МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему

Система для проведення благодійних NFT аукціонів

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Комп'ютерні науки»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи КН-41
Ковальчук М. В, _____
(прізвище та ініціали)

Керівник: Кіктєв М.О. _____
(прізвище та ініціали)
к.т.н., доцент
_____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *інтелектуальних технологій*
Протокол No 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2022 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Ковальчука Миколи Валентиновича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): «Система для проведення благодійних NFT аукціонів» затверджена протоколом засідання кафедри від «23» грудня 2021 р. № 4
2. Термін здачі студентом закінченого проекту (роботи) 31 травня 2022 року
3. Вихідні дані до проекту (роботи):
   Розгорнута прозора система для проведення благодійних NFT аукціонів.
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) Аналіз предметної області, розробка архітектури, аналіз існуючих підходів, вибір стеку технологій та мови програмування, розробка смарт-контрактів та мануальне тестування з аналізом розробленої системи.
5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій) Презентація, яка містить короткі відомості про задачу, інформацію про використані технології та підходи, архітектуру системи, програмну реалізацію, демонстрацію роботи системи.

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Кіктєв М.О.	01.03.2022	01.03.2022
2	Кіктєв М.О.	05.04.2022	05.04.2022
3	Кіктєв М.О.	10.05.2022	10.05.2022

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Кіктєв М. О. /

(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Ковальчук М.В. /

(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Обговорення постановки завдання та змісту пояснювальної записки	25.01.2022 – 15.02.2022	
2	Вибір та формування теми дипломної роботи	16.02.2022 – 20.02.2022	
3	Аналіз предметної області	21.02.2022 – 10.03.2022	
4	Розробка архітектури системи	11.03.2022 – 25.03.2022	
5	Вибір основних технологій та мови програмування	26.03.2022 – 05.04.2022	
6	Розробка смарт-контрактів	06.04.2022 – 01.05.2022	
7	Тестування та аналіз розробленої системи	01.05.2022 – 11.05.2022	
8	Оформлення пояснювальної записки	12.05.2022 – 02.06.2022	

Студент-дипломник _____ / Ковальчук М.В. /

(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Кіктєв М.О. /

(підпис) (ПІБ)

Анотація

Ковальчук Микола Валентинович виконав випускну кваліфікаційну роботу на тему «Створення смарт-контракту для проведення благодійних NFT аукціонів» за спеціальністю 122 - «Комп’ютерні науки».

Дипломну роботу було виконано на 63 аркушах, вона містить в собі один додаток з лістингом програм та перелік посилань на використані джерела з 13 найменувань. Також в роботі наведено 30 рисунків.

Метою цієї дипломної роботи було створення системи для проведення благодійних NFT аукціонів, яка є повністю прозорою, безпечною та гнучкою зручною у використанні, щоб залучити більше людей до благодійної діяльності.

Було проведено аналіз існуючих варіантів надання благодійної допомоги у вигляді криптовалюти, виділені основні переваги та недоліки. Спроектовано та розроблено систему, яка вирішує ці проблеми з відповідним функціоналом.

Ключові слова: NFT, криптовалюта, смарт-контракт, благодійність, благодійні аукціони, блокчейн.

Summary

Kovalchuk Mykola Valentynovych completed the final qualifying work on "Creating a smart contract for NFT charity auctions" in specialty 122 - "Computer Science".

The thesis was completed on 63 sheets, it contains one appendix with a list of the program and a list of references to the sources used from 13 titles. Also in the work are 30 drawings.

The aim of this thesis was to create a system for conducting NFT charity auctions, which is completely transparent, secure and flexible, easy to use to attract more people to charity.

The analysis of the existing options for providing charitable assistance in the form of cryptocurrency was carried out, the main advantages and disadvantages were highlighted. A system has been designed and developed that solves these problems with the appropriate functionality.

Keywords: NFT, cryptocurrency, smart contract, charity, charity auctions, blockchain.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ETH – Ethereum cryptocurrency.

NFT – non-fungible token.

UXTO – Unspent transaction output.

PoW – Proof of Work.

EVM – Ethereum Virtual Machine.

DAO – Decentralized Autonomous Organisation.

ВСТУП

Благодійність є добровільною сферою діяльності, що направлена на допомогу іншим особам, державі, підприємствам тощо. Особами, що надають добровільну допомогу можуть бути окремі фізичні особи або компанії. Враховується будь-яка допомога, така як, наприклад, допомога послугами, матеріальна допомога, фінансова або надання компанією деяких потрібних товарів.

Зазвичай, люди які мають намір зайнятись благодійністю, просто не мають необхідної інформації про те, як це правильно зробити. А в багатьох випадках люди сумніваються в прозорості та чесності платформ для благодійності. Багато ресурсів, які використовують тільки переказ на рахунок коштів, що не створює у людей відчуття того, що вони беруть участь у благодійності, так як бачать куди саме вони йдуть.

За даними інструменту пошуку гаманців Blockchain.com, два криптовалютні гаманці отримали загалом близько 10,3 мільйона доларів США за перші чотири дня після повномасштабного вторгнення росії в Україну з яких мільйони вже були переведені з гаманців, імовірно урядом України. Також, на церемонії вручення премії «Оскар» наприкінці березня голлівудська хвилина мовчання для жителів України перерізала рекламу Crypto.com, яка спрямовувала глядачів на сайт, де вони могли пожертвувати криптовалюту, купити токени, що не можна обмінювати, або використати картку. Усі кошти були спрямовані на підтримку зусиль Червоного Хреста з надання допомоги.

Тому необхідно було створити повністю безпечну та прозору систему засновану на блокчейні, яка б дозволяла людям брати участь у благодійних NFT аукціонах, оскільки це дуже добре розвинена практика в усьому світі і виявилася дуже ефективним способом залучення громадян до допомоги. людей, які потребують.

РОЗДІЛ 1. АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ СИСТЕМИ БЛАГОДІЙНИХ NFT АУКЦІОНІВ

1.1 Аналіз актуальності благодійності в криптосфері

NFT, або non-fungible token, - це розрахункова одиниця, за допомогою якої можна створити цифровий зліпок для будь-якого унікального предмета. Це може бути будь-який контент, який претендує на хоч якусь унікальність: картини, фотографії, відео, музика, гіфки. Вони є великою цінністю серед колекціонерів, любителів мистецтва та геймерів, а купують та продають їх через аукціони.

Так як це цифровий актив (токен), випущений на блокчейн платформі, то, хоч це і електронний запис у базі даних, але завдяки особливим математичним захистам та порядку роботи блокчейн-технології, ці електронні записи перетворюються на річ, яку можна передавати між власниками. так само як ми передаємо звичайні речі з рук до рук. Тільки замість рук у блокчейні використовуються рахунки.

Рідкісність і непоновлюваність NFT мають на увазі, що з часом, цей рідкісний актив тільки дорожчатиме. Звідси є надія у благодійника, що в майбутньому він, можливо, отримає ще й дохід від перепродажу своєї рідкісної, хоч і цифрової речі.

Спільним NFT і криптовалютою є їх значне зростання вартості за короткий період. Люди часто жертвують свій найбільш цінний актив, а для більш молодих інвесторів цим активом є біткойн, ефіріум або NFT.

В основному випускають NFT саме художники, співаки, фан-клуби тощо. Продаючи ці токени, вони в першу чергу отримують донат. До того ж на деяких майданчиках, кожен наступний перепродаж, також досить відчутний відсоток віддає автору NFT. Чим це не є аналог благодійних аукціонів? Наважусь припустити що скоро NFT почнуть випускати звичайнісінькі благодійні фонди і навіть церкви!

Факт здійснення благодійності при покупці NFT виражений самим цим токеном, який переходить на рахунок благодійника. І хоч цей рахунок

анонімний, але все ж таки його власник може похвалитися у вузькому колі своїх друзів і знайомих розкривши їм факт володіння цим рахунком. До того ж, в деяких блокчейн середовищах, наприклад блокчейн Erachain, можна взагалі публічно прив'язати свій рахунок до цифрового профілю персони і тоді весь світ дізнається, що саме ви є меценатом!

Благодійне використання NFT, яке найбільше подобається благодійним організаціям і художникам NFT, полягає в продажі цифрових робіт і націлювання певного відсотка виручених коштів на благодійність. Розумні контракти в NFT можуть бути закодовані таким чином, що кожен наступний продаж цього NFT продовжує спрямовувати відсоток від продажі на благодійність. Проте важко пожертвувати NFT безпосередньо благодійній організації, оскільки вартість подарованого майна потребує незалежної оцінки. Часто важко визначити, скільки коштує цифровий витвір мистецтва, закодований печаткою автентичності, правильно буде влаштовувати аукціони на такі пожертви.

Продажі NFT минулого року досягли 25 мільярдів доларів, за даними DappRadar, аналітичної компанії, що займається блокчейном, а цьогорічні продажі вже підскочили до 27 мільярдів доларів до 10 березня. Проте були ознаки того, що ринок охолов. Не всім криптодонорам 20-30 років. Можливо, найбільшим крипто-пожертвуванням на сьогоднішній день був подарунок у розмірі 10 мільйонів доларів «від анонімного джентльмена у віці 70 років».

Дослідження показують, що аукціони, де виручені кошти передаються на благодійність, призводять до значно вищих відпускних цін, що є результатом більшої частки благодійно мотивованих учасників, а не збільшення кількості учасників торгів.

Крім того, аукціони, які жертвують 25% доходів на благодійність, мають вищий чистий дохід, ніж аукціони, що не належать до благодійних. У результаті компанії можуть використовувати благодійні аукціони як частину своєї стратегії корпоративної соціальної відповідальності, одночасно підвищуючи прибутковість, навіть якщо вони віддають частину виручених коштів на

благодійність. Запорукою успіху на аукціоні є момент хвилювання. Всі ми знаємо, що всі люблять перемагати. У разі благодійного аукціону ми можемо використати його на добру справу.

На благодійних аукціонах перемога в торгах приносить користь добрій справі, яку, ймовірно, оцінять як учасники торгів, так і учасники-конкуренти. Таким чином, учасник лоту отримує вигоду від власної оплати – вартості виграного проекту та наданої допомоги – так само, як і інші учасники торгів, оскільки їх благодійні організації підтримуються. Отже, учасники торгів мають дві основні цілі: виграти те, що вони цінують, і підтримати благодійність, частково підвищивши ціну, яку вони готові заплатити.

Тому система благодійних аукціонів може стати ще одним інструментом збору коштів для благодійних фондів, щоб залучити більше людей до благодійної діяльності.

Мета створення інтелектуального застосунку

Мета дипломної роботи – розробити повністю прозору, зручну, інтуїтивно зрозумілу та просту у використанні систему, щоб залучити більше людей до благодійності. Система має надати можливість проводити благодійні NFT аукціони для збору коштів для благодійних організацій по всьому світу.

Щоб цього досягти, потрібно:

- Проаналізувати існуючі системи благодійної діяльності, які дають можливість брати участь у аукціонах.
- Розробити систему проведення благодійних аукціонів з урахуванням недоліків існуючих систем.
- Виберіть технологічний стек, необхідний для впровадження.
- Реалізація необхідних функцій за допомогою смарт-контрактів.
- Тестувати та аналізувати розроблені системи.

1.2 Аналіз існуючих способів надання благодійної допомоги

У ході свого дослідження та аналізу наявних способів долучитись до благодійності онлайн, я знайшов сайти благодійних фондів та NFT аукціонів, на яких є можливість зробити пожертву.

Наприклад, нижче наведені одні з найпопулярніших:

- **«Binance.charity»** - благодійний фонд найбільшої криптобіржи Binance.

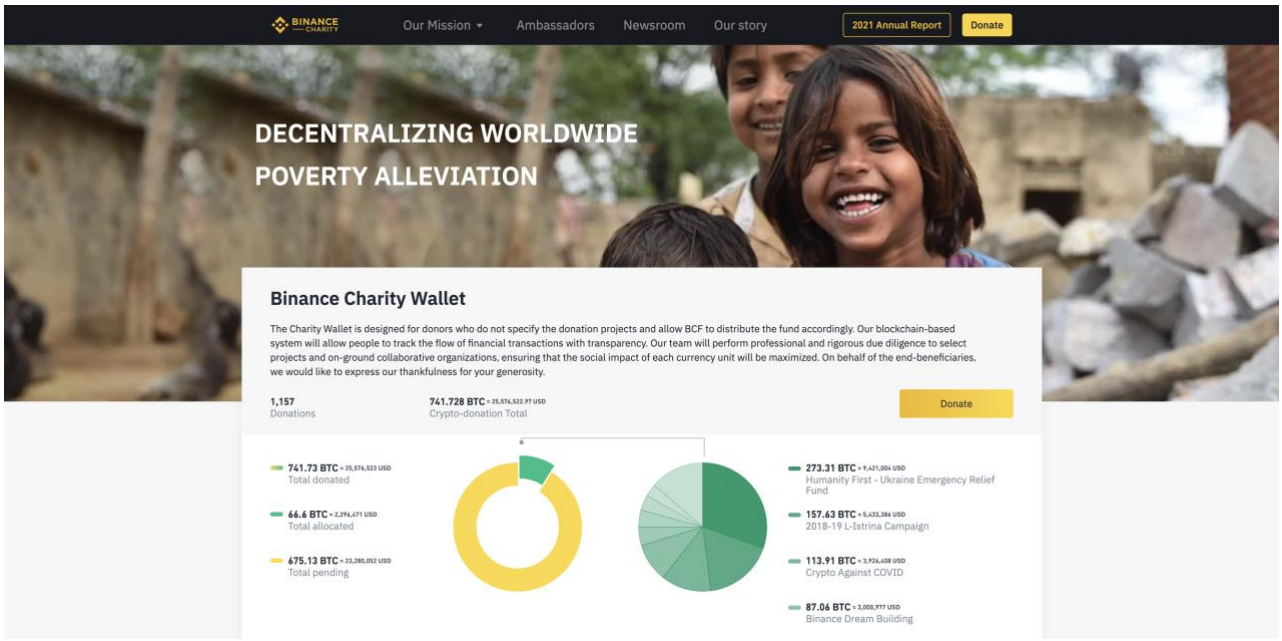


Рисунок 1.1 – Інтерфейс сайту « Binance.charity»

Найбільша криптобіржа Binance створила власний благодійний фонд та платформу для збору пожертв у криптовалюті. Користувачі можуть пожертвувати будь-яку суму в криптовалюті, а Binance подвоїть її та направить на вирішення соціально значущих проблем. У 2020 році фонд Binance.Charity залучив понад $6 млн на боротьбу з коронавірусом та ще понад $2 млн на боротьбу з наслідками лісових пожеж в Австралії.

Особливу підтримку Binance.Charity надає країнам Африки. Фонд залучав кошти на закупівлю швейних машин для матерів Кенії, закупівлю корисної їжі для дітей цього континенту, брав участь у фінансуванні відновлення згорілого собору Нотр-Дам у Парижі.

У перші дні повномасштабного вторгнення, Binance створила «Фонд екстреної допомоги» для надання негайної підтримки та безпечного прихистку біженцям подалі від зон конфлікту. Пожертвування Binance у розмірі $10 млн — одне з найбільших пожертвувань, зроблених приватною компанією для України і Binance продовжує надавати допомогу.

