTx.makeLift це твердження шаблону Haskell, який генерує екземпляр PlutusTx.Lift.Class.Lift класу для "Campaign". Цей клас використовується компілятором Plutus під час компіляції для створення Plutus Core програм.

3.3.3. Визначення сценаріїв валідаторів. Сценарій валідатора — це частина нашого додатка Plutus, що входить до коду на ланцюгу. Завдання валідатора — розглядати окремі транзакції окремо і вирішувати, чи є вони дійсними. Валідатори Plutus мають такий підпис типу:

```
d -> r -> ValidatorCtx -> Bool
```
(3.1)

де d – тип <datum>; r – тип <redeemer>.

Між програмами Plutus і сценаріями перевірки існує зв’язок n-to-n. Програми можуть працювати з кількома валідаторами, а валідатори можуть використовуватися різними програмами.

В даній кваліфікаційній роботі розробляється смарт-контракт та програма Plutus для громадського фінансування (краудфандінг). Програма приймає кошти від учасників кампанії, дає проміжок часу власнику кампанії щоб зібрати кошти і якщо кошти не було зібрано то повертає їх учасникам кампанії.

У цьому смарт-контракті нам потрібно декілька валідаторів (див.рис.3.7-8). Валідатор переглядає значення транзакії щоб перевірити, чи виконуються умови для здійснення платежу.

```
74  --Валідатор повернення коштів:
75  validRefund :: Campaign -> PubKeyHash -> TxInfo -> Bool
76  validRefund campaign contributor txinfo =
77      -- Перевірка, що транзакція потрапляє в діапазон відшкодування кампанії
78      (refundRange campaign `Interval.contains` txInfoValidRange txinfo)
79      -- Перевірка, що транзакція підписана учасником
80      && (txinfo `V.txSignedBy` contributor)
81
82  --Валідатор збору коштів кампанії
83  validCollection :: Campaign -> TxInfo -> Bool
84  validCollection campaign txinfo =
85      -- Перевірка, чи транзакція потрапляє в діапазон збору кампанії
86  (collectionRange campaign `Interval.contains` txInfoValidRange txinfo)
87      -- Перевірка, що транзакція підписана власником кампанії
88      && (txinfo `V.txSignedBy` campaignOwner campaign)
```

Рис. 3.7. Лістинг визначення валідаторів

Сценарій валідатора має тип "CrowdfundingValidator" і додатково параметризований визначенням "Campaign". Цей аргумент надається клієнтом Plutus, використовуючи 'PlutusTx.applyCode'. У результаті визначення "Campaign" є частиною адреси сценарію, а різні кампанії мають різні адреси.

```
92  mkValidator :: Campaign -> PubKeyHash -> CampaignAction -> ScriptContext -> Bool
93  mkValidator c con act p = case act of
94      -- "refund" гілка повернення коштів
95      Refund  -> validRefund c con (scriptContextTxInfo p)
96      -- "collection" гілка збору коштів
97      Collect -> validCollection c (scriptContextTxInfo p)
98
99  --Скрипт валідатора який визначає, чи може власник кампанії отримати кошти,
100 --та чи учасники можуть вимагати повернення коштів.
101 contributionScript :: Campaign -> Validator
102 contributionScript = Scripts.validatorScript . typedValidator
```

Рис. 3.8. Лістинг визначення валідаторів

Також нам знадобиться шаблон для компіляції валідатора до сценарію Plutus(див.рис.3.9).

```
104  data Crowdfunding
105  instance Scripts.ValidatorTypes Crowdfunding where
106      type instance RedeemerType Crowdfunding = CampaignAction
107      type instance DatumType Crowdfunding = PubKeyHash
108
109  typedValidator :: Campaign -> Scripts.TypedValidator Crowdfunding
110  typedValidator = Scripts.mkTypedValidatorParam @Crowdfunding
111      $$(PlutusTx.compile [|| mkValidator ||])
112      $$(PlutusTx.compile [|| wrap ||])
113      where
114          wrap = Scripts.wrapValidator
```

Рис. 3.9. Лістинг шаблону для компіляції валідатора

Клас Ledger.Typed.Scripts.Validators.ValidatorTypes визначає типи валідатора, а typedValidator містить скомпільований основний код Plutus typedValidator.

3.3.4. Визначення адреси, контракту та створення зразку кампанії. Для роботи смарт-контракту нам потрібно визначити екземпляр кампанії, адресу кампанії та контракт кампанії (див.рис.3.10).

```
116  -- | Адреса кампанії [[Campaign]]
117  campaignAddress :: Campaign -> Ledger.ValidatorHash
118  campaignAddress = Scripts.validatorHash . contributionScript
119
120  -- | Краудфандинговий контракт для кампанії 'Campaign'.
121  crowdfunding :: AsContractError e => Campaign -> Contract () CrowdfundingSchema e ()
122  crowdfunding c = selectList [contribute c, scheduleCollection c]
123
124  -- | Зразок кампанії
125  theCampaign :: POSIXTime -> Campaign
126  theCampaign startTime = Campaign
127      { campaignDeadline = startTime + 10000
128      , campaignCollectionDeadline = startTime + 20000
129      , campaignOwner = pubKeyHash $ Emulator.walletPubKey (Emulator.knownWallet 1)
130      }
```

Рис. 3.10. Лістинг визначення адреси, контракту та створення зразку кампанії

Власник кампанії представлений публічним ключем 1 гаманця емулятора в смарт-контракті на блокчейні потрібно представляти публічний ключ гаманця власника кампанії.

3.3.5 Гілки смарт-контракту. Гілка контракту внесок "contribute" для конкретної Кампанії "Campaign"(див.рис.3.11) надає кінцеву точку, яка дозволяє користувачеві вводити свій відкритий ключ і внесок. Потім чекає завершення кампанії та збирає відшкодування, якщо кошти не були зібрані.

```
134 contribute :: AsContractError e => Campaign -> Promise () CrowdfundingSchema e ()
135 contribute cmp = endpoint @"contribute" $ \Contribution{contribValue} -> do
136     contributor <- pubKeyHash <$> ownPubKey
137     let inst = typedValidator cmp
138         tx = Constraints.mustPayToTheScript contributor contribValue
139                 <> Constraints.mustValidateIn (Interval.to (campaignDeadline cmp))
140     txid <- fmap txId (submitTxConstraints inst tx)
141     utxo <- watchAddressUntilTime (Scripts.validatorAddress inst) (campaignCollectionDeadline cmp)
142     let flt Ledger.TxOutRef{txOutRefId} _ = txid Haskell.== txOutRefId
143         tx' = Typed.collectFromScriptFilter flt utxo Refund
144                 <> Constraints.mustValidateIn (refundRange cmp)
145                 <> Constraints.mustBeSignedBy contributor
146     if Constraints.modifiesUtxoSet tx'
147     then void (submitTxConstraintsSpending inst utxo tx')
148     else pure ()
```

Рис. 3.11. Лістинг гілки контракту внеску

Гілка контракту власника кампанії для даної кампанії "Campaign" (див.рис.3.12), яка спостерігає за адресою кампанії для отримання внесків і збирає їх, якщо ціль фінансування була досягнута вчасно. Відкриємо кінцеву точку, яка дозволяє власнику кампанії зібрати кошти.

```
152 scheduleCollection :: AsContractError e => Campaign -> Promise () CrowdfundingSchema e ()
153 scheduleCollection cmp =
154     endpoint @"schedule collection" $ \() -> do
155       let inst = typedValidator cmp
156
157       _ <- awaitTime $ campaignDeadline cmp
158       unspentOutputs <- utxosAt (Scripts.validatorAddress inst)
159
160       let tx = Typed.collectFromScript unspentOutputs Collect
161                   <> Constraints.mustValidateIn (collectionRange cmp)
162       void $ submitTxConstraintsSpending inst unspentOutputs tx
```

Рис. 3.12. Лістинг гілки контракту збору коштів

3.3.6 Транзакції в смарт-контракті. Припустимо, що існує кампанія `c :: Campaign` з двома учасниками (ідентифікується відкритим ключем `pc_1` і `pc_2`) та одним власником кампанії (pco).

Кожен учасник створює транзакцію, `t_1` та `t_2`, виходи яких яких є заблоковано сценаріями `contributionScript c pc_1` та `contributionScript c pc_1` відповідно.

Є два результати кампанії.

1. Власник кампанії збирає кошти від обох учасників. У цьому випадку власник створює одну транзакцію з двома входами, посилаючись на "t_1" і "t_2". Кожен вхід містить скрипт `contributionScript c`, спеціалізований для учасника. Сценарій викупу цієї транзакції містить значення `Collect`, що спонукає сценарій перевірки перевірити гілку на наявність `Collect`.

2. Відшкодування. У цьому випадку кожен вкладник створює транзакцію з одним введенням, вимагаючи повернення своєї частини коштів. Цей випадок охоплює гілка відшкодування "Refund", а його сценарієм викупу є дія відшкодування "Refund".

В обох випадках сценарій перевірки виконується двічі. У першому випадку існує одна транзакція, яка використовує обидва входи. У другому випадку є дві різні транзакції, які можуть відбутися в різний час.

3.3.7 Розгортання програми на Plutus Playground. У нас є всі функції, необхідні для на ланцюгу та поза ланцюгом частин програми. Кожен контракт на Playground має визначати свій публічний інтерфейс. Сервер Playground використовує визначення кінцевих точок для заповнення інтерфейсу користувача (через схему, у нашому випадкCrowdfundingSchema) та для початку моделювання. Кінцеві точки – це визначення високого рівня нашої програми.

Функція "selectList" (див рис.3.10) діє як вибір між двома гілками. Ліва гілка починається з "contribute", а права гілка починається з

"scheduleCollection". Програма відкриває обидві кінцеві точки та переходить до гілки, яка першою отримує відповідь. Отже, якщо ви викликаєте кінцеву точцу "contribute"в одному із змодельованих гаманців, вона викличе "contribute" і проігнорує сторону "scheduleCollection" контракту. Нам також знадобиться кілька декларацій, які генерують код скріплення для Playground.

Лістинг визначення публічного інтерфейсу кінцевих точок то код скріплення зображено на рис. 3.13.

```
endpoints :: AsContractError e => Contract () CrowdfundingSchema e ()
endpoints = crowdfunding (theCampaign $ TimeSlot.scSlotZeroTime def)
mkSchemaDefinitions ''CrowdfundingSchema
$(mkKnownCurrencies [])
```

Рис. 3.13. Лістинг визначення кінцевих точок та коду скріплення

3.3.8 Моделювання. Після написання смарт-контракту та Plutus програми можна скомпілювати договір і створити моделювання.

Наступна послідовність дій (див.рис.3.14) призводить до успішної кампанії.

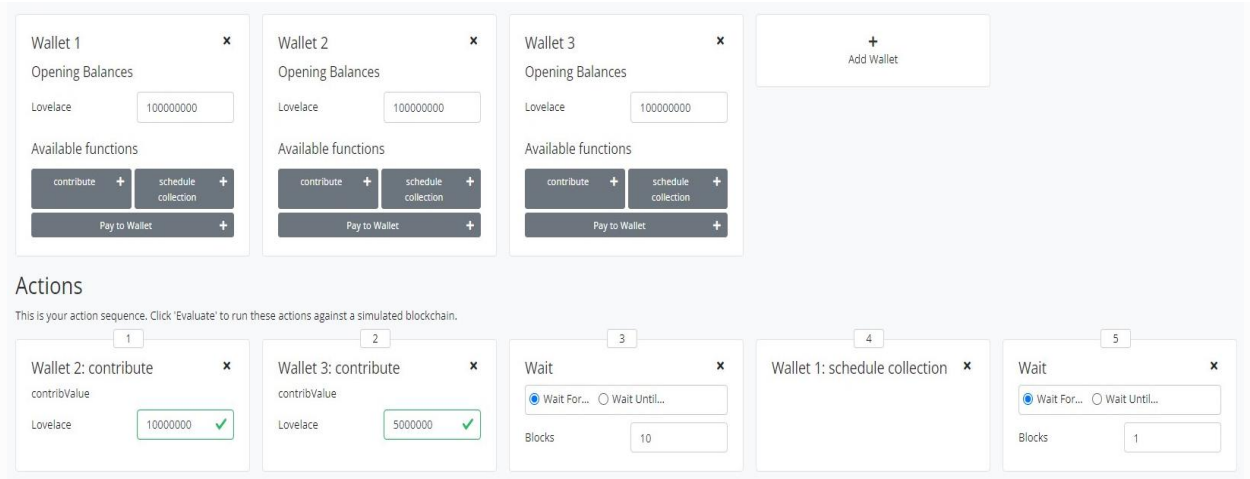

Рис. 3.14. Симуляція успішної кампанії з 2 гаманцями учасниками

Гаманець 2 та 3 заносять кошти в кампанію. Після очікування в 10 блоків власник кампанії має право зібрати кошти, що виконується через кнопку «schedule collection». Чекаємо ще 1 блок щоб виконалася транзакція.

Перша транзакція це генезис яка створює початкові кошти на гаманцях (див.рис.3.15). Гаманець 2 вносить кошти в розмірі 10,000,000 Lovelace (див.рис.3.16), а гаманець 3 в розмірі 5,000,000 Lovelace (див.рис.3.17). Власник кампанії збирає кошти в 10 слоті (див.рис.3.18-19).