- **«Giving Block»**

Рисунок 1.2 — Інтерфейс сайту «Giving Block»

The Giving Block — це рішення №1 для крипто-пожертв, яке забезпечує екосистему для некомерційних та благодійних організацій для збору коштів біткойн та інших криптовалют, легкий шлях криптодонорами для миттєвого внесення коштів на благодійність і стати частиною мережі крипто-медіа-партнерів для підтримки їхніх місій.

Giving Block , працює з 1500 некомерційними організаціями і обробив понад 100 мільйонів доларів у вигляді пожертвування цифрових активів, переважно за останній рік. Серед перших користувачів був «Save the Children», який зібрав понад 5 мільйонів доларів у вигляді крипто-пожертв за допомогою своєї кампанії «HodlHope».

- **«Opensea.io»**

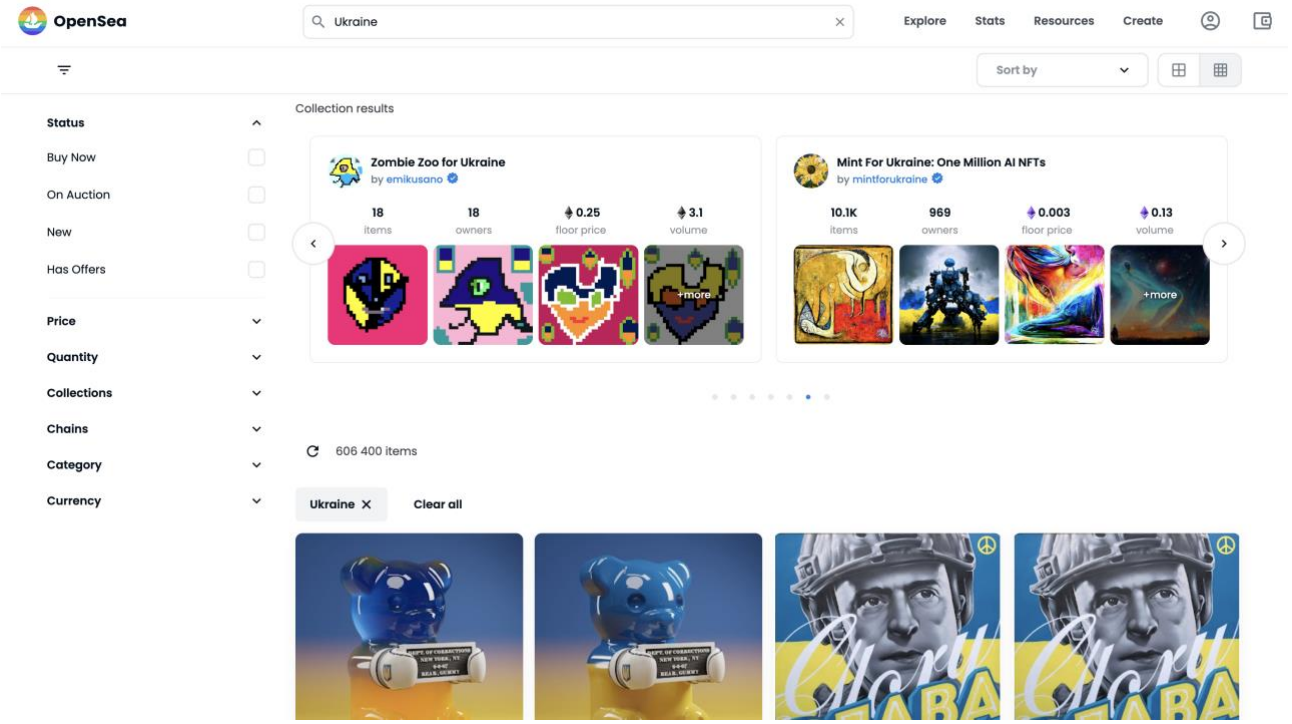


Рисунок 1.3 — Інтерфейс сайту « Opensea.io»

OpenSea – перший NFT-маркетплейс, запущений у 2017 році. Платформа працює за принципами децентралізованої спільноти: криптовалютні гаманці, адреси та активи контролюються самими користувачами. Протокол маркетплейсу підтримує блокчейн Ethereum, Polygon та Klaytn Blockchain.

Протокол OpenSea NFT використовує токени стандарту ERC721 та ERC1155 (для колекцій). Сплачувати за покупки можна будь-якими токенами ERC20. Купити NFT за банківською карткою можна через сервіс MoonPay.

Особливість OpenSea – широкий вибір категорій. Популярні категорії: мистецтво, спорт, колекційні картки та проекти від відомих артистів. Також на майданчику можна купити NFT у вигляді аудіо, доменного імені чи 3D-об'єкта.

Стандартна комісія OpenSea – 2,5% вартості NFT. Комісія стягується з покупця. Автор токена оплачує газ транзакції тільки при першому розміщенні.

- **«Crypto.com»**

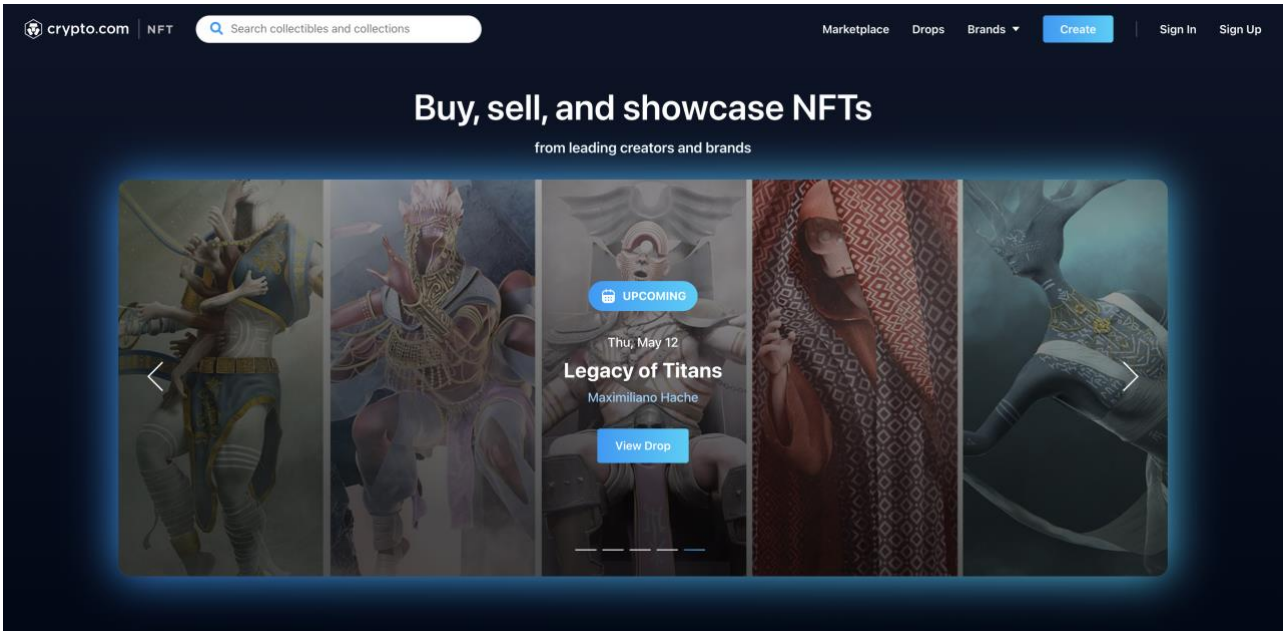


Рисунок 1.3 — Інтерфейс сайту «Crypto.com»

Компанія Crypto.com зібрала 1,6 мільйона доларів у вигляді пожертв, 60% з яких надійшло від продажів NFT, які артисти створили для цієї мети, плюс додатково 1 мільйон доларів від компанії, головний маркетинговий директор сінгапурської компанії. NFT спрямовують більше половини пожертв, деякі покупці зробили цифрові ілюстрації своїми аватарами в соціальних мережах, щоб транслювати свою участь у зборі коштів.

Переваги знайдених способів долучення до благодійності:

- Хороша звітність.
- Зрозумілі користувацькі інтерфейси.
- Не складно знайти, як зробити пожертвування.

Недоліки:

Зробивши аналіз всіх знайдених аналогів я знайшов єдиний, на мою думку, недолік. Ці інструменти хороші, але вони не дають повної гнучкості в виборі фонду для пожертви та свободи власника лоту створювати свої умови для проведення аукціону.

1.3 Постановка задачи розробки системи

Необхідно створити систему з проведення благодійних NFT аукціонів для заохочення людей прийняти участь в благодійності в розважальному вигляді. Створена система повинна, об’єднувати в собі інформацію про благодійні організації й актуальні потреби людства та проведення NFT аукціонів, комісія з яких буде йти на благодійність.

Вимоги до благодійної системи

Функціональні вимоги:

- можливість приймати участь в аукціоні абсолютно анонімно.
- можливість переглянути лоти, доступні у даний момент, та прийняти участь в аукціоні, тобто зробити ставку на обраний лот.
- можливість встановити відсоток на пожертву від продажу лоту.
- пожертвувати гроші користувачеві без участі в аукціоні.
- додавання, редагування та видалення лотів.
- відображення панелі прогресу, що показує відношення коштів, які уже зібрано.

Нефункціональні вимоги:

- система повинна бути повністю прозора для будь-якого користувача;
- будь які дії в системі повинні бути збереженні;
- система повинна бути не дорогою в використанні;
- неможливість змінити історію системи;
- система повинна бути безпечною, надійною та зрозумілою у використанні;

Тестування методом "чорної скриньки" - це метод тестування програмного забезпечення, при якому перевіряється робота програми без знання її внутрішньої побудови та схеми роботи. Іншими словами, не маючи доступу до коду програми. Тому на рисунку 1.4 представимо систему у вигляді «чорної скриньки».

Система буде мати наступні вхідні дані:

– дані про NFT;

– дані користувачів;

– дані благодійного фонду;

На виході система повинна проводити аукціони для збору коштів благодійним організаціям.

Рисунок 1.4 - Система у вигляді «чорної скриньки»

Бізнес-аналітики можуть отримати наскрізне уявлення про життєвий цикл бізнес-процесів за допомогою моделювання бізнес-процесів, техніки управління бізнес-процесами (BPM), яка створює візуалізацію робочих процесів на основі даних. Ці моделі процесів допомагають організаціям документувати робочі процеси, відображати ключові показники, точно визначити потенційні проблеми та розумно автоматизувати процеси.

Модель бізнес-процесу – це графічне представлення бізнес-процесу або робочого процесу та пов’язаних з ним підпроцесів. Моделювання процесів генерує вичерпні кількісні діаграми діяльності та блок-схеми, що містять важливе уявлення про функціонування даного процесу, включаючи наступне:

- Події та дії, які відбуваються в рамках робочого процесу;
- Хто є власником або ініціатором цих подій і заходів;
- Точки прийняття рішень і різні шляхи, які робочі процеси можуть приймати на основі їхніх результатів;
- Пристрої, що беруть участь у процесі;
- Терміни загального процесу та кожного етапу процесу;
- Показники успіхів і невдач процесу.

Тому створимо загальну діаграму бізнес процесів торгів благодійного аукціону, яку зображено на рисунку 1.5.

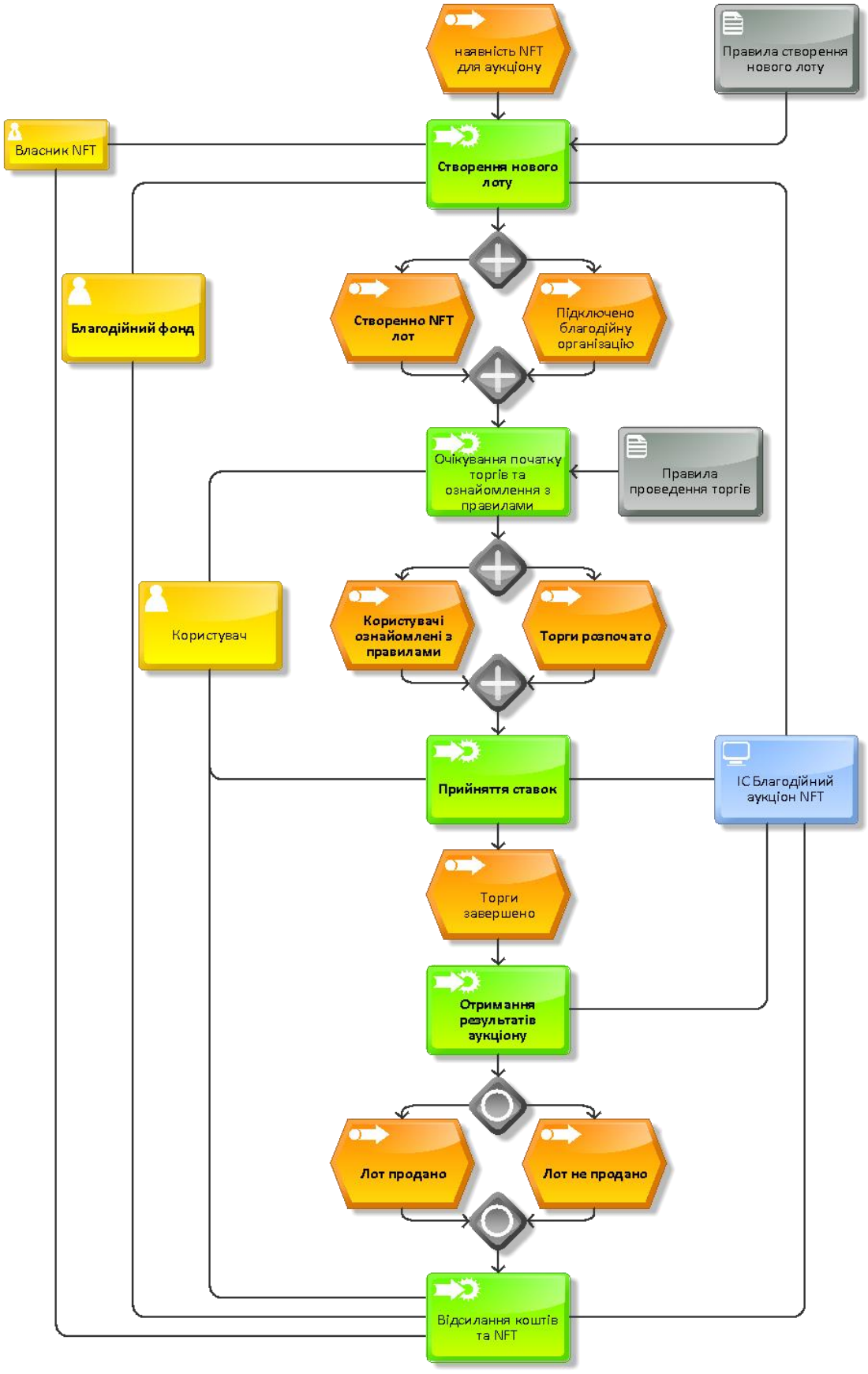


Рисунок 1.5 - Узагальнена бізнес діаграма системи для благодійності

Висновки до розділу 1

Отже, у результаті виконання першого розділу було проведено аналітичний огляд дипломної роботи. Проаналізовано існуючі підходи, результати та рішення, визначено актуальність проблеми, виявлено проблемні моменти і визначено мету кваліфікаційної роботи.

Було сформульовано об’єкт, предмет та мету дослідження кваліфікаційної роботи. Визначено функціональні та нефункціональні вимоги. Представлено систему у вигляді «чорної скрині».

Було проаналізовано існуючі рішення щодо благодійності в криптовалюті та проведення NFT аукціонів. В результаті було з’ясовано, що необхідно створити повністю прозору систему, яка дає повну гнучкість в виборі фонду для пожертви та свободи власника лоту створювати свої умови для проведення аукціону.

РОЗДІЛ 2. ТЕХНІЧНА ВІДОМОСТА ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ НА ОСНОВІ БЛОКЧЕЙНУ

У цьому розділі ми дамо короткий вступ та пояснення деяких основних концепцій криптографії, технології блокчейн і пов’язаних концепцій, таких як смарт-контракти.

2.1 Криптографія

Криптографія надає методи перетворення даних, щоб зробити їх непотрібними для ненавмисних одержувачів даних. Безкорисність у цьому контексті означає зрив двох основних дій: вилучення інформації з даних і введення неправдивих даних або зміна даних. Це називається відповідно проблемою конфіденційності та цілісності. Крім того, можна уявити випадок, коли відправник шифрує і надсилає повідомлення лише для того, щоб пізніше заперечити його відправлення. Неможливість правдоподібно заперечити надсилання конкретних даних є ще однією криптографічною метою, яка називається невідмовністю.

За своєю суттю криптографія є теорією, але також значною мірою практикою запобігання та виявлення шахрайства або забороненого доступу до даних і їх використання. Шифрування даних можна розділити на три гілки: без ключів, симетричним та асиметричним ключем. Примітиви без ключів – це функції, які не використовують ключ для шифрування повідомлення, наприклад. хешування та перестановки довільної довжини. Примітиви з симетричним ключем використовують один і той же ключ для шифрування та дешифрування, тоді як криптографія з асиметричним ключем використовує систему відкритого та приватного ключа (не рівні один одному), які необхідні як для шифрування, так і для дешифрування.

2.2 Перша критовалюта - Bitcoin

В останні роки криптовалюта біткойн привернула велику увагу громадськості. На відміну від попередніх спроб використання цифрових валют, які покладалися на TTP для підтримки стану мережі, біткойн використовує повністю децентралізований протокол для підтримки згоди між усіма вузлами. Цей протокол забезпечує безпечні, надійні та в високій мірі анонімні транзакції.

У мережі Bitcoin кожен учасник мережі має унікальну адресу і може надсилати валюту в транзакціях іншим учасникам. Під час кожної транзакції певна кількість біткоїнів буде переведена з адреси відправника на адресу одержувача. Транзакції, що відбулися за певний проміжок часу, групуються в окремі блоки, які утворюють розподілену книгу транзакцій, яка називається blockchain. Цю розподілену структуру даних підтримують і розширюють учасники, які називаються майнерами, які постійно намагаються вирішити криптографічну головоломку під назвою «Доказ роботи» (PoW), щоб перевірити нові блоки. Коли трансакції транслюються учасниками мережі, майнери включають їх у створені ними блоки. Майнери, які успішно підтвердили нові блоки, винагороджуються новоствореними біткойнами та комісіями за транзакції.

2.2.1 Адреси

Кожен учасник мережі біткойн повинен створити файл гаманця, в якому зберігаються цифрові ключі, які використовуються для підписання транзакцій іншим учасникам, і біткойн-адреса, яка використовується для отримання транзакцій від інших учасників. Цей файл гаманця зберігається не в блокчейні, а на локальному жорсткому диску користувача.

Цифрові ключі, які використовуються для підписання транзакцій, складаються з публічної та приватної частин:

Закритий ключ - це просто випадково згенероване 256-бітове число:

K priv ∈{0 , 1} 256

Потім 512-бітовий відкритий ключ генерується з приватного ключа за допомогою необоротного відображення, відомого як алгоритм Еліптичної кривої

DSA (ECDSA): K паб = ECDSA 512 (K priv)

Коли власник адреси хоче опублікувати транзакцію, вона створює підпис за допомогою свого приватного ключа і в той же час розкриває свій відкритий ключ. Потім майнери можуть перевірити справжність транзакції за допомогою відкритого ключа облікового запису.

Відкритий ключ облікового запису розкривається лише тоді, коли він підписує транзакцію. Для отримання біткойнів в якості адреси одержувача використовується хеш-функція відкритого ключа. Цей 160-бітовий хеш генерується за допомогою хеш-функцій RIPMED 160 і SHA 256 :

Адреса K = RIPEMD 160 (SHA 256 (K pub)))

2.2.2 Транзакції

Транзакції — це структури даних, які використовуються для передачі вартості між учасниками. Кожна транзакція містить певну кількість входів і виходів. Входи посилаються на вихідні дані транзакцій у минулому, які називаються «невитраченими результатами транзакцій» або UTXO. UTXO — це шматки біткойнів, які належать певному власнику та записуються в блокчейні. UTXO, які належать певному власнику, розкидані серед сотень транзакцій і блоків. Таким чином, фактичний залишок біткойн-адреси не зберігається в мережі і повинен бути розрахований програмним забезпеченням гаманця.

Оскільки баланс власника складається з кількох UTXO з різними значеннями, часто неможливо надіслати певну кількість біткойнів на адресу одержувача. Замість цього кілька UTXO споживаються як вхідні дані транзакції, а вихід складається з нового UTXO, що представляє фактичну суму, надіслану бажаному одержувачу, та іншого UTXO, що представляє зміну, яка надсилається назад до гаманця відправника. Це показано на рисунку 2.1.

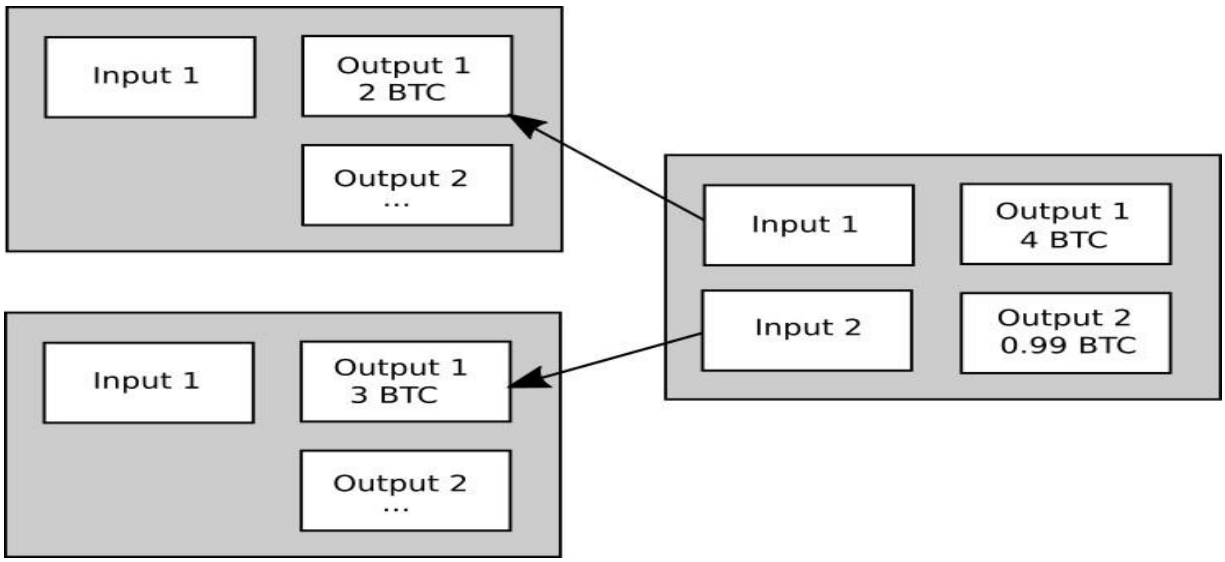


Рисунок 2.1 Вихідні дані транзакцій A і B як вхідні дані транзакції C.

У цьому прикладі чотири біткойни надсилаються іншому учаснику. Оскільки відправник не має UTXO з доступним точним значенням, вона посилається на дві попередні виходи транзакції зі значеннями двох і трьох біткойнів як вхідні дані транзакції C. Потім вона надсилає чотири біткойни одержувачу у виході 1 транзакції C. Вихід 2 це зміна, спрямована назад до неї самої. Різниця вхідних і вихідних значень — це комісія за транзакцію, яку збирають майнери, які включають цю транзакцію у свої блоки. Комісія за транзакції стимулює майнерів надавати пріоритет транзакціям над іншими або включати їх взагалі. Комісія за транзакції також запобігає зловживання транзакціям, які дестабілізують мережу. UTXO, створений трансакцією, включає сценарій блокування, який «блокує» UTXO, вказуючи умови, необхідні для проведення нової транзакції. Сам сценарій реалізовано на простій повній мові сценаріїв, заснованій на стеку, без огляду, під назвою «Script». Коли транзакція створена, відправник може вирішити, чи хоче він використовувати сценарій pay-to-pub-key-hash (p2pkh) або pay-to-script-hash (p2sh). У першому випадку хеш відкритого ключа одержувача UTXO включається в це поле, а власник асоційованого приватного ключа потім вважається власником UTXO. Це, безумовно, найпоширеніший випадок використання. Використовуючи скрипт p2sh, можна вказати складніші умови шляхом комбінування операцій мови

сценаріїв. Поширеним випадком використання p2sh є схема мультипідпису, в якій кілька власників ключів повинні підписати UTXO, щоб витратити його на транзакцію. Вхідні дані транзакції є покажчиками на UTXO, які посилаються на хеш транзакції, яка їх створила, та на індекс UTXO у цій транзакції. Вхідні дані транзакції також включають сценарій розблокування, який задовольняє умовам сценарію блокування. У більшості випадків це підпис, який підтверджує право власності на адресу в сценарії блокування.

2.2.3 Блокчейн

Серцем децентралізованої криптовалютної мережі є блокчейн, структура даних лише для додавання, яка зберігає кожну транзакцію, коли-небудь виконану в мережі. Кожен блок в блокчейні містить хеш свого попередника. Це створює ланцюжок блоків, що тягнеться назад до першого блоку, блоку генезису. Оскільки наступні блоки залежать один від одного, неможливо змінити блок ретроспективно, якщо він був присутній протягом тривалого часу в ланцюжку. Це дуже ускладнює проведення атак подвійних витрат.

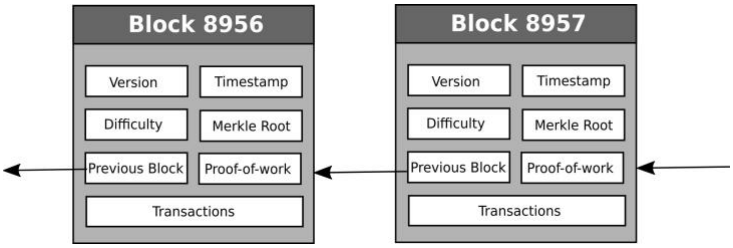


Рисунок 2.2 Структура блоку в блокчейні.

На рисунку 2.2 показана структура окремого блоку в блокчейні. Крім транзакцій, які відбулися протягом приблизно 10 хвилин, він також містить заголовок з метаданими. Заголовок містить підтвердження роботи, яке перевіряє цілісність блоку та кореня дерева Меркла, щоб покращити масштабованість системи. Далі містить мітку часу його створення, версію протоколу біткойн та хеш попереднього блоку.

2.2.4 Дерево Меркла

Дерево Меркла — це структура даних, яка зберігає цифровий підпис для всього списку транзакцій у блоці. Він використовується для ефективної перевірки цілісності транзакції за допомогою двійкового хеш-дерева.

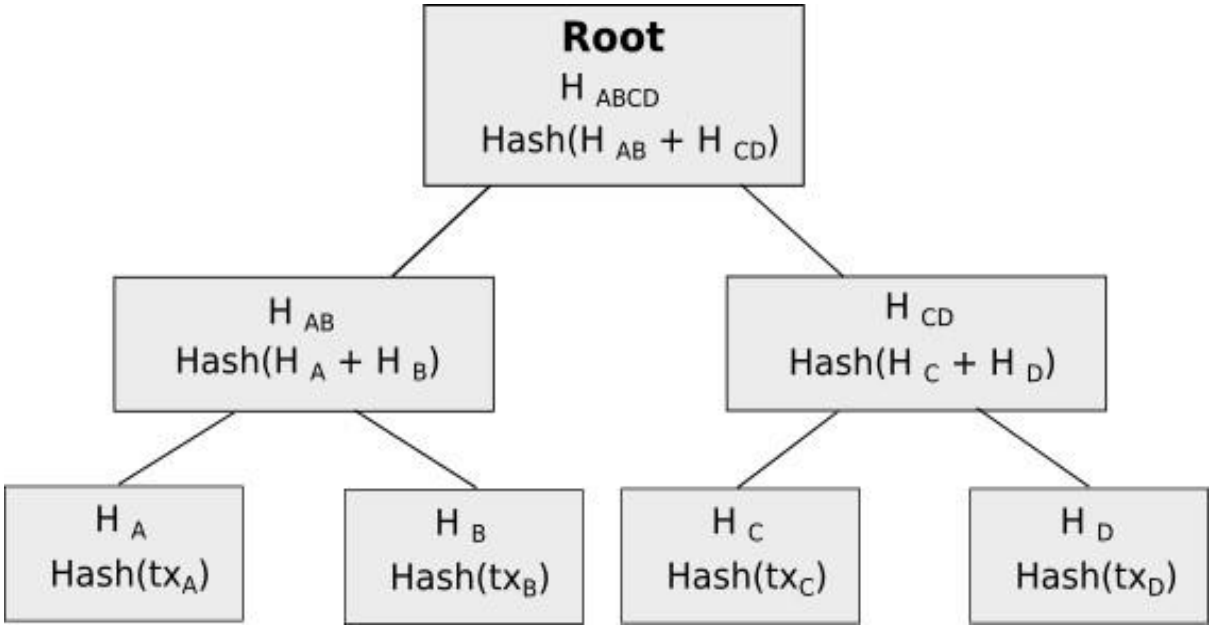


Рисунок 2.3 Структура дерева Меркла.

Рисунок 2.3 ілюструє структуру дерева Меркла. Його кінцеві вузли містять хеш-значення окремих транзакцій блоку. Кожне хеш-значення нелістових вузлів є хешем двох його дочірніх. Отриманий кореневий вузол цього дерева потім включається в блок, який містить пов’язані транзакції.

Дерево Меркла дозволяє вузлу завантажувати лише заголовок блоку та невелику кількість вузлів із дерева Меркла для перевірки транзакції. У прикладі, показаному на рисунку 3.4, лише вузли H(D) , H(AB) і H(EFGH) потрібні для підтвердження включення транзакції H(C) в блок. Ця перевірка займає в середньому O(log n) і не більше O(n), оскільки структура така ж, як і в двійковому дереві пошуку. Якщо зловмисник незаконно вносить недійсну транзакцію десь у нижній частині дерева, хеш цієї транзакції поширюється вгору до кореневого вузла і робить хеш всього блоку недійсним.