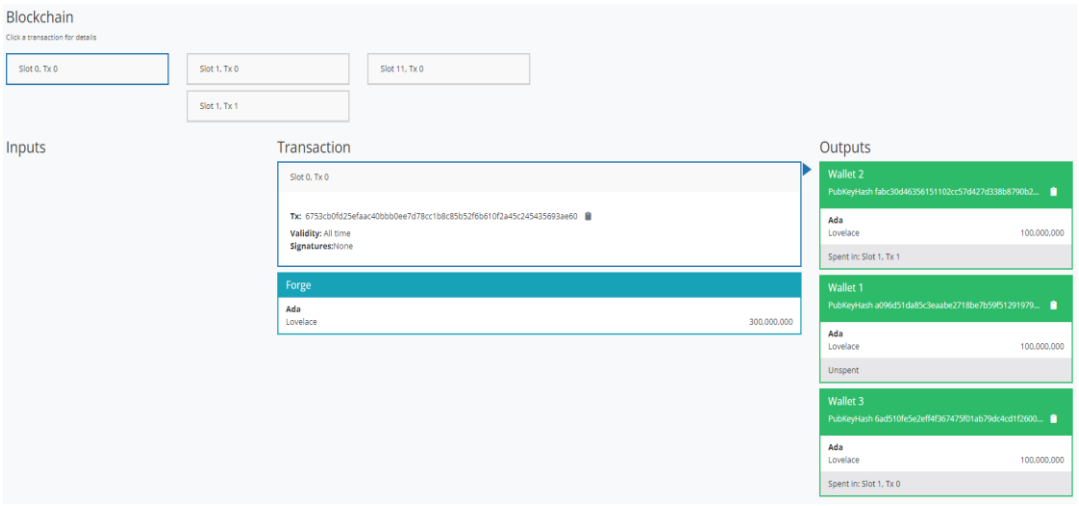

Рис. 3.15. Симуляція 1 генезис 0 слот

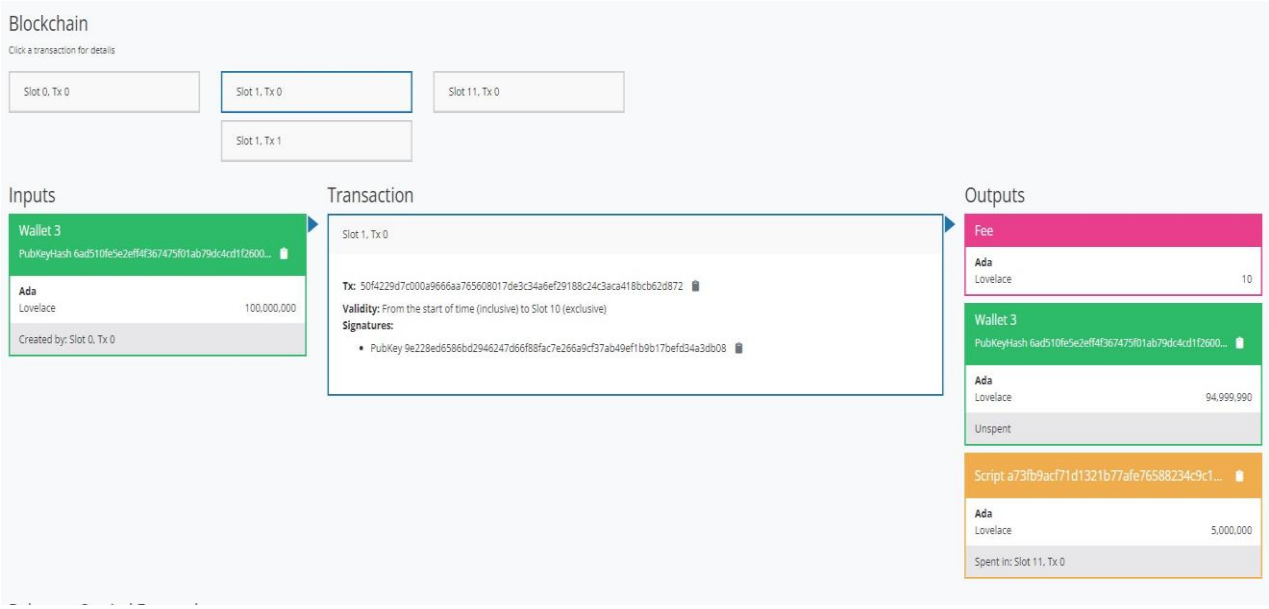

Рис. 3.16. Симуляція 1 слот 1 внесення коштів 2 гаманцем

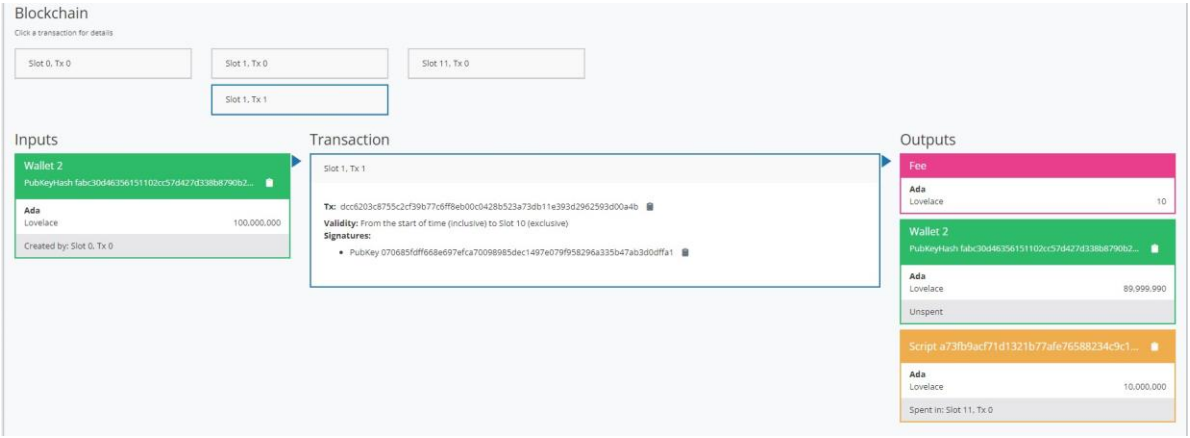

Рис. 3.17. Симуляція 1 слот 1 внесення коштів 3 гаманцем

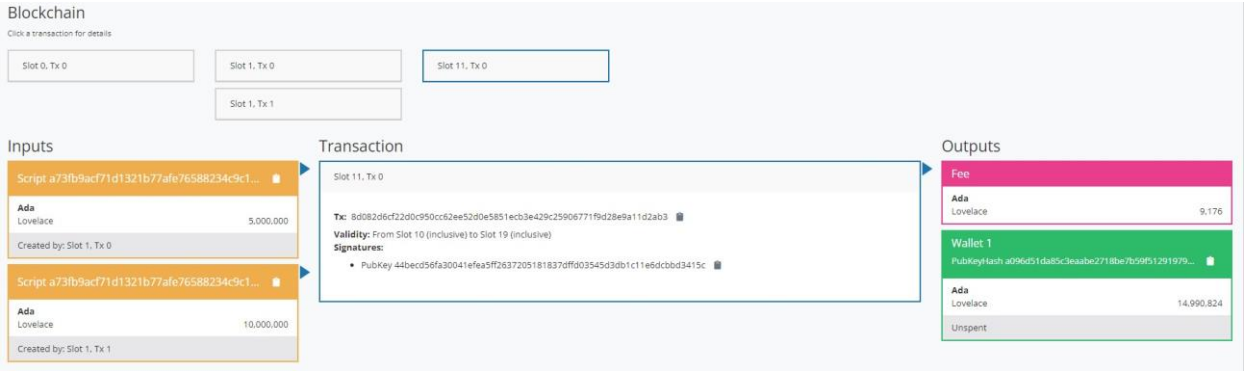

Рис. 3.18. Симуляція 1 слот 10 збір коштів власником кампанії

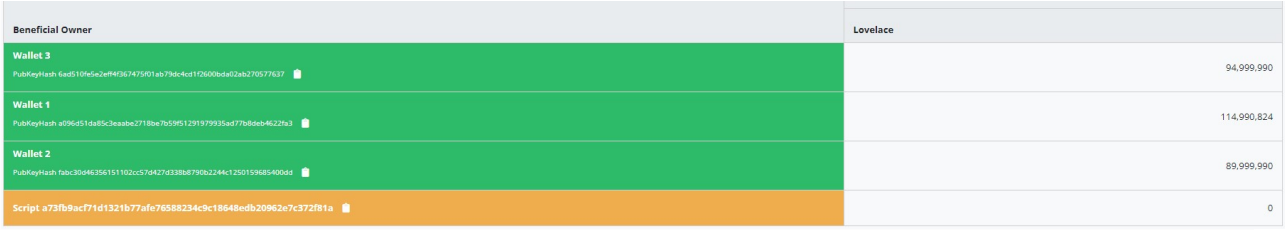

Рис. 3.19. Симуляція 1 слот 10 остаточний баланс

Як бачимо кампанія пройшла успішно і 15,000,000 Lovelace з вирахуванням комісії за транзакції поступили на гаманець власника кампанії.