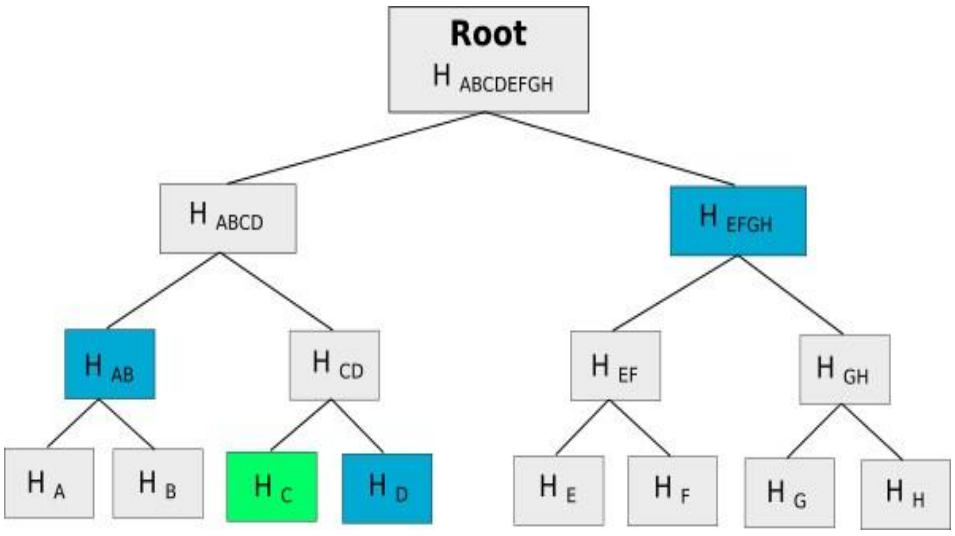


Рисунок 2.4 Перевірка транзакції в дереві Меркла.

Можливість завантажувати та перевіряти лише частину блоку для перевірки окремих транзакцій важлива для стійкості мережі. Повний вузол, що зберігає та обробляє всі транзакції кожного блоку, споживає 15 ГБ дискового простору з квітня 2014 року та зростає на 1 Гб на місяць. Мобільний пристрій не може зберігати такі обсяги даних.

Протокол, відомий як спрощена перевірка платежу (SPV), дозволяє використовувати так звані «світлі вузли» або вузли SPV. Вузли SPV завантажують лише заголовок блоків і частину дерева Меркла, що має значення для підтвердження включення транзакції. Подібним чином вузли SPV підтверджують включення блоку в блокчейн шляхом перевірки заголовка блоків. Поєднання цих процесів перевірки дозволяє вузлу SPV довести, що транзакція належить до блокчейну, завантаживши менше 1 кілобайта даних.

2.2.5 Майнінг

Кожен вузол майнінгу в мережі приймає всі транзакції, які відбулися з моменту створення останнього блоку. Потім вузол майнінгу включає транзакції в новий блок і намагається підтвердити його, неодноразово генеруючи випадкове ціле число (nonce), доки хеш цього цілого числа разом із заголовком блоку не дасть значення, менше за попередньо визначене цільове значення. Цільове

значення коригується динамічно і залежить від загальної потужності хешування мережі. Він налаштований таким чином, що блок створюється в середньому кожні 10 хвилин. Процес майнінгу ускладнює для окремого вузла маніпулювання станом мережі, якщо він не контролює більшу частину потужності майнінгу.

Перший вузол, якому вдається обчислити nonce, отримує винагороду за транзакцію та винагороду coinbase. У вузлі генезису ця винагорода coinbase становила 50 біткойнів, але кожні 210 000 блоків винагорода зменшується наполовину. Винагорода coinbase є єдиним джерелом нової валюти, тому пропозиція біткойнів обмежена 21 мільйоном.

Коли вузол майнінгу знаходить одноразовий номер, він додає його до заголовка блоку і передає його всім вузлам мережі. Оскільки процес трансляції займає деякий час, можливо, два або більше майнерів знайдуть рішення для одного блоку і поширять його в мережі. Це створює розгалуження в блокчейні, і протягом короткого часу кілька ланцюжків можуть співіснувати. Коли це відбувається, нові блоки додаються до найдовшого ланцюга чесними майнерами.

2.2.7 Обмеження Bitcoin

На додаток до простих грошових транзакцій, криптовалютні мережі теоретично можуть використовуватися для зберігання довільних даних і виконання довільних переходів станів і, таким чином, увімкнення більш складних і потужних додатків, таких як смарт-контракти або розподілені анонімні організації (DAO).

Було кілька спроб модернізувати просту мову сценаріїв біткойн для реалізації різних розумних контрактів. Однак, оскільки ця система сценаріїв має низьку виразність і не є повною, багато завдань є дуже дорогими для обчислення або навіть неможливими. Тому краще використовувати мову сценаріїв розумних контрактів загального призначення. Віртуальна машина Ethereum (EVM), яка використовується в цій програмі, є першою системою, яка має такі можливості.

2.3 Платформа Ethereum

Ethereum — це розподілена обчислювальна платформа, яка використовує блокчейн для зберігання не тільки стану облікових записів користувачів, а й коду програми та пов'язаного з ним стану. Він дозволяє розподілене виконання довільного коду та забезпечує відповідну платформу для розробки смарт-контрактів, які розглядаються далі. Ethereum був спеціально розроблений, щоб дозволити будь-кому писати смарт-контракти та децентралізовані програми. Він підтримує кілька мов сценаріїв, які можна скомпілювати в байтовий код, який виконується на віртуальній машині Ethereum (EVM).

Станом на травень 2017 року Ethereum є другою за величиною криптовалютою з ринковою капіталізацією понад 8 мільярдів доларів США.

У наступних підрозділах розглядаються найважливіші аспекти Ethereum.

2.3.1 Аккаунти

Аккаунт – це об'єкт, який зберігає стан балансу облікового запису користувача або контракту. Перший обліковий запис називається зовнішнім обліковим записом (EOA), оскільки він належить зовнішньому об'єкту і контролюється закритим ключем. Він може надсилати повідомлення, створюючи та підписуючи транзакції своїм закритим ключем.

Другий рахунок називається контрактним рахунком, і його стан контролюється кодом контракту. Коли обліковий запис контракту отримує повідомлення, його код виконується. Він також може надсилати повідомлення іншим контрактам або створювати нові контракти.

Віртуальна машина Ethereum не робить різниці між двома типами облікових записів. Кожен обліковий запис має сховище ключ-значення, яке називається сховищем, і баланс у wei, який можна змінити шляхом відправки транзакцій, які включають ефір.

2.3.2 Транзакції та повідомлення

Транзакція — це пакет даних, підписаний зовнішнім обліковим записом. Він містить відправника транзакції, одержувача транзакції, кількість надісланого ефіру, необов'язкове поле даних, ліміт газу та поле ціни на газ. Повідомлення подібне до транзакції, яка надсилається від одного контракту до іншого. Він веде себе подібно до звичайного виклику функції в інших мовах програмування. Кожного разу, коли поточний код облікового запису контракту виконує код операції «виклику», генерується повідомлення, яке надсилається контракту одержувача або зовнішньому обліковому запису. І транзакції, і виклики можуть використовуватися для створення нових контрактів, для виклику функцій контракту або для передачі ефіру в контракт або на зовнішній обліковий запис.

Ether — це токен валюти, що використовується Ethereum для оплати комісій за транзакції. Розробники повинні надавати ефір, коли вони розгортають код контракту в блокчейні, а користувачі повинні витрачати або спалювати ефір, коли вони викликають транзакції за контрактом. Ефір можна обміняти на інші фіатні валюти через різні біржі криптовалют. Ефір можна розділити далі на більш дрібні одиниці, найменша з них називається вей. Один ефір дорівнює 10 18 wei.

Газ використовується для оплати транзакцій в Ethereum. Газ - це не валюта, а внутрішня одиниця ціноутворення, яка відокремлює ринкову ціну ефіру від вартості транзакцій. 1 одиниця газу являє собою ціну за найпростішу операцію, що виконується на EVM. Ціна за одиницю газу, ціна газу, може динамічно коригуватися, щоб адаптуватися до коливань значень ефірної валюти. Ціну на газ може визначити ініціатор транзакції, і майнери вирішують, чи хочуть вони включати транзакції в свої блоки чи ні. Чим більше рядків коду виконує контракт і чим більше пам'яті та сховища він використовує, тим вищим має бути ліміт газу для початкової транзакції. Комісія за транзакції та ліміт газу допомагають запобігти виконанню несправного коду, наприклад нескінченних циклів, і допомагають заощадити обчислювальні ресурси в мережі. Обмеження газу також не стимулює потенційні атаки відмови в обслуговуванні в мережі.

2.3.3 Блокчейн і майнінг

Блокчейн Ethereum дуже схожий на біткойн. Найважливіша відмінність полягає в тому, що блок містить не тільки список транзакцій, але і весь стан мережі. Стан зберігається в структурі даних під назвою «Дерево Патріції».

Дерево Патріції — це модифіковане дерево Меркла, яке оптимізовано для вставки та видалення вузлів. Він зберігає стан усіх контрактних і зовнішніх облікових записів. Кожен блок зберігає посилання на корінь дерева й оновлює лише ті частини, які змінилися внаслідок впливу транзакцій у цьому блоці. Це дозволяє новим вузлам завантажувати лише дерево Patricia замість всіх блоків, щоб отримати стан усіх облікових записів, і, отже, економить значний обсяг дискового простору. Підраховано, що якщо цю концепцію застосувати до біткойн, то вузлу буде потрібно зберігати від 5 до 20 разів менше даних. Ethereum також використовує інший алгоритм Proof-of-work, який називається Ethash, який виробляє блок кожні 12 секунд у середньому порівняно з 10 хвилинами в Bitcoin. Це має перевагу в тому, що транзакції можуть оброблятися швидше, а одержувачу транзакції не доведеться довго чекати, поки вона не зможе вважати транзакцію безпечною. Це також підвищує інтерактивність додатків, які взаємодіють з контрактами на блокчейні.

Негативний ефект швидкого блокування полягає в тому, що збільшується швидкість старіння, швидкість, з якою виробляються блоки, які не є частиною основного ланцюга. Це ризик для безпеки, який може призвести до централізації пулів для майнінгу, оскільки багато майнерів не отримають винагороду за свої зусилля по видобутку нових блоків. Хоча в Bitcoin такі блоки вважаються «сиротськими» і більше не використовуються, протокол GHOST Ethereum також дозволяє включати такі «дядькові» блоки та винагороджує майнерів за них.

На відміну від біткойна, винагорода за майнінг блоку є статичною і становить рівно 5,0 ефіру. Успішні майнери також збирають весь газ, який використовується в транзакціях блоку. Майнери блоків «дядько» отримують 7/8 статичної винагороди за блок.

2.4 Смарт-контракт

Смарт-контракт — це спеціальний обліковий запис, який зберігає виконуваний код разом із пов'язаними з ним даними та балансом рахунку в блокчейні. Смарт-контракти мають адресу (відкритий ключ) і створюються за допомогою транзакцій. Трансакції також використовуються для взаємодії з контрактом на блокчейні, надсилаючи гроші на баланс рахунку або виконуючи код. Для виконання коду контракту виклик функції, що містить ім'я функції та її параметри, кодується у двійковому коді та надсилається до контракту в полі даних транзакції.

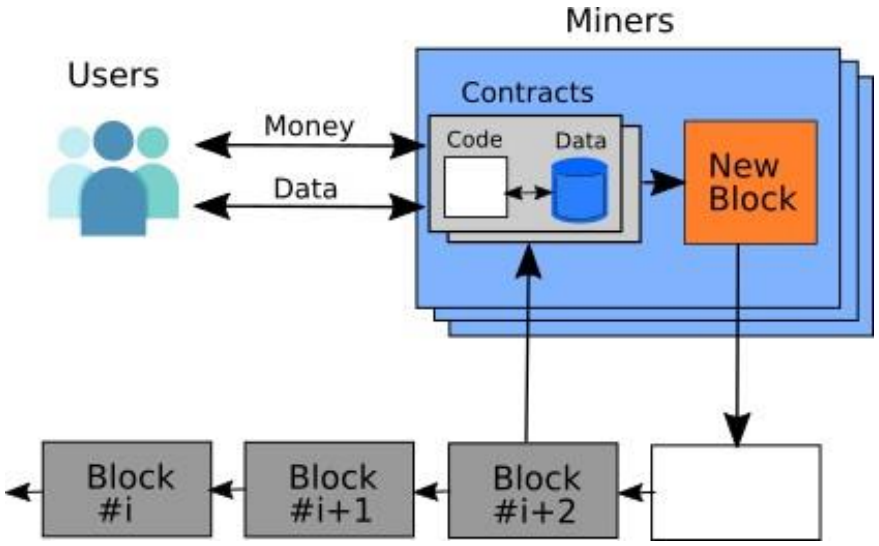


Рисунок 2.5 Виконання смарт-контракту на блокчейні.

Рисунок 2.5 ілюструє взаємодію зовнішніх облікових записів (користувачів) із контрактом. Кожен раз, коли контракт отримує повідомлення від іншого контракту або транзакцію від користувача, він може отримати ефір або виконати функцію, яка вказана в полі даних. Таким же чином контракт може надсилати гроші зі свого балансу на інші рахунки або виконувати функції за іншими контрактами через трансляцію повідомлень.

Виконання коду відбувається на всіх вузлах майнінгу в мережі одночасно, які досягають консенсусу щодо нового стану контракту за допомогою алгоритму підтвердження роботи. Постійні змінні контракту зберігаються в сховищі,

сховищі ключ-значення, пов’язаному з контрактом, який зберігається в блокчейні. Доступ до сховища дуже дорогий (20000 одиниць газу на 256-бітове слово), оскільки його потрібно зберігати на кожному повному вузлі мережі. Проміжні результати обчислень зберігаються в пам'яті, непостійному масиві байтів. Стан, як і кодекс договору, є публічними, і код договору не може змінюватися ретроспективно.

2.5 Мова програмування Solidity

Платформа Ethereum підтримує різні мови програмування для написання коду смарт-контракту. Найпопулярнішим серед них є Solidity, контрактно-орієнтована мова високого рівня із системою статичних типів, яка має синтаксис, дуже схожий на javascript.

Основною конструкцією Solidity є контракт. Подібно до класу, контракт може містити поля, функції, модифікатори функцій, типи структур і типи перерахувань, і він також успадковується.

У наступних підрозділах коротко описані найважливіші поняття контракту в Solidity, а також обговорити деякі з підводних каменів і проблем безпеки, які можуть виникнути під час написання смарт-контрактів у Solidity.

2.5.1 Змінні та типи даних

Змінні стану містять значення, які постійно зберігаються в сховищі контракту. Оскільки Solidity є статично типізованою мовою, потрібно вказати тип даних змінної. Solidity надає такі елементарні типи, які можна комбінувати, щоб утворити складніші типи:

- Логічні значення: зберігає логічне значення (true, false)
- Цілі числа: включає різні підтипи цілих чисел зі знаком (int) і без знака (uint) з різною довжиною байтів.
- Адреса: представляє 20-байтну адресу Ethereum.

- Байтові масиви фіксованого розміру: включає масиви байтів довжиною від 1 до 32.
- масиви динамічного розміру: це включає тип «байти», масив байтів динамічного розміру та тип рядка, рядок із кодуванням UTF8 динамічного розміру.

Solidity надає деякі спеціальні функції та змінні, які завжди існують у глобальному просторі імен. Вони використовуються для надання інформації про стан блокчейну або контракту. Найважливішою змінною, що використовується в контракті, є об'єкт «msg», який надає інформацію про повідомлення, надіслане контракту під час виклику функції, наприклад, відправник повідомлення, залишок газу для виклику повідомлення або значення, надіслане разом із повідомленням.

2.5.2 Виклики функцій

Функції є виконуваними одиницями контракту і можуть викликатися як внутрішньо, так і зовнішньо. Внутрішній виклик функції — це виклик функції поточного контракту, що виконується. Виклик зовнішньої функції — це виклик функції в екземплярі зовнішнього контракту.