Симуляція 2 повернення коштів. Наступна послідовність дій (див. рис. 3.20) призводить до повернення коштів.

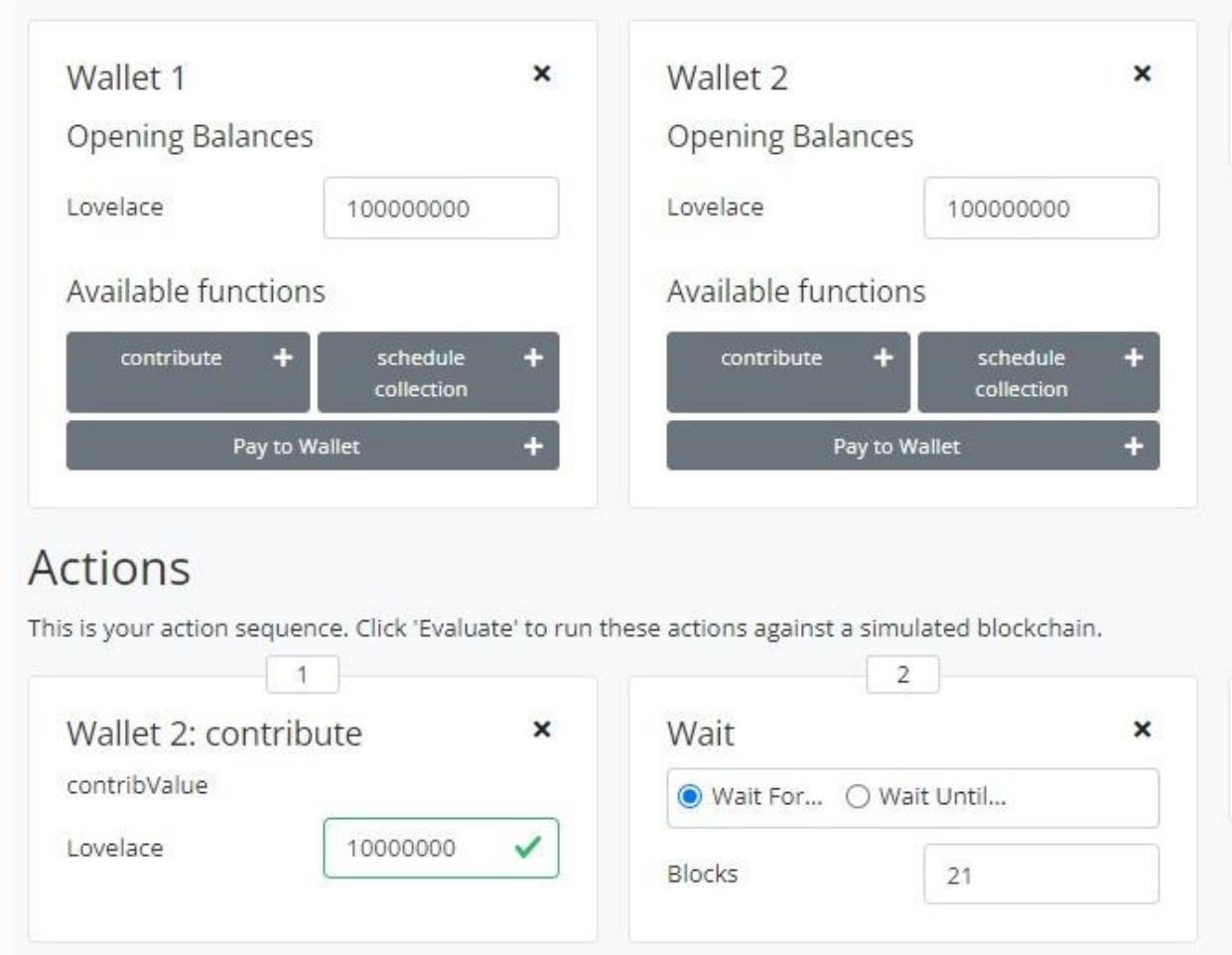

Рис. 3.20. Симуляція повернення коштів з 1 гаманцем учасником

Гаманець 2 заносить кошти в кампанію. Очікуємо 21 блок. Перша транзакція це генезис яка створює початкові кошти на гаманцях (див. рис. 3.21). Гаманець 2 вносить кошти в розмірі 10,000,000 Lovelace (див. рис. 3.22). і після очікування більше 20 блоків якщо власник кампанції не зібрав кошти відбувається відшкодування (див. рис. 3.23-24).