Виклик зовнішньої функції використовує директиву виклику повідомлення EVM, яка очікує назву функції та її аргументи в хешованому форматі. Коли контракт хоче викликати функцію іншого контракту, він може вказати кількість ефіру, який він хоче надіслати, і кількість газу, яку він хоче витратити у функціях значення та газу об’єкта функції.

Поведінка виклику в разі невідповідності типу може відрізнятися:

- якщо адреса абонента не є адресою контракту, створюється виняток;
- якщо адреса належить контракту, який має функцію з таким же підписом, то функція виконується;
- якщо адреса належить контракту, який не має відповідної функції, виконується його резервна функція.

Взаємодія з іншими контрактами може бути небезпечною, оскільки їх реалізації не визначаються їхнім інтерфейсом. Викликаний контракт може виконувати довільні інструкції і, отже, може спричинити серйозні ризики безпеки.

2.5.3 Резервна функція

Коли контракт Solidity компілюється в байт-код EVM, його функції відправляються на початку скомпільованого контракту та ідентифікуються підписом, отриманим з назви функції та її аргументів. Коли функція викликається, цей підпис надається у виклику повідомлення. Якщо не вдається знайти відповідну функцію або підпис не надано взагалі, замість цього викликається резервна функція контракту.

Резервна функція не має аргументів і не може нічого повернути. Якщо контракт отримує порожній підпис разом з ефіром, але не визначає резервну функцію, він генерує виняток (починаючи з Solidity v0.4.0). Резервна функція не повинна містити інструкції, які використовують багато газу, оскільки це також може викликати виняток, коли функція відправлення використовується для передачі ефіру. У цьому випадку доступно лише 2300 газу, і контракт не зможе отримати ефір, коли резервна функція використовує більше газу. Той факт, що резервна функція може бути виконана несподівано, виявляє вразливість безпеки, яка може бути використана зловмисниками для виконання нерекурсивної функції в циклі, це називається «повторним входом».

Щоб запобігти вразливостям безпеки, які виникають внаслідок повторного введення коду, контракт завжди повинен спочатку перевіряти умови, необхідні для виконання дії. Потім слід призначити локальні змінні стану, які змінилися внаслідок дії, перш ніж здійснювати будь-які виклики, які передають ефір. Ця парадигма також називається «Перевірки-Ефекти-Взаємодії».

2.5.4 Обробка винятків

Якщо обліковий запис не містить коду, або функція створює виняток, або у викликаному контракті закінчується запас, виклик функції спричинить виняток. Виняток зупинить виконання поточної функції та поверне всі зміни до стану та балансу контракту, викликані ним. Виняток потім рекурсивно «вибухне» в ланцюжку викликів.

Наразі перехоплення винятків ще неможливе, і єдиний побічний ефект, який помічає абонент, — це те, що в транзакції закінчується газ. Інша проблема полягає в тому, що при використанні функції відправки контракту або однієї з низькорівневих функцій call або delegatecall стан контракту не повертається, а натомість ці функції повертатимуть логічне значення «false». Ці порушення в тому, як винятки обробляються в Solidity, можуть вплинути на безпеку контрактів. Наприклад, коли контракт не перевіряє значення, що повертає функцію відправки , він може помилково вважати, що транзакція була успішною, оскільки не було викликано жодного винятку.

2.5.5 Видимість функцій і змінних стану

Існує чотири різних типи видимості для функцій і змінних у Solidity:

- public: публічні функції можуть бути викликані як внутрішньо цим класом, так і похідними класами, а також зовнішньо за допомогою виклику повідомлення. Для загальнодоступних змінних створюється функція автоматичного отримання.
- приватні: приватні функції та змінні стану видимі лише всередині контракту, що оголошує, а не похідних контрактів.
- зовнішні: зовнішні функції поводяться як публічні функції з тією різницею, що вони не можуть бути викликані всередині контракту, викликаючи «f()», а викликаючи «this.f()».
- внутрішні: внутрішні функції та змінні поводяться як приватні з тією різницею, що їх не можна викликати за допомогою параметра «this» всередині контракту.

Важливо зазначити, що навіть якщо змінна стану або функція позначена як «приватна», це не означає, що її неможливо побачити ззовні блокчейну. Це означає лише те, що інший контракт не може отримати доступ до змінної чи функції. Оскільки всі транзакції в блокчейні є видимими для всіх, стан і функціональність кожного контракту також видимі для всіх.

2.5.6 Модифікатори функцій

Модифікатори функцій можна використовувати для зміни поведінки функції. Вони перевіряють, що функція може виконуватися лише в певному контексті. Наприклад, функція може мати модифікатор, який перевіряє, що абонент повинен мати конкретну адресу. Спеціальний модифікатор — це модифікатор «оплачуваний», який вказує, що функція може отримувати ефір. Якщо ефір надсилається функції, яка не має цього модифікатора, функція видає виняток.

2.5.7 Події

Іншим поняттям Solidity є події. Події – це інтерфейси, які дозволяють використовувати засоби реєстрації EVM. Ці інтерфейси використовуються програмами для підписки на події контракту та для моніторингу стану контракту без безпосередньої взаємодії з ним.

Коли подія викликається, її аргументи зберігаються в структурі даних журналу транзакцій блокчейну, яка недоступна безпосередньо з контракту. Хеш підпису події використовується для ідентифікації і називається темою. Щоб дозволити пошук конкретних значень аргументів теми, до трьох параметрів події можуть отримати атрибут «індексований».

2.6 Застереження щодо безпеки

У цьому розділі розглядаються загальні вразливості безпеки Ethereum, які можуть виникнути під час роботи зі смарт-контрактами, і можливі способи, якими програміст може запобігти їх або обмежити їх вплив.

2.6.1 Обмеження стеку викликів

Щоразу, коли контракт викликає функцію іншого контракту або сам по собі, пов’язаний стек викликів транзакції зростає. Максимальна глибина, яку може досягти стек викликів, становить 1024 кадри. До 18 жовтня 2016 року цей факт можна було використати, створивши майже повний стек викликів, а потім викликавши функцію жертви. Тоді це призведе до несподіваної поведінки, яка була використана разом з іншими вразливими місцями. Потім ця проблема була вирішена за допомогою хардфорка блокчейну Ethereum. Після розвилки абонент не може використовувати більше 63/64 свого газу і, отже, не може досягти глибини стека викликів близько 1024.

2.6.2 Зберігання таємниці в контракті

Є багато договорів, які потрібно деякий час зберігати державну таємницю. Наприклад, ігри для кількох гравців часто не можуть розкрити рішення гравця до того, як усі інші гравці приймуть рішення, оскільки це дасть їм перевагу. У таких випадках гравці повинні використовувати криптографічне «зобов’язання», щоб зв’язати свій вибір, не розкриваючи його іншим гравцям.

У найпростішому випадку таким зобов’язанням може бути хеш-функція випадкового числа (nonce) і фактичного вибору гравця. Після того, як усі гравці надсилають свої зобов’язання, вони відкривають його, викликаючи функцію, яка приймає одноразовий текст і вибір у вигляді простого тексту.

2.6.3 Незмінність договорів

Після розгортання контракту його код більше не можна змінювати. Це означає, що якщо контракт виявляє будь-які вразливості безпеки, немає способу безпосередньо виправити його. Спосіб обмеження впливу незмінності полягає в тому, щоб зберегти архітектуру системи якомога більш модульною та вбудувати якусь функцію «обслуговування», яка може використовуватися власником контракту для заміни помилкових реалізацій його залежностей.

2.6.4 Непередбачуваний стан

Оскільки стан контракту можна оновлювати в кожному блоці, а транзакції в блоці можуть бути згруповані майнером довільно, абонент не може точно знати, в якому стані знаходиться контракт під час відправлення транзакції. Фактичний стан контракту також може бути неясним, коли блокчейн розривається. Можливо, контракт оновлюється на гілці блокчейну, яка пізніше виявиться найкоротшою. Зміни, внесені до договору, будуть потім скасовані. Той факт, що стан контракту не завжди відомий абоненту, може бути використаний супротивниками.

2.6.5 Генерування випадковості

Для генерування випадкових чисел контракти часто використовують хеш або мітку часу блоку, який з’явиться в певний момент у майбутньому, оскільки це значення буде однаковим для всіх у мережі. Оскільки майнери можуть контролювати вміст і порядок блоків, зловмисний майнер, який контролює лише меншу частину мережі, може змінити результат випадкового процесу.

Запропонованим рішенням для цієї атаки є включення протоколів зобов’язань із тимчасовим виконанням. Усі сторони надсилають свої зобов’язання разом із депозитом. Згодом учасники розкривають свої зобов’язання та розкривають свої секрети. Потім на основі розкритих секретів обчислюється випадкове число. Якщо учасник не розкриває свою таємницю, вона втрачає депозит.

2.6.6 Часові обмеження

Коли контракт покладається на часові позначки блоку для реалізації обмежень часу, зловмисний майнер, зацікавлений в маніпулюванні поведінкою контракту, може спробувати маніпулювати часовою міткою своїх блоків до певної міри.

Висновки до розділу 2

Отже в цьому розділі ми зробили аналіз блокчейн технологій і побачили, що блокчейн підпорядковується суворим математичним і криптографічним правилам, що робить його неймовірно безпечною системою, стійкою, як до брутфорсу, так і до будь-яких інших атак, саме через її складну структуру та децентралізацію процесів. А саме це нам потрібно для створення безпечної та прозорої благодійної системи.

РОЗДІЛ 3. ПРОЕКТНО-ТЕХНІЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Архітектура благодійної системи

Благодійна система буде включати в себе дві основні функції:

- прийняття участі в NFT аукціонах;
- взаємодія з благодійним фондом.

Функція прийняття участі в NFT аукціонах;

- створення нового лота;
- видалення лота;
- вибір організації для благодійності;
- зробити ставку;
- забрати свої кошти або виграний лот;
- перегляд доступних лотів.

Взаємодія з благодійним фондом буде включати у себе:

- добавлення благодійної організації;
- видалення благодійної організації;
- отримання інформації про організацію;
- вибір організації для пожертви;
- зробити анонімну пожертву;
- зробити пожерту в якості будь-якого токену ERC-20.

Із описаного вище функціонального аналізу побудуємо дерево функцій, яке зображене на рисунок 3.1.

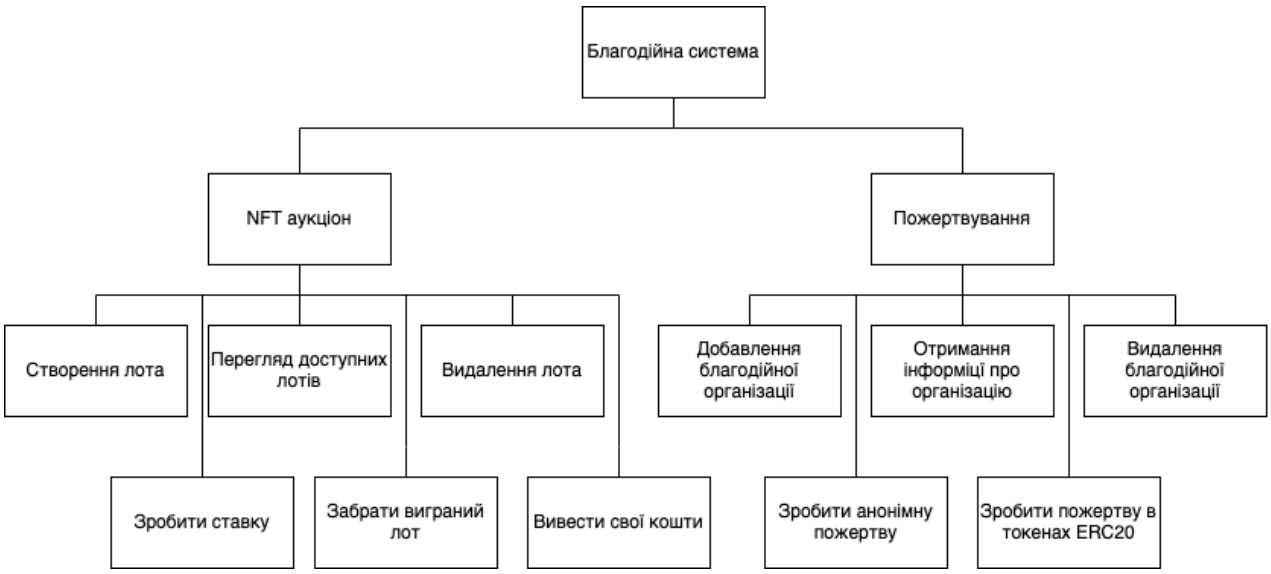


Рисунок 3.1 – Дерево функцій системи

Із вищеописаного дерева функцій можна описати карту процесів. Побудова класифікаційної моделі складається із чітко визначених функцій що включає:

- для NFT аукціону: створення та видалення лота, перегляд доступних лотів, зробити ставку, забрати виграний лот та вивід власних коштів;
- для пожертвування: добавлення та видалення благодійної оргінізації, отримання інформації про благодійну організацію, зробити анонімну пожертву та зробити пожертву в токенах ERC-20.

Опишемо діаграму сутностей, за якими буде існувати предметна область та иділимо такі об’єкти:

- користувач;
- благодійна організація;
- благодійний фонд;
- NFT аукціон;
- NFT актив;

Діаграма сутностей предметної області наведено на рисунок 3.2.

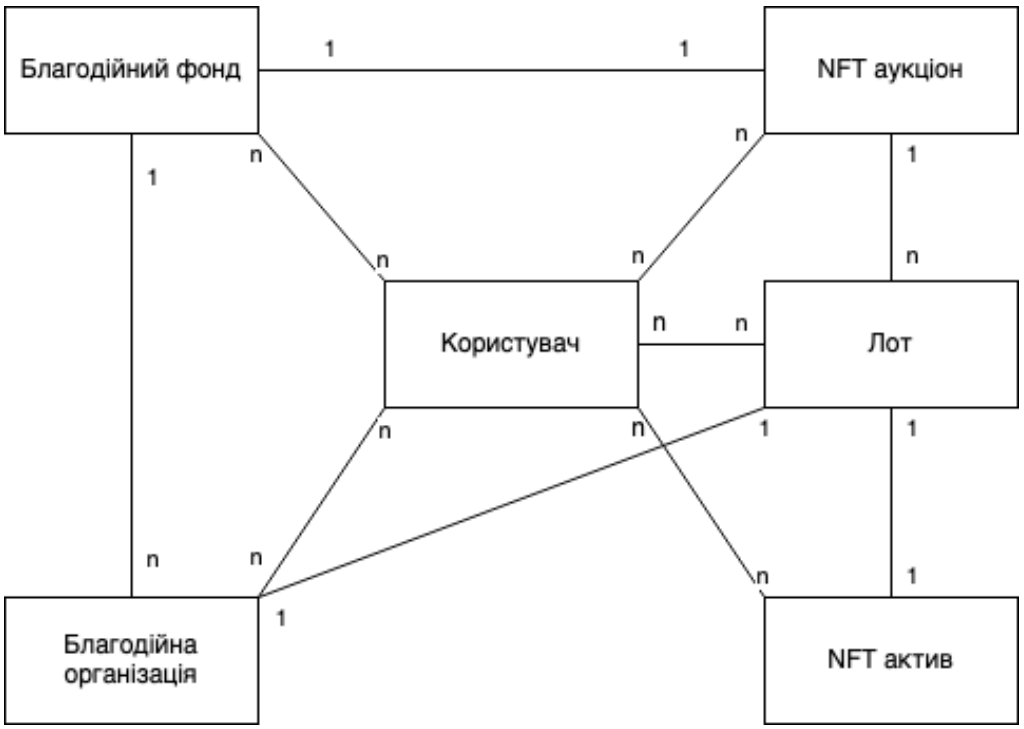


Рисунок 3.2 – Діаграма сутностей

Визначимо зв’язки між сутностями (таблиця 3.1):

Таблиця 3.1 - Зв’язки між сутностями

№	Сутності, які утворюють зв’язок	Тип зв’язку	Пояснення
1	Користувач – Благодійний фонд	Багато-до-багатьох	Багато користувачів може взаємодіяти з багатьма благодійними фондами.
2	Користувач – Благодійна організація	Багато-до-багатьох	Багато користувачів може взаємодіяти з багатьма благодійними організаціями.
3	Користувач – NFT аукціон	Багато-до-багатьох	Багато користувачів може приймати участь в багатьох аукціонах.
4	Користувач – Лот	Багато-до-багатьох	Багато користувачів може зробити ставку на багато лотів.