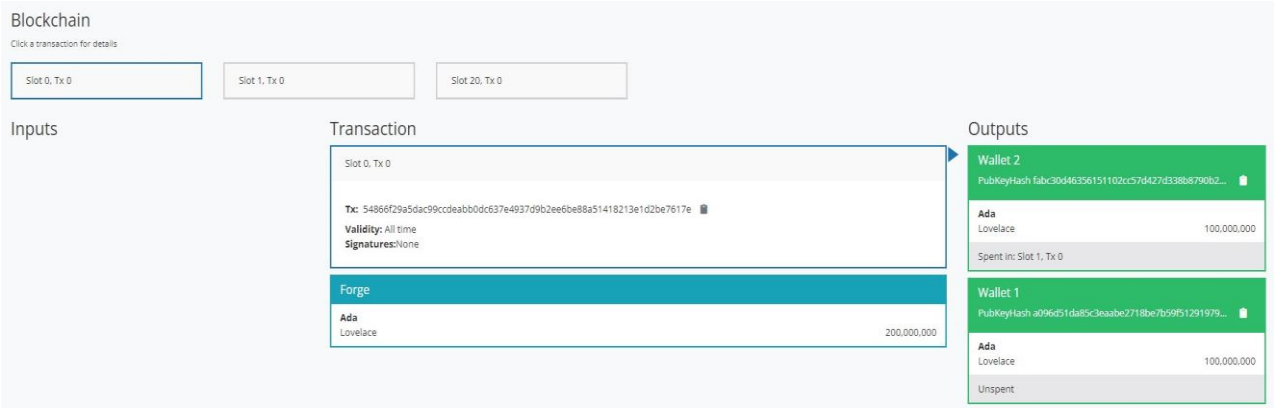

Рис. 3.21. Симуляція 2 генезис 0 слот

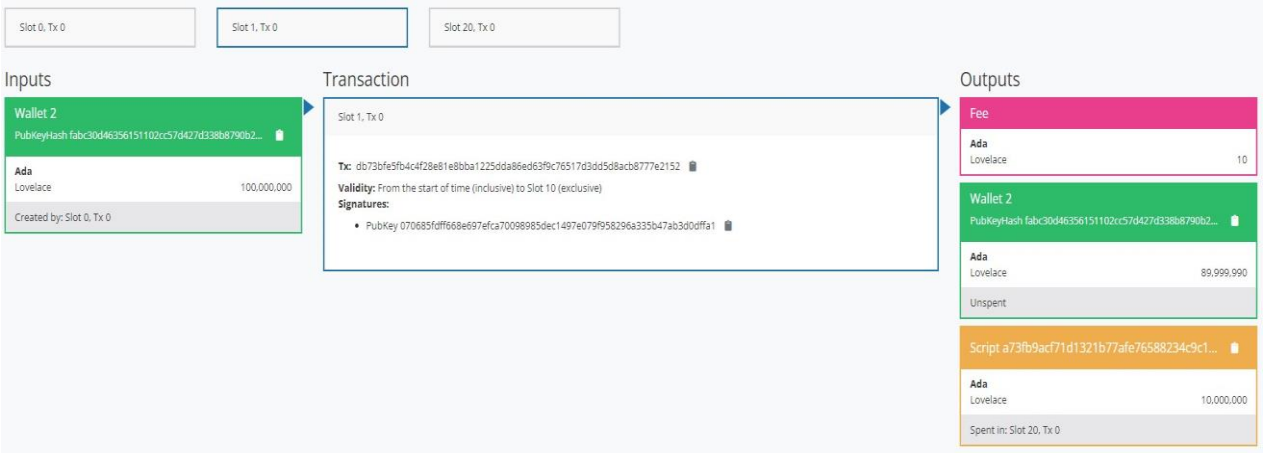

Рис. 3.22. Симуляція 2 слот 1 внесення коштів 2 гаманцем

Рис. 3.23. Симуляція 2 слот 20 повернення коштів 2 гаманцю

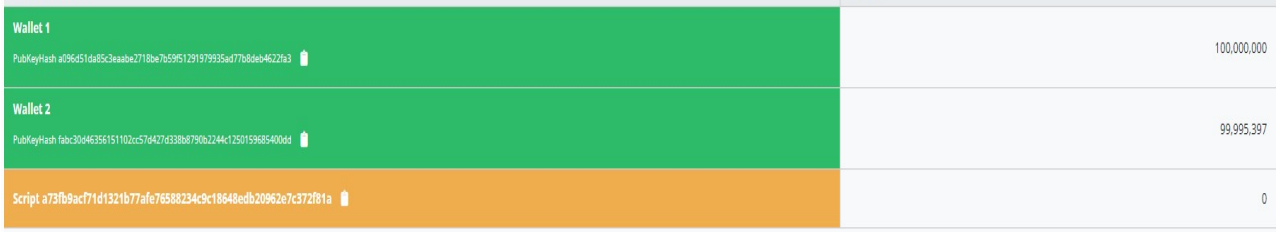

Рис. 3.24. Симуляція 2 слот 20 остаточний баланс

Оскільки власник кампанії не зібрав кошти в заданий період між слотом 10 і 20 то на 20 слоті скрипт автоматично повертає кошти учасника. По остаточному балансу видно, що 10,000,000 Lovelace з вирахуванням комісії за транзакції поступили назад на гаманець учасника кампанії.

3.4. Висновки до розділу

У даному розділі одержано наступні практичні результати:

- було розроблено смарт-контракт та програму Plutus для громадського фінансування (краудфандінг) в середовищі Plutus Playground;
- змодельвано варіанти проведення компанії;
- за результати моделювання було підтверджено коректність роботи смарт-контракту;
- експериментально доведено ефективність застосування запропонованої платформи для написання смарт-контрактів.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Тема кваліфікаційної роботи магістра присвячена дослідженню методів та засобів реалізації смарт-контрактів на блокчейні Cardano. Оскільки, розробка смарт-контрактів передбачає використання електронно-обчислювальної техніки, то важливим є дотримання вимог з охорони праці і техніки безпеки. Проаналізуємо основні правила і норми, яких необхідно дотримуватись при експлуатації комп'ютерів та периферійних пристроїв.

В загальному, поняття охорона праці в комп'ютерних системах являє собою дотримання всіх вимог і нормативів, що присутні в законодавчих актах про охорону праці. Закони цієї області спрямовані на якісну і безпечну експлуатацію робочих приладів і приміщень, дотримання санітарно-гігієнічних умов праці і захист від інших небезпечних чинників на підприємстві. В основних законодавчих актах про охорону праці приділяється велика увага поліпшенню умов праці в усіх галузях господарства, впровадженню сучасних засобів техніки безпеки і забезпечення санітарно-гігієнічних умов, що запобігають виробничому травматизму і професійним захворюванням.

Охорона життя і здоров'я людини є пріоритетним напрямком соціальної політики держави. В Україні прийнято закон прямої дії «Про охорону праці», який регламентує захист конституційного права працівників на безпечні умови праці. Законодавство України про охорону праці складається із загальних законів України та спеціальних законодавчих актів. Загальними законами України, що визначають основні положення з охорони праці є Конституція України, Закон України «Про охорону праці», Кодекс законів про працю (КЗпП), Закон України «Про загальнообов'язкове державне соціальне

страхування від нещасного 105 випадку на виробництві та професійного захворювання, які спричинили втрату працездатності».

Одним із найбільш важливих нормативних документів щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98. Дотримання даних правил значно знижує наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з 106 відео-дисплейними матеріалами, зокрема можливість зорових, нервовоемоційних переживань, серцево-судинних захворювань[20]. Виходячи з цього, роботодавець повинен забезпечити гігієнічні й ергономічні вимоги щодо організації робочих приміщень для експлуатації електронно-обчислювальних машин (ЕОМ) з ВДТ, робочого середовища, робочих місць з ЕОМ, режиму праці і відпочинку при роботі з ЕОМ тощо, які викладені у нормах НПАОП 0.00-7.15- 18.

При виконанні досліджень методів та засобів реалізації смарт-контрактів на блокчейні Cardano, які передбачали використання ПК, площа та об'єм для одного робочого місця оператора визначається згідно вимог НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», зокрема площа повинна становити не менше 6,0 м², об'єм - не менше 20,0 м³, відстань робочого місця від стіни повинна складати 1м, а відстань між робочими місцями повинна становити 1,7 м[19].

Згідно вимог охорони праці та державних санітарних правил, стіни, стеля та підлога приміщень, в яких розміщені ЕОМ, повинні бути виготовлені з матеріалів, дозволених для оформлення приміщень органами державного санітарно-епідеміологічного нагляду.

При виборі кімнат для розміщення робочих місць ПК враховано ступінь відбиття світла на екранах дисплеїв, яке проходить через вікна і яке може викликати значне осліплення в тих, хто сидить перед ними, особливо влітку та в сонячні дні. Тому, ПК і оргтехніка розміщені біля стін, які не знаходяться біля вікон або навпроти них.

Оскільки, при незадовільному освітленні знижується продуктивність праці користувачів ПК, і можливі негативні впливи на здоров’я такі, як короткозорість, швидка втомленість, тому всі приміщення, які облаштовані робочими місцями з ПК, мають природне і штучне освітлення. Не допускається розташування робочих місць з ПК в підвальних приміщеннях.

Штучне освітлення у приміщеннях повинно бути виконано у вигляді комбінованої системи освітлення з використанням люмінесцентних джерел світла у світильниках загального освітлення, які розташовувати над робочими поверхнями у рівномірно-прямокутному порядку. Штучне освітлення забезпечує на робочих місцях з ПК освітленість 300 – 500 Лк[20].

Для запобігання засвітленню екранів ПК прямими світловими потоками лінії світильників розташовані з достатнім бічним зміщенням відносно рядів робочих місць, а також паралельно до світлових отворів. При цьому кожне вікно повинно мати світлорозсіюючі штори з коефіцієнтом відбивання 0,7.