5	Користувач – NFT	Багато-до-багатьох	Багато користувачів може мати багато NFT активів.
6	Благодійний фонд – Благодійна організація	Один-до-багатьох	Один фонд може мати багато благодійних організацій активів.
7	Благодійний фонд – NFT аукціон	Один-до-одного	NFT аукціон працює лише з одним благодійним фондом.
8	NFT аукціон - Лот	Один-до-багатьох	Один аукціон може мати багато лотів.
9	NFT - Лот	Один-до-одного	Один лот відповідає лише за один NFT актив.

Після того, як було визначено основні сутності опишемо та побудуємо життєвий цикл благодійного NFT аукціону (рисунок 3.3):

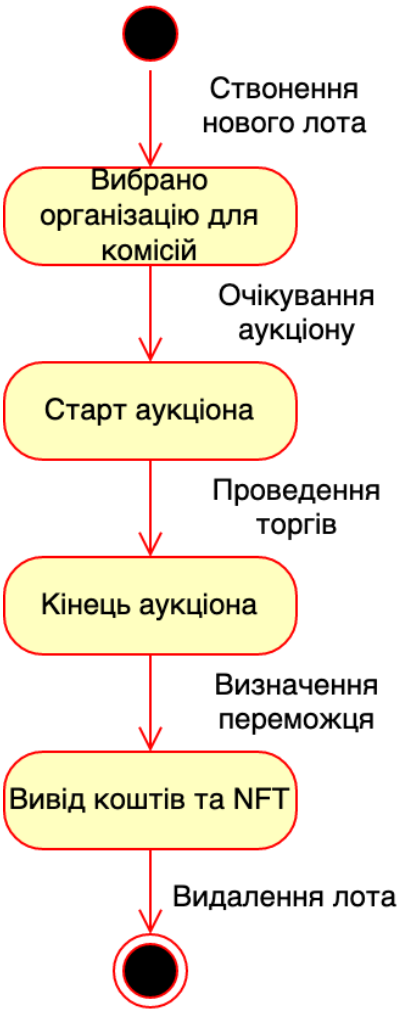


Рисунок 3.3 – Життєвий цикл прогнозу

Життєвий цикл процесу проведення аукціонів буде складатись із наступних станів:

- вибрано благодійну організацію для комісій аукціону;
- старт аукціону;
- кінець аукціону;
- вивід коштів та NFT.

Після створення нового лоту, власник вибирає благодійну організацію для комісій аукціону. У результаті виконання проведення аукціону маємо переможця після чого можна виводити кошти та видаляти лот.

Визначимо акторів, прецедентів та їх відношення системи. У застосунку буде брати участь користувач та сама система.

Для користувача буде доступно наступні дії:

- прийняти участь в аукціоні;
- отримати результати аукціону;
- вивести активи.
- зробити добровільну пожертву;
- отримати інформацію про благодійну організацію,

Для власника благодійного фонду буде доступно наступні дії:

- добавити благодійну організацію;
- видалити благодійну організацію;

Для власника аукціону буде доступно наступні дії:

- створити лот;
- видалити лот;

З неведеного вище опису побудуємо UML діаграму (рисунок 3.4).

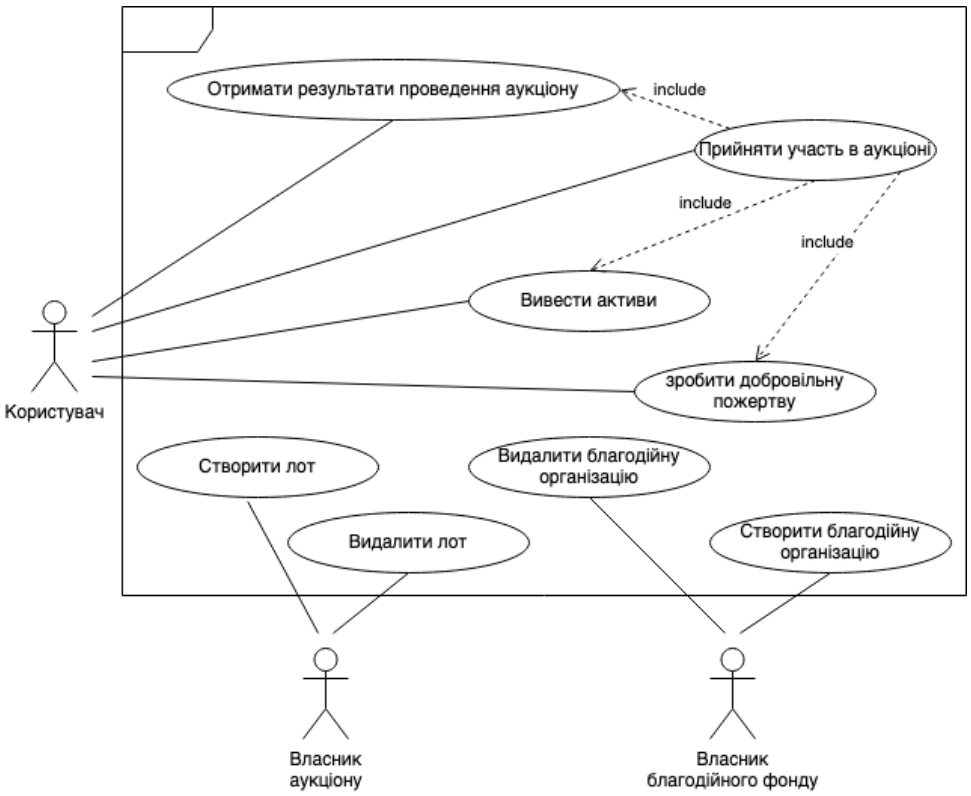


Рисунок 3.4 – UML діаграма прецедентів

Варіанти використання відповідають діям акторів. На діаграмі також показано, що варіанти використання «отримати результати проведення аукціону», «вивести активи» та «зробити добровільну пожертву» включені у прецедент «прийняти участь в аукціоні».

3.2 Обґрунтування вибраних технологій і мови програмування

Для розробки смарт-контрактів благодійної системи було використано декілька технологій:

- Solidity – мова програмування для написання смарт контрактів;

Solidity — це мова програмування на основі фігурних дужок. Натхненна C++, Python і JavaScript, мова розроблена для віртуальної машини Ethereum (EVM). Solidity — це статично типізована мова, яка серед інших функцій підтримує успадкування, бібліотеки та складні визначені користувачем типи. За допомогою Solidity ви можете створювати контракти, наприклад для голосування, краудфандингу чи сліпих аукціонів.

- Remix – IDE для зручної розробки, налаштування та тестування смарт-контрактів;

Remix — це настільний і веб-додаток з відкритим кодом, який є IDE або інтегрованим середовищем розробки, що спеціалізується на розробці Ethereum. Це робить Remix відмінним інструментом для розробки смарт-контрактів на мові програмування Solidity. Середовище розробки має швидкий цикл розробки з широким набором корисних плагінів. Remix IDE не тільки чудово підходить для розробки смарт-контрактів, але також є навчальною платформою.

Оскільки Remix IDE спеціалізується на розробці Ethereum і смарт-контрактів, середовище містить кілька цінних інструментів, які можна використовувати в процесі розробки. Прикладом є вбудований компілятор, який дозволяє компілювати свої контракти безпосередньо в середовищі розробки. Також можна розгорнути контракти через Remix і перевірити їх безпосередньо. Крім того, ви також можете налаштувати свою IDE за допомогою плагінів відповідно до ваших конкретних потреб.

- MetaMask – гаманець Ethereum для зручної взаємодії з blockchain;

MetaMask — це гаманець Ethereum з відкритим вихідним кодом, який підтримує всі типи токенів на основі Ethereum (наприклад, токени, сумісні зs стандартом ERC-20, або невзаємозамінні, також відомі як NFT). Крім того, ви можете отримати їх від інших користувачів мережі або придбати за допомогою вбудованих інтеграцій з Coinbase і ShapeShift. Відмінною особливістю MetaMask є те, що розширення може взаємодіяти з іншими веб-сайтами. При використанні інших гаманців потрібно скопіювати та вставити платіжну адресу або відсканувати QR-код на окремому пристрої. Використовуючи розширення MetaMask, сайт просто надішле запит на гаманець, і вам буде запропоновано прийняти або відхилити транзакцію. MetaMask може бути звичайним криптовалютним гаманцем, але головна його перевага — безперебійна взаємодія зі смарт-контрактами та децентралізованими додатками. Гаманець MetaMask можна встановити в Google Chrome, Firefox і Brave. Він також працює на iOS та Android.

3.3 Опис реалізації смарт-контрактів

Розпочнемо розробку системи, яка буде мати наступну структуру робочого простору.

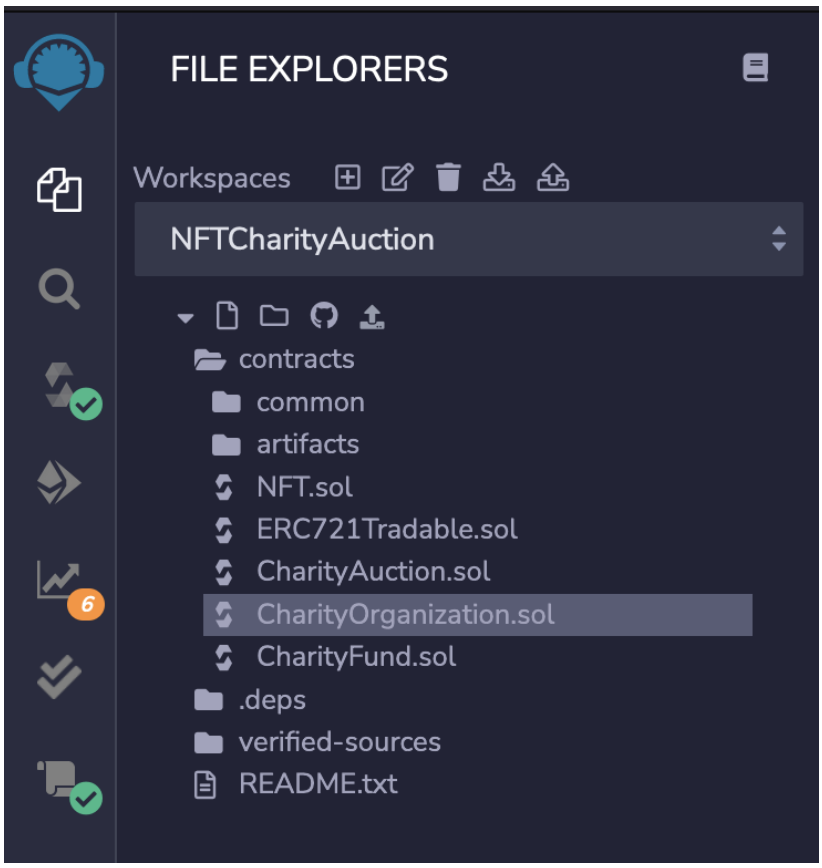


Рисунок 3.5 – Структура проекту

На рис. 3.5 зображено структуру системи, що складається із основних чотирьох смарт-контрактів:

- NFT.sol – файл для створення NFT;
- CharityOrganization.sol – файл для створення смарт-контракту благодійної організації;
- CharityFund.sol – файл для створення смарт-контракту благодійної фонду;
- CharityAuction.sol – файл для створення смарт-контракту аукціону;
- artifacts – папка, для зберігання успішно відкомпільованих смарт-контрактів.
- .deps – папка, для збереження імпортованих контрактів.

Лістинги відповідних смарт контрактів буде наведено у додатку А.

NFT смарт-контракт

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.13;

import "./ERC721Tradable.sol";

contract NFT is ERC721Tradable {
    string private _baseTokenURI;

    string private _contractURI;

    constructor(
        string memory name,
        string memory symbol,
        string memory baseTokenUri,
        string memory contractUri,
        address _proxyRegistryAddress
    ) ERC721Tradable(name, symbol, _proxyRegistryAddress) {
        _baseTokenURI = baseTokenUri;
        _contractURI = contractUri;
    }

    function _baseURI() internal view override returns (string memory) {
        return _baseTokenURI;
    }

    function contractURI() public view returns (string memory) {
        return _contractURI;
    }
}
```

Рисунок 3.6 – Структура смарт контракту NFT.sol

Смарт контракт Nft наслідується від контракту ERC721Tradable, який в свою чергу наслідується від базових контрактів: ERC721URIStorage, ContextMixin, NativeMetaTransaction та Ownable.

NFT та містить такі основні атрибути:

- name – повна назва NFT токена;
- symbol – скорочена назва, яка виступає як ідентифікатор;
- baseTokenUri – посилання на JSON файл, який містить повну інформацію про NFT;
- contractUri – посилання на JSON файл, який містить повну інформацію про цей контракт;

Смарт-контракт благодійної організації

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

contract CharityOrganization is Ownable {
    string public organizationName;
    string public organizationDescription;
    address public organizationManager;
    uint256 public minimumDonation;
    uint256 public numberOfRequests;
    address[] public donators;
    mapping(address => bool) IsAlreadyDonator;
    mapping(address => address) tokenToPriceFeed;

    constructor(
        address _organizationManager,
        string memory _organizationName,
        string memory _organizationDescription,
        uint256 _minimumDonation
    ) {
        organizationManager = _organizationManager;
        organizationName = _organizationName;
        organizationDescription = _organizationDescription;
        minimumDonation = _minimumDonation;
    }
```