Отже, при дослідженні методів та засобів реалізації смарт-контрактів на блокчейні Cardano, проаналізовано та враховано необхідні вимоги щодо охорони праці при використанні електронно-обчислювальної техніки і забезпечено умови для зручної та ефективної роботи працівників.

4.2. Застосування положень концепції захисту населення і територій у разі загрози та виникнення надзвичайних ситуацій при напрацюванні заходів захисту працівників, матеріальних цінностей суб’єкта господарювання та населення.

Забезпечення захисту населення і територій у разі загрози та виникнення надзвичайних ситуацій, які згідно з класифікацією поділяються за характером на техногенні, природні, воєнні та соціально-політичні, а за рівнем - на загальнодержавні, регіональні, місцеві та об'єктові, є одним з найважливіших завдань держави.

Актуальність проблеми забезпечення природно-техногенної безпеки населення і територій зумовлена тенденціями зростання втрат людей і шкоди територіям, що спричиняються небезпечними природними явищами, промисловими аваріями і катастрофами. Ризики надзвичайних ситуацій природного і техногенного характеру невпинно зростають

Традиційна орієнтація системи цивільної оборони на вирішення завдань воєнного часу, її відомчий характер не дозволяли створити сталу організаційну 85 структуру, органи управління, сили і засоби, які сприяли б ефективному здійсненню заходів щодо захисту населення в сучасних умовах, наземного прикриття основних регіонів країни.

Політичні зміни, значна кількість великих катастроф, що сталися останнім часом на території України, серед яких особливе місце займає Чорнобильська, змінили попередню парадигму цивільної оборони на таку, що базується на визнанні пріоритету захисту населення і територій від загроз мирного часу і пошуку нової моделі такого захисту з урахуванням необхідності переходу від галузевого до функціонального принципу реагування на надзвичайні ситуації.

Захист населення і територій є системою загальнодержавних заходів, які реалізуються центральними і місцевими органами виконавчої влади, виконавчими органами рад, органами управління з питань надзвичайних ситуацій та цивільного захисту, підпорядкованими їм силами та засобами підприємств, установ, організацій незалежно від форм власності, добровільними формуваннями, що забезпечують виконання організаційних, інженерно-технічних, санітарно-гігієнічних, протиепідемічних та інших заходів у сфері запобігання та ліквідації наслідків надзвичайних ситуацій.

Загрози життєвоважливим інтересам громадян, держави, суспільства поділяються на зовнішні та внутрішні і виникають під час надзвичайних ситуацій техногенного і природного характеру та воєнних конфліктів

Зовнішні загрози безпосередньо пов'язані з безпекою життєдіяльності населення і держави у разі розв'язання сучасної війни або локальних збройних

конфліктів, виникнення глобальних техногенних екологічних катастроф за межами України (на землі, в навколоземному просторі), які можуть спричинити негативний вплив на населення та територію держави.

Внутрішні загрози пов'язані з надзвичайними ситуаціями техногенного і природного характеру або можуть бути спровоковані терористичними діями.

З метою захисту населення, зменшення втрат та шкоди економіці в разі виникнення надзвичайних ситуацій має проводитися спеціальний комплекс заходів. Він має такі складові:

- оповіщення та інформування;
- спостереження і контроль;
- укриття у захисних спорудах;
- евакуаційні заходи;
- інженерний захист: будувати будинки, споруди, інженерні мережі і транспортні комунікації з заданими рівнями безпеки і надійності, а також розробляти і запроваджувати заходи щодо безаварійного функціонування потенційно небезпечних об'єктів;
- медичний захист: заходами запобігання або зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідемічного благополуччя в районах надзвичайних ситуацій;
- біологічний захист: захист від біологічних засобів ураження включає своєчасне виявлення чинників біологічного зараження, їх виду і масштабів, проведення комплексу адміністративно-господарських, режимнообмежувальних і спеціальних протиепідемічних та медичних заходів;
- радіаційний і хімічний захист: включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного і хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки;

Отже, головною метою захисту населення і територій під час надзвичайних ситуацій є забезпечення реалізації державної політики у сфері запобігання і ліквідації їх наслідків, зменшення руйнівних наслідків терористичних актів та воєнних дій.

Основними завданнями у сфері захисту населення і територій від надзвичайних ситуацій є:

− здійснення комплексу заходів для запобігання надзвичайним ситуаціям техногенного та природного характеру та реагування на них;

− забезпечення готовності та контролю за станом готовності до дій і взаємодій органів управління у цій сфері, сил та засобів, призначених для запобігання надзвичайним ситуаціям техногенного та природного характеру і реагування на них;

ВИСНОВКИ

У кваліфікаційній роботі магістра досягнуто поставленої мети та розв’язано наступні задачі, зокрема:

− обґрунтовано та застосовано технології смарт-контрактів в електронній комерції;

− розглянуто концепцію смарт-контракту, проаналізовані платформи, що їх підтримують та вибрано платформу Cardano;

− було проведено аналіз архітектури та технологій блокчейну Cardano;

− описано модель програмування Plutus та основні компоненти смарт-контрактів блокчейну Cardano;

− експериментально доведено ефективність застосування платформи Plutus для написання смарт-контрактів.

Cardano розроблений як блокчейн третього покоління з метою вирішити проблеми масштабованості. Даний блокчейн здатний обробляти велику кількість транзакцій, не впливаючи на продуктивність мережю. Також розробники Cardano працють над впровадженням оновлення "Hydra ", яке дозволить функціонування декількох бічних ланцюгів.

Смарт-контракт – це автоматизована цифрова угода, написана в коді, яка відстежує, перевіряє та виконує обов’язкові транзакції контракту між різними сторонами. Це сучасна і прогресивна технологія яка може допомогти вирішити велику кількість завдань в комерційній галузі.

Вважаю, що написання смарт-контрактів на базі блокчейна Cardano є технологічно і економічно хорошим рішенням для використання в електронній комерції.

Проаналізовано практичну цінність смарт-контрактів, і результати підтверджують актуальність цієї технології на сьогоднішній день. Були також прийняті до уваги недоліки розумних контрактів, найважливішим з яких є відсутність законодавчої бази в більшості країн.

Plutus – це смарт-контрактна платформа блокчейну Cardano. Вона дозволяє писати програми, які взаємодіють з блокчейном Cardano. Plutus забезпечує новий інтегрований підхід до розробки смарт-контрактів і розподілених додатків, який є зручнішим і безпечнішим, ніж попередні альтернативи.

Практичними результатами роботи є розроблений смарт-контракт та програма Plutus для громадського фінансування (краудфандінг). Програма приймає кошти від учасників кампанії, дає проміжок часу власнику кампанії щоб зібрати кошти і якщо кошти не було зібрано то повертає їх учасникам кампанії.

Програма була розроблена в Plutus Playground – середовище для написання та тестування смарт-контрактів на блокчейні Cardano. Plutus Core, яка є мовою смарт-контрактів, вбудована в реєстр, заснована на формальних принципах програмування Haskell і дозволяє розробникам писати програми з високою надійністю, які взаємодіють з Cardano.

Вважаю, що подальше використання смарт-контрактів набуде особливої актуальності, оскільки, як зазначено в даній кваліфікаційній роботі, ці технології можуть замінити існуючу практику паперових договорів водночас з забезпеченням безпеки, надійності та швидкості децентралізованої мережі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Documentation for the Cardano ecosystem URL: https://docs.cardano.org/introduction (дата звернення:10.10.2021).