Рисунок 3.8 – Структура смарт контракту CharityOrganization.sol

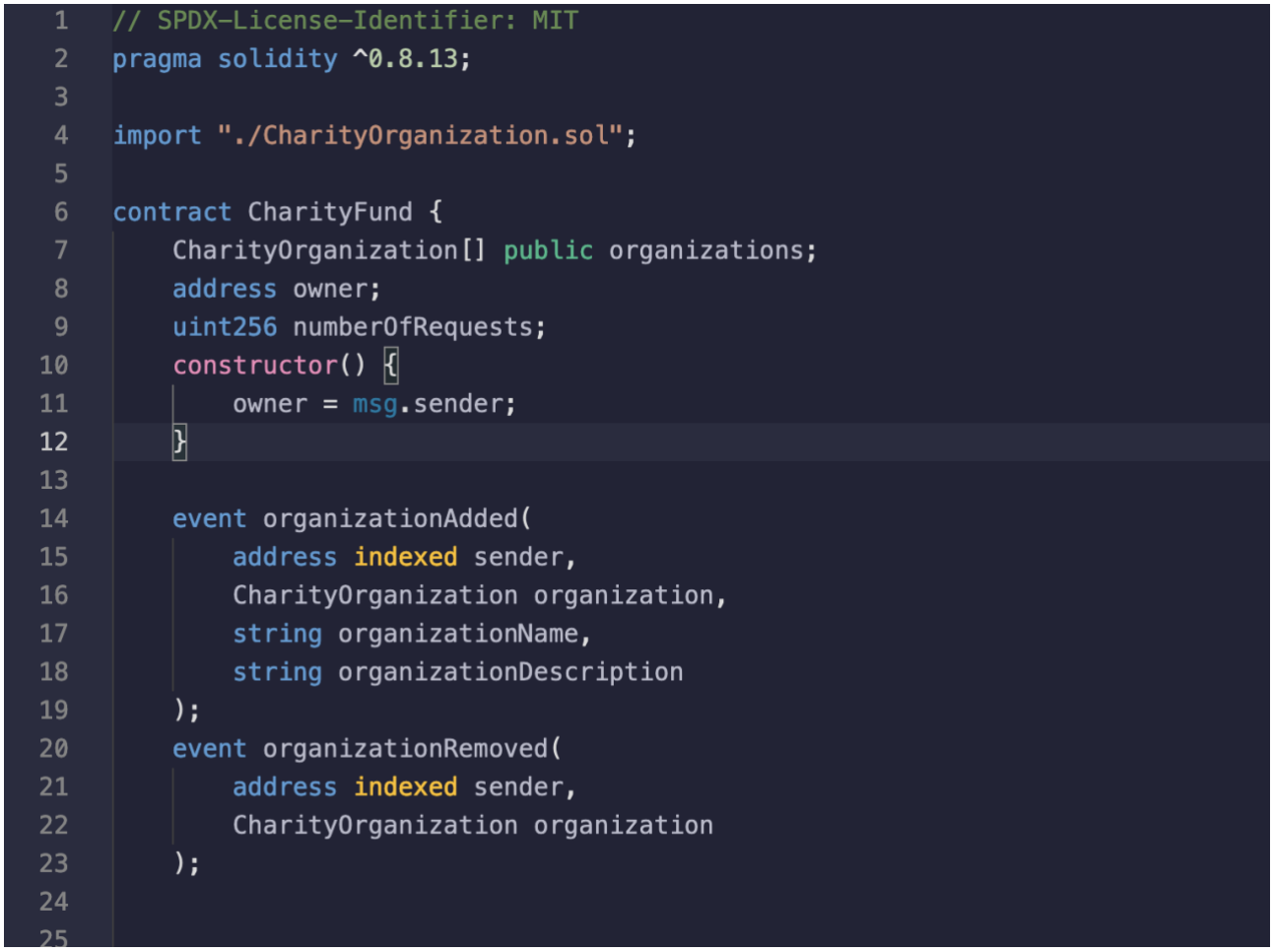
Смарт контракт CharityOrganization містить такі основні атрибути:

- organizationName – назва організації;
- organizationDescription – опис огранізації;
- organizationManager – менеджер цієї організації;
- donators – список благодійників;

та методи, що будуть відповідати на запити:

- organizationDetails – повертає детальну інформацію про благодійну організацію;
- donate – метод для створення пожертви;
- getTokenValue – повертає суму всіх пожертв в доларовому еквіваленті.

Смарт-контракт благодійної фонду



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

import "./CharityOrganization.sol";

contract CharityFund {
    CharityOrganization[] public organizations;
    address owner;
    uint256 numberOfRequests;
    constructor() {
        owner = msg.sender;
    }

    event organizationAdded(
        address indexed sender,
        CharityOrganization organization,
        string organizationName,
        string organizationDescription
    );
    event organizationRemoved(
        address indexed sender,
        CharityOrganization organization
    );
```

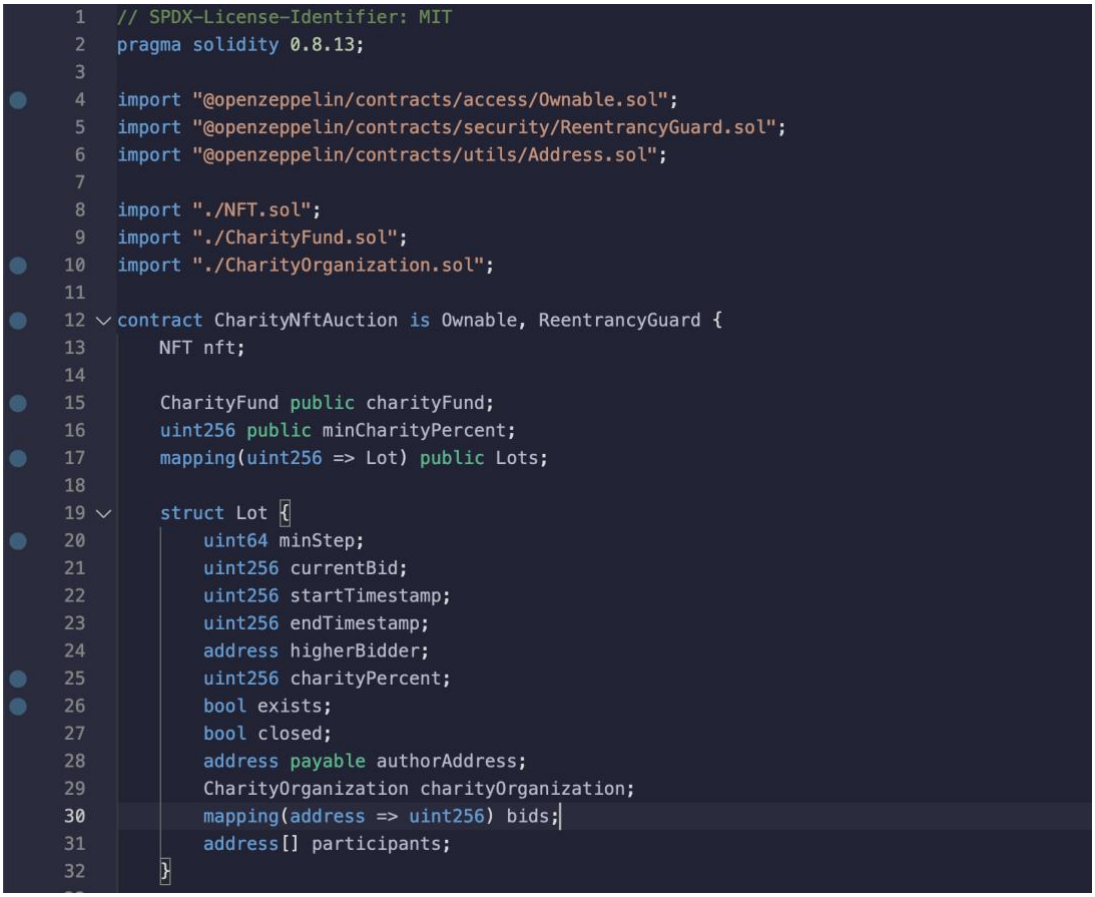
Рисунок 3.7 – Структура смарт контракту CharityFund.sol

Смарт контракт CharityOrganization містить такі основні атрибути:

- organizations – список благодійних організації;
- owner – керівник цього фону.

та методи, що будуть відповідати на запити:

- addOrganization – метод для створення нової благодійної організації;
- removeOrganization – метод для виделення організації.



```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.13;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "@openzeppelin/contracts/utils/Address.sol";

import "./NFT.sol";
import "./CharityFund.sol";
import "./CharityOrganization.sol";

contract CharityNftAuction is Ownable, ReentrancyGuard {
    NFT nft;

    CharityFund public charityFund;
    uint256 public minCharityPercent;
    mapping(uint256 => Lot) public Lots;