2. Functional Blockchain Contracts URL: https://iohk.io/en/research/library/ papers/functional-blockchain-contracts/

3. Plutus: what you need to know URL: https://iohk.io/en/blog/posts/ 2021/04/13/plutus-what-you-need-to-know/ (дата звернення:10.10.2021).

4. The Plutus Platform and Marlowe URL: https://plutus-apps. readthedocs.io/ (дата звернення:10.10.2021).

5. PLUTUS PLAYGROUND URL: https://playground.plutus.iohkdev.io/ (дата звернення:10.10.2021).

6. Cardano Developer Portal URL:https://developers.cardano.org/ (дата звернення:10.10.2021).

7. Що таке смарт-контракт? URL: https://bankchart.com.ua/finansoviy _gid/ investitsiyi/statti/scho_take_smart_kontrakt_ (дата звернення:10.10.2021).

8. Comparing Ethereum (ETH) vs. Cardano (ADA) Blockchains URL: https://academy.shrimpy.io/post/comparing-ethereum-eth-vs-cardano-ada-blockchains (дата звернення:10.10.2021).

9. Discover Cardano URL: https://cardano.org/

10. Real-World Use Cases for Smart Contracts and dApps URL: https://www.gemini.com/cryptopedia/smart-contract-examples-smart-contract-use cases (дата звернення:10.10.2021).

11. What are smart contracts on blockchain? URL: https://www.ibm.com/ topics/smart-contracts (дата звернення:10.10.2021).

12. What are Smart Contracts? URL: https://corporatefinanceinstitute.com/ resources/knowledge/deals/smart-contracts/ (дата звернення:10.10.2021).

13. Introducing the new Plutus Playground URL: https://iohk.io/en/blog/posts /2021/01/25/introducing-the-new-plutusplayground/ (дата звернення: 10.10.2021).

14. Proof of Stake, Proof of Work URL: https://www.bitdegree.org/tutorials/proof-of-work-vs-proof-of-stake/

15. Що таке Смарт-Контракти URL: https://cryptobook.pro/shcho-take-smart-kontrakty.html. (дата звернення:10.10.2021).

16. What is Blockchain URL: https:// www.ibm.com/blockchain/what-is-blockchain. (дата звернення:10.10.2021).

17. Блокчейн технологія і платформа URL: https://uacrypto.top/blog/ blockchain-guide.

18. CARDANO ROADMAP URL: https://roadmap.cardano.org/en/

19. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Київ. 2018.

20. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

21. Соленко С.В. Жаровський Р.О. Можливості середовища Plutus Playground для написання та тестування смарт-контрактів *Інформаційні моделі, системи та технології* : Матеріали IX наук.-техн. конф. ТНТУ ім. І.Пулюя (8-9 грудня 2021). Тернопіль, 2021. с. 203.

22. Соленко С.В. Жаровський Р.О. Використання smart-контрактів на базі блокчейна Cardano в електронній комерції *Інформаційні моделі, системи та технології*: Матеріали IX наук.-техн. конф. ТНТУ ім. І.Пулюя (8-9 грудня 2021). Тернопіль, 2021. с. 203.

23. Дацко М.В. Артим-Дрогомирецька З.Б. Технологія блокчейн та перспективи її застосування *Моделювання економіки: проблеми, тенденції, досвід :* Матеріали VIII Міжнародної науково-методичної конференції Форум молодих економістів-кібернетиків ТНТУ ім. І.Пулюя (28-29 вересня 2017). Львів, 2017. с. 192.

24. В.О. Дармограй А. М. Луцків Аналіз бібліотек для реалізації Blockchain-інфраструктури для систем IoT *Актуальні задачі сучасних технологій :* Матеріали VIII Міжнародної науково-технічної конференція молодих учених та студентів Том 2 ТНТУ ім. І.Пулюя (28 - 29 листопада 2018) Тернопіль, 2018. с. 132.

25. О.С. Тхір Реалізація smart-контрактів для страхової компанії на платформі Ethereum *Актуальні задачі сучасних технологій :* Матеріали VIII Міжнародної науково-технічної конференція молодих учених та студентів с.213.

Додаток А

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

ІХ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.658.114
С.В. Соленко, Р.О. Жаровський канд. техн. наук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

МОЖЛИВОСТІ СЕРЕДОВИЩА PLUTUS PLAYGROUND ДЛЯ НАПИСАННЯ ТА ТЕСТУВАННЯ СМАРТ-КОНТРАКТІВ

UDC 004.658.114
S. Solenko, R. Zharovskyi Ph.D.

PLUTUS PLAYGROUND ENVIRONMENT CAPABILITIES FOR WRITING AND TESTING SMART CONTRACTS

Платформа Plutus – це платформа для написання додатків, які взаємодіють із розподіленим реєстром із можливостями написання сценаріїв, зокрема блокчейном Cardano.

Практичними результатами роботи є розробка smart-контракту та програми Plutus для громадського фінансування (краудфандінг). Програма приймає кошти від учасників кампанії, дає проміжок часу власнику кампанії щоб зібрати кошти і якщо кошти не було зібрано то повертає їх учасникам кампанії.

Розроблятися програма буде в середовищі Plutus Playground. Plutus Playground забезпечує середовище для написання та тестування смарт-контрактів перед їх випуском на блокчейні Cardano. Plutus Core, яка є мовою smart-контрактів, вбудована в реєстр, заснована на формальних принципах програмування Haskell і дозволяє розробникам писати програми з високою надійністю, які взаємодіють з Cardano. Haskell був обраний як основа для платформи Plutus, оскільки він виділяється серед інших мов програмування тим, що пропонує можливість писати більш безпечний код. Використання Haskell для розгортання smart-контрактів гарантує, що контракти запрограмовані на те, що від них очікується, і їх можна перевірити на точність перед впровадженням.

Є два результати кампанії.

1. Власник кампанії збирає кошти від обох учасників. У цьому випадку власник створює одну транзакцію з двома входами, посилаючись на "t_1" і "t_2". Кожен вхід містить скрипт `contributionScript c`, спеціалізований для учасника. Сценарій викупу цієї транзакції містить значення `Collect`, що спонукає сценарій перевірки перевірити гілку на наявність `Collect`.
2. Відшкодування. У цьому випадку кожен вкладник створює транзакцію з одним введенням, вимагаючи повернення своєї частини коштів. Цей випадок охоплює гілка відшкодування "Refund", а його сценарієм викупу є дія відшкодування "Refund".

В обох випадках сценарій перевірки виконується двічі. У першому випадку існує одна транзакція, яка споживає обидва входи. У другому випадку є дві різні транзакції, які можуть відбутися в різний час.

Ouroboros протокол розділяє час на епохи. Кожна епоха Cardano складається з кількох слотів, де кожен слот триває одну секунду.

Наступна послідовність дій (див. рис. 1) призводить до успішної кампанії.

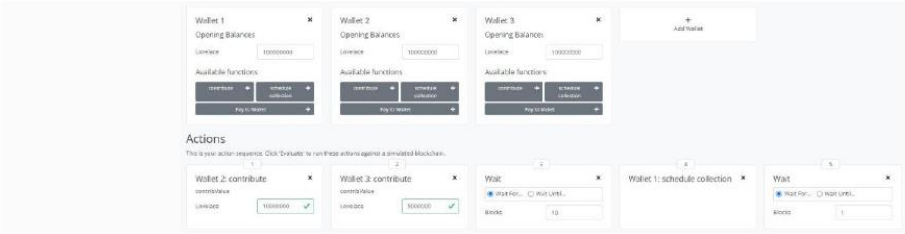

Рисунок 1. Симуляція успішної кампанії з 2 гаманцями учасниками

Гаманець 2 та 3 заносять кошти в кампанію. Після очікування в 10 блоків власник кампанії має право зібрати кошти, що виконується через кнопку «schedule collection». Чекаємо ще 1 блок щоб виконалася транзакція.

Перша транзакція це генезис яка створює початкові кошти на гаманцях (див. рис. 2). Гаманець 2 вносить кошти в розмірі 10,000,000 Lovelace, а гаманець 3 в розмірі 5,000,000 Lovelace (див. рис. 3). Власник кампанії збирає кошти в 11 слоті (див. рис. 4–5).

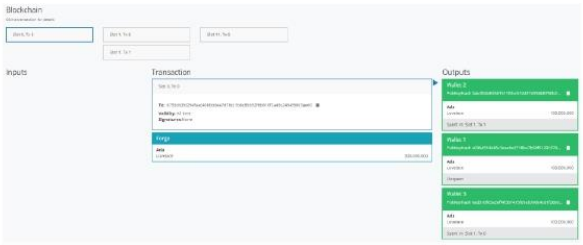

Рисунок 2. Слот 0 генезис

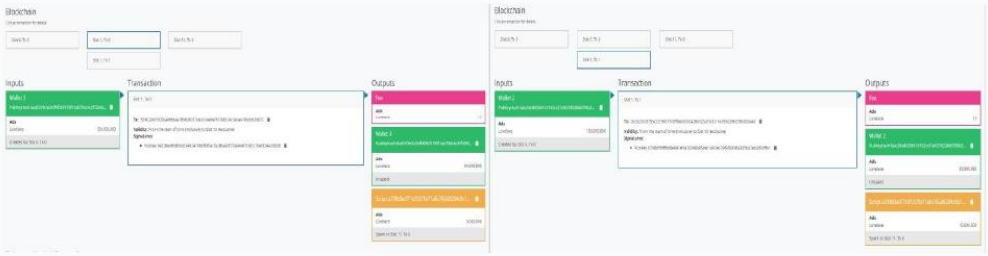

Рисунок 3. Слот 1 внесення коштів 2 та 3 гаманцем

Рисунок 4. Слот 11 збір коштів власником кампанії

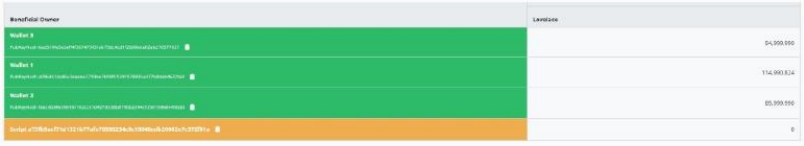

Рисунок 5. Слот 11 остаточний баланс

Як бачимо кампанія пройшла успішно і 15,000,000 Lovelace з вирахуванням комісії за транзакції поступили на гаманець власника кампанії.

Smart-контракти це сучасна і прогресивна технологія яка може допомогти вирішити велику кількість завдань в комерційній галузі. Вважаю, що написання smart-контрактів на базі блокчейна Cardano є технологічно і економічно хорошим рішенням для використання в електронній комерції. В роботі буде проведено аналіз архітектури та технології блокчейну Cardano і розглянуті методи та засоби реалізації smart-контрактів на базі даного блокчейну.

УДК 004.658.114
С.В. Соленко, Р.О. Жаровський канд. техн. наук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ВИКОРИСТАННЯ SMART-КОНТРАКТІВ НА БАЗІ БЛОКЧЕЙНА CARDANO В ЕЛЕКТРОННІЙ КОМЕРЦІЇ

UDC 004.658.114
S. Solenko, R. Zharovskyi Ph.D.

USE OF SMART-CONTRACTS BASED ON CARDANO BLOCKCHAIN IN ELECTRONIC COMMERCE

Smart-контракт (розумний контракт) – це автоматизована цифрова угода, написана у вигляді цифрового коду, яка відстежує, перевіряє та виконує обов'язкові транзакції контракту між різними сторонами. Транзакції контракту автоматично виконуються кодом Smart-контракту, коли виконуються заздалегідь визначені умови. По суті, smart-контракт – це коротка програма, вхідні та вихідні дані якої є транзакціями в блокчейні.

Технологія блокчейн, інакше відома як технологія розподіленої книги (DLT), забезпечує децентралізовану та доступну структуру даних для різних записів. Такі записи можуть включати інформацію про фінансові платежі та транзакції, а також інші типи інформації – від комерції до записів Інтернету речей (IoT). Важливою особливістю блокчейну є те, що він зберігає записи незмінно, що означає, що їх не можна змінити, підробити або видалити, оскільки це порушить ланцюжок записів. Блокчейни не лише забезпечують незмінну та безпечну базу даних, але й діють як функціональне середовище для трансакцій коштів, створення цифрових валют та обробки складних угод за допомогою цифрових угод (smart-контрактів).

Smart-контракти є самовиконуваними, надійними та не вимагають дій чи присутності третіх сторін. Код smart-контракту зберігається і поширюється в децентралізованій мережі блокчейні.

На рисунку 1 зображено переваги smart-контрактів.

Рисунок 1. Переваги Smart-контрактів

Прообразом smart-контрактів є звичайні паперові контракти, які використовує у своїй діяльності будь-яка сучасна організація. Після складання такі контракти зазвичай вручну підписуються, і далі учасники особисто виконують всі їх положення.

На рисунку 2 зображено графічне представлення процедури роботи smart-контракту.

Рисунок 2. Життєвий цикл Smart-контракту

Найбільш очевидним розвитком цих процесів могли б стати технології, які, по-перше, дозволили б долати відстані за лічені секунди, а по-друге, автоматизували б основні умови угоди. Контракт, в такому випадку, почав би виконуватися автоматично, без фізичної присутності сторін. Чим більшим є розвиток технологій блокчейн, тим краще smart-контракти підходять для цього рішення.

Приклади використання smart-контрактів зображено на рисунку 3.

Рисунок 3. Приклади використання Smart-контрактів

Cardano – це блокчейн-проект із відкритим вихідним кодом, який розпочав у 2015 році для вирішення існуючих проблем блокчейну в розробці та моделі криптовалют. Він спрямований на забезпечення більш збалансованої та стійкої екосистеми, яка краще враховує потреби своїх користувачів, а також інших систем, які прагнуть інтеграції. Cardano впроваджує підтримку Smart-контрактів у 2021 році.

Plutus – спеціально створена платформа для розробки та виконання Smart-контрактів на блокчейні Cardano. Контракти Plutus складаються з частин, які виконуються на блокчейні (код у мережі), і частин, які виконуються на машині користувача (код поза ланцюгом або клієнт). Plutus спирається на дослідження сучасних мов, щоб забезпечити безпечне середовище програмування з повним стеком на основі Haskell, провідної функціональної мови програмування.

Вважаю, що подальше використання smart-контрактів набуде особливої актуальності, оскільки, як зазначено вище, даний елемент системи електронного документообігу має ряд переваг над традиційним, паперовим методом. В роботі буде проведено аналіз архітектури та технології блокчейну Cardano і розглянуті методи та засоби реалізації smart-контрактів на базі даного блокчейну.