    struct Lot {
        uint64 minStep;
        uint256 currentBid;
        uint256 startTimestamp;
        uint256 endTimestamp;
        address higherBidder;
        uint256 charityPercent;
        bool exists;
        bool closed;
        address payable authorAddress;
        CharityOrganization charityOrganization;
        mapping(address => uint256) bids;
        address[] participants;
    }
```

Рисунок 3.9 – Структура смарт контракту CharityFund.sol

Смарт контракт CharityNftAuction наслідується від базових класів Ownable та ReentrancyGuard, та містить такі основні атрибути:

- charityFund – благодійний фонд ;
- minCharityPercent – мінімальний процент комісії благодійності;
- Lots – список активних лотів;

та власні методи, що будуть відповідати на запити:

- createLot – метод для створення нового лоту;
- makeBid – метод щоб зробити ставку на вибраний лот;
- WithdrawRefund – метод для виведення коштів та NFT;
- closeLot – метод для закриття та видалення лоту;
- minAmoutForBid – повертає мінімальну суму щоб перебити актуальну ставку.

Сутності Lot являє собою структуру з 12 атрибутів:

- minStep – мінімальний крок для підвищееня ставки;
- currentBid – актуальну найвищу ставку;
- startTimestamp – час початку аукціону;
- endTimestamp – час коли закінчується аукціон;
- higherBidder – адреса переможця лоту;
- exist – логічне значення, чи існує ще цей лот;
- closed – логічне значення, чи закритий лот;
- authorAddress – адреса власника лоту;
- charityOrganization – адреса благодійної організації, куди підуть комісійні з цього лоту;
- bids – список всіх ставок зроблених учасниками;
- participants – список учасників;

3.3 Опис розгортання та верифікації контрактів

Ми будемо використовувати JS VM в Remix. Перед розгортанням код необхідно зберегти. Metamask — це розширення для Chrome, яке вводить нас у мережу Ethereum. Його можна завантажити та додати до розширень Chrome безпосередньо з браузера. Після того, як ми приймаємо ліцензійну угоду та ліцензійні ліцензії на розширення MetaMask, ми створюємо обліковий запис, який надає нам «початкові слова». Оскільки ми не працюємо з оригінальними монетами Ethereum, нам потрібно перейти на мережу Goerli, щоб продовжити роботу з підробленими ефірами для створення та тестування програми. Створюємо обліковий запис облікові записи «Charity», «Account 1», «Account 2».

3.3.1 Отримання ефіру від Goerli

Заходимо на сайт goerli-faucet.pk910.de. Потім він каже нам вставити адресу нашого Ethereum запису, та проходимо перевірку на робота, після чого натискаємо на кнопку «Start mining». Під час добування ефіру буде

транслюватись вся інформація: хешрейт, кількість задіяних процесорів, середня швидкість видобування за одну годину та кількість добутого ефіру.

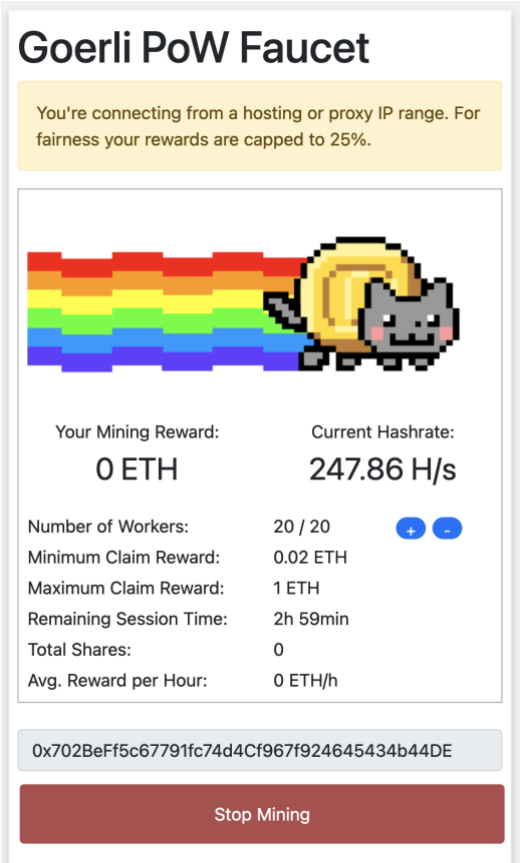


Рисунок 3.10 Процес добування ефіру

Після того, як ми заробимо достатню кількість ефіру можна зупиняти добування та отримати його на баланс.

3.3.2 Розгортання в Goerli TestNet

Щоб розгорнути контракти, які були створені в Remix, нам потрібно переконатися, що ми отримали ефір в наш обліковий запис MetaMask Goerli TestNet. Після того як переконалися, що ми отримали ефір, переходимо до редактора Remix і вибираємо середовище Injected Web 3.0, щоб підключити нашу програму до MetaMask і Goerli. Вибираємо смарт контракт який будемо розгортувати, після чого спочатку компілюємо та натискаємо розгорнути, як показано на рисунку 3.11.

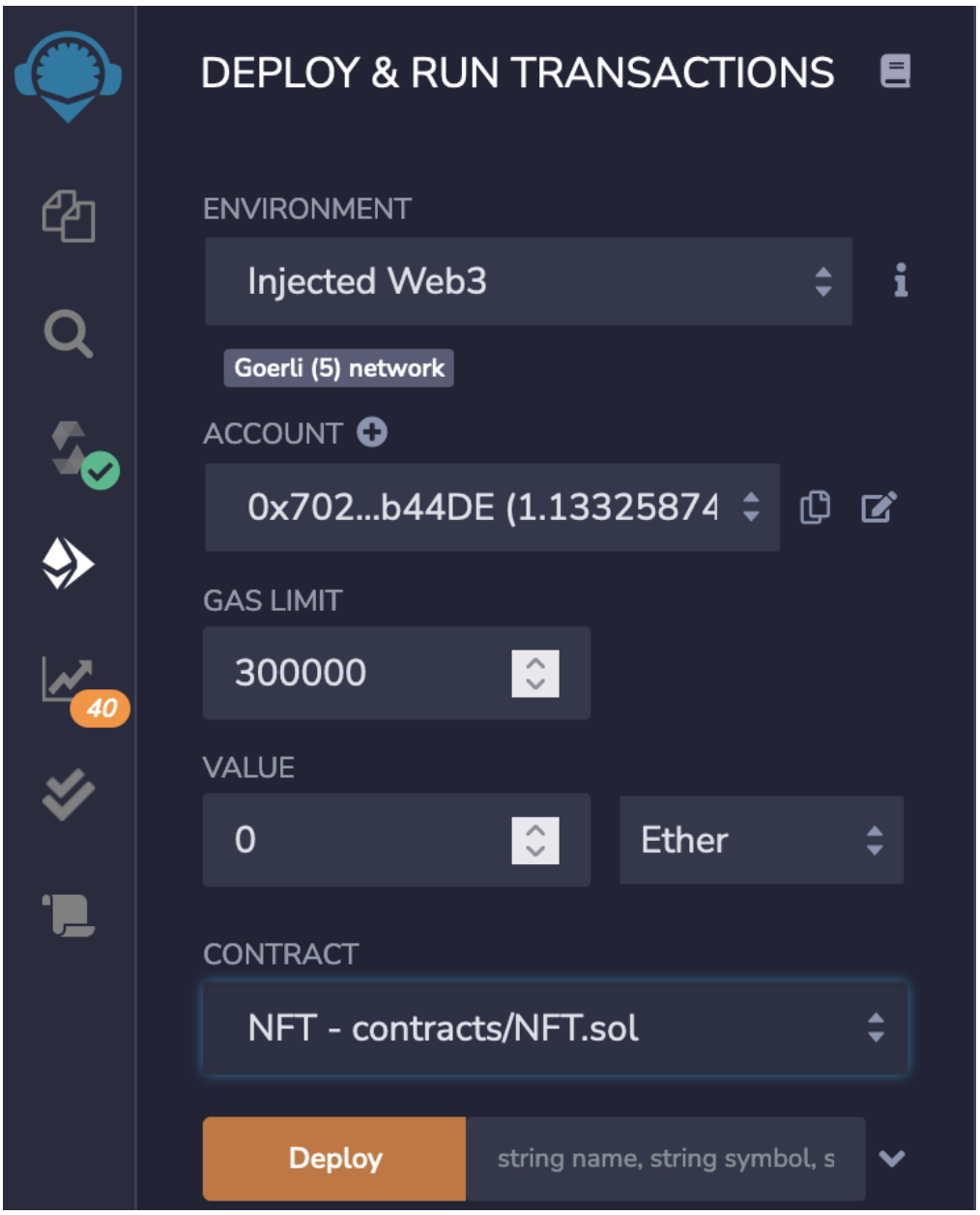


Рисунок 3.11 Вигляд форми для розгортання смарт-контракту

MetaMask запитає у нас підтвердження транзакції, яке ми повинні підтвердити у вікні, що з’явилося, яке показано на рисунку 3.12. Для підтвердження транзакції ми можемо перевірити всі дані, адресу отримувача та вартість транзакції. Для пришвидшення проходження транзакції можна зменшити використання газу, що в свою чергу підніме її вартість., або навпаки зменшити для економії ефіру. Після перевірки всіх даних необхідно підтвердити транзакцію, натиснувши на кнопку «Підтвердити».

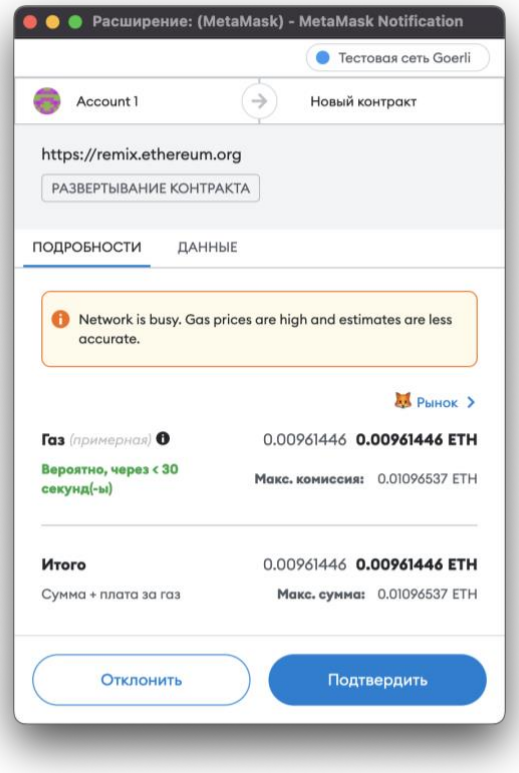


Рисунок 3.12 Вигляд запиту MetaMask для проведення транзакції

Після виконання Remix надає нам посилання etherscan, яке можна використовувати для перевірки того, що блок створюється, показано на рисунку 3.13.

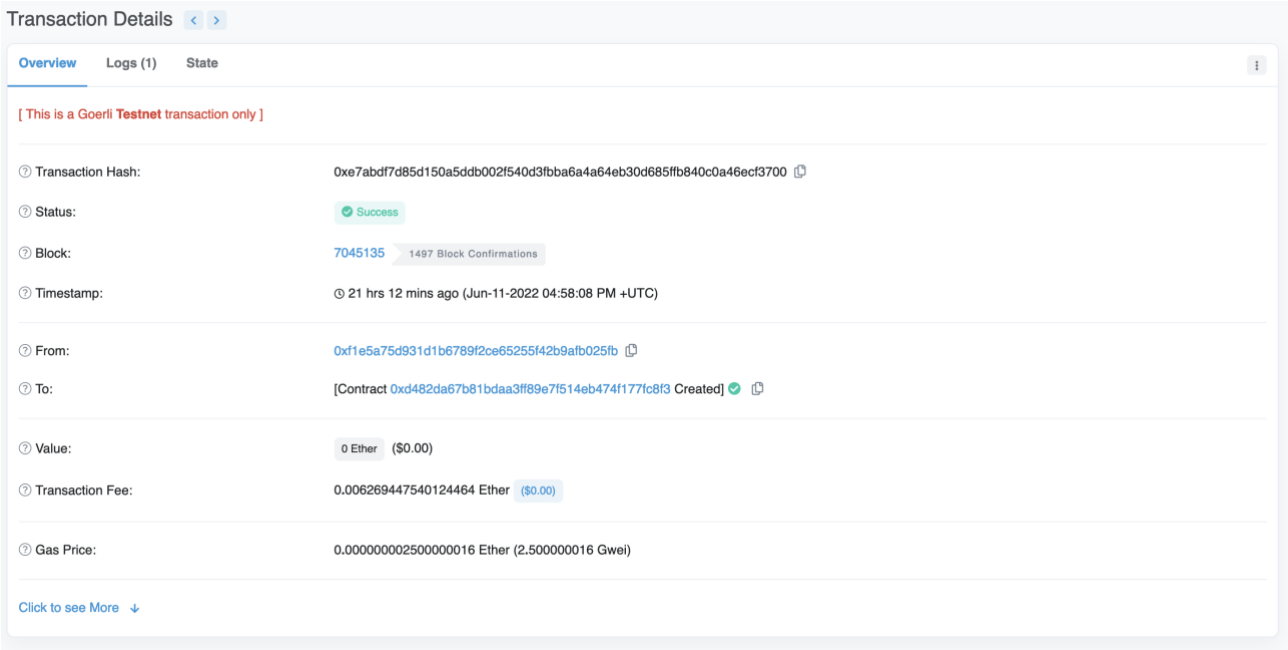


Рисунок 3.13 Вигляд створеної транзакції на etherscan

Статус TX стане успішним після того, як його буде підтверджено, і адресу можна буде використовувати для вказівки на адресу створеного контракту. Такі операції потрібно виконати для кожного з створених смарт-контрактів: NFT, CharityOrganization, CharityFund та CharityNftAuction.

3.3.3 Верифікація розгорнутих смарт контрактів

Для того щоб повноцінно використовувати розгорнуті вами смарт-контракти, потрібно пройти верифікацію, в мережі в якій вони були розгорнуті. Для цього в першу чергу встановимо розширення «FLUTTENER» для нашого робочого середовища Remix. Після чого перейдемо на сторінку смарт-контракту для якого хочемо пройти верифікацію, та у вкладенні «Contract» натиснути на посилання «Verify and Publish», як показано на рисунку 3.14.

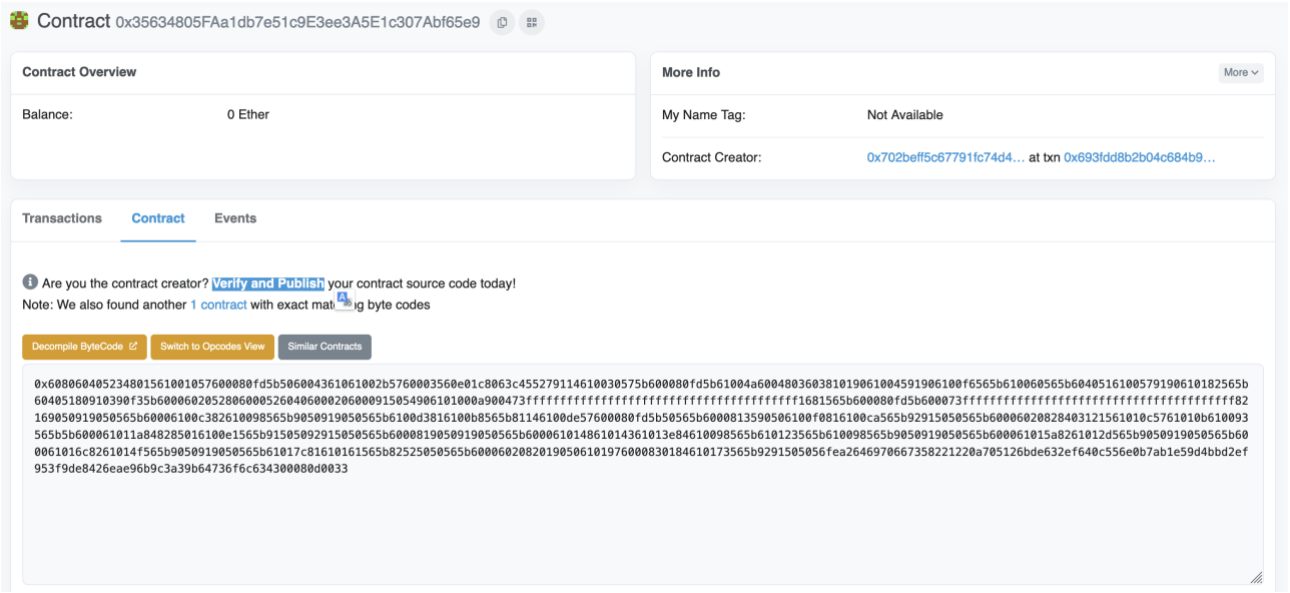


Рисунок 3.14. Вигляд неверифікованого контракту

У новому вікні потрібно заповнити всю необхідну інформацію про цей контракт, як показано на рисунку 3.15: тип та версія компілятора, вид ліцензії (хто проводив аудит контракту) та повний код контракту в одному файлі, який ми скопіюємо з встановленого розширення FLUTTER.

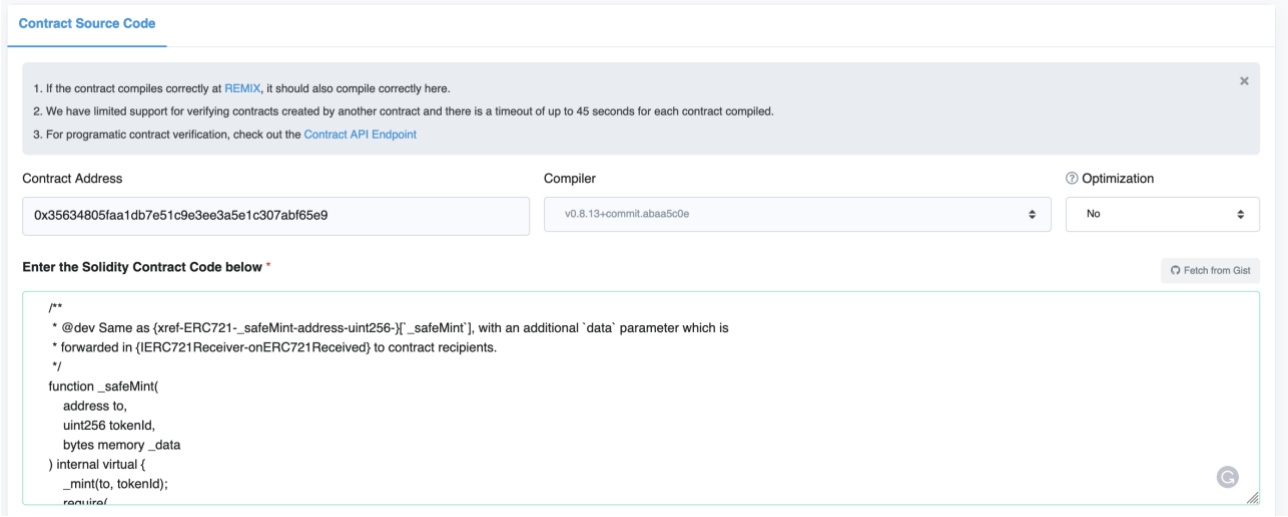


Рисунок 3.15 Форма для проходження верифікації

Після підтвердження, успішна верифікація займе не більше тридцяти секунд, після чого можна перейти на сторінку контракту, щоб перевірити що він пройшов верифікацію (зелений прапорець) та повна інформація про цей контракт з відкритим кодом, як показано на рисунку 3.16.

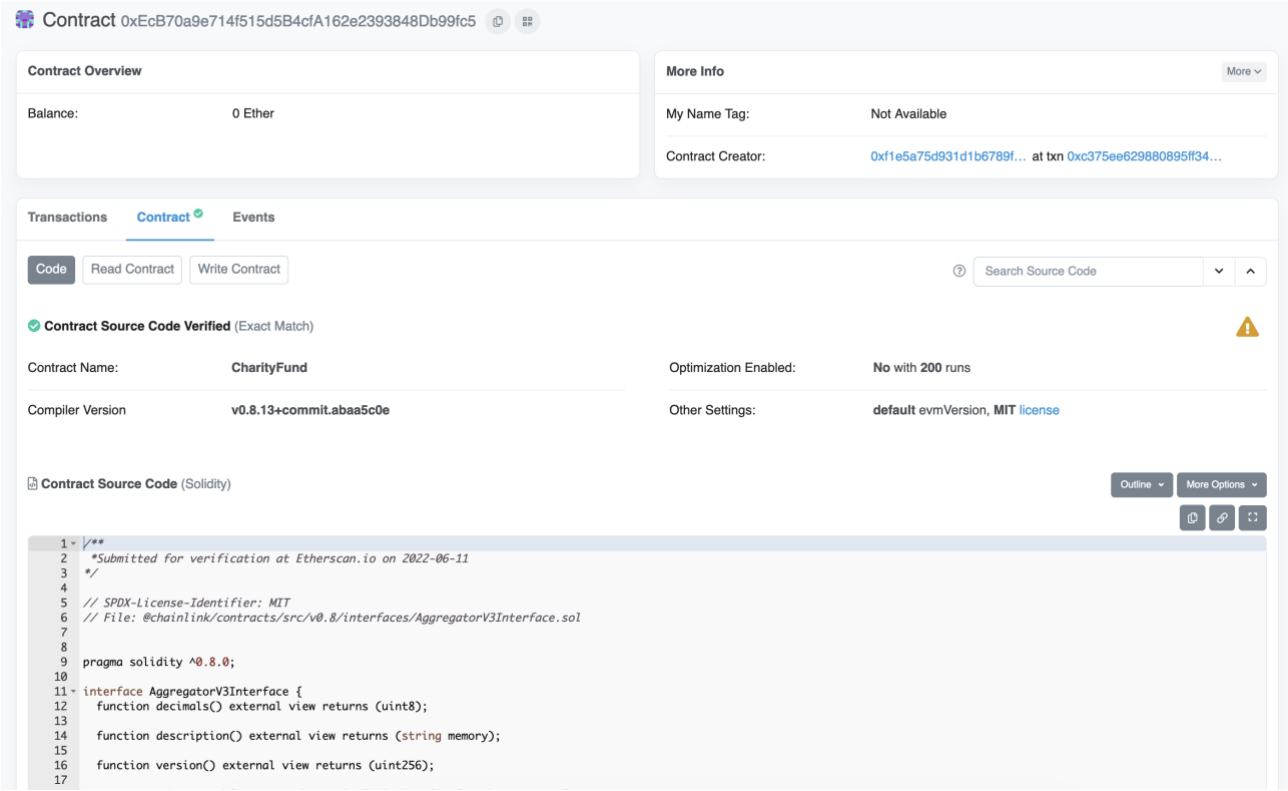


Рисунок 3.16 Вигляд публічного смарт-контракту

Тепер для будь якого користувача будуть доступні всі операції (GET та POST запитів) для використання, як показано на рисунку 3.17 та рисунку 3.18 відповідно.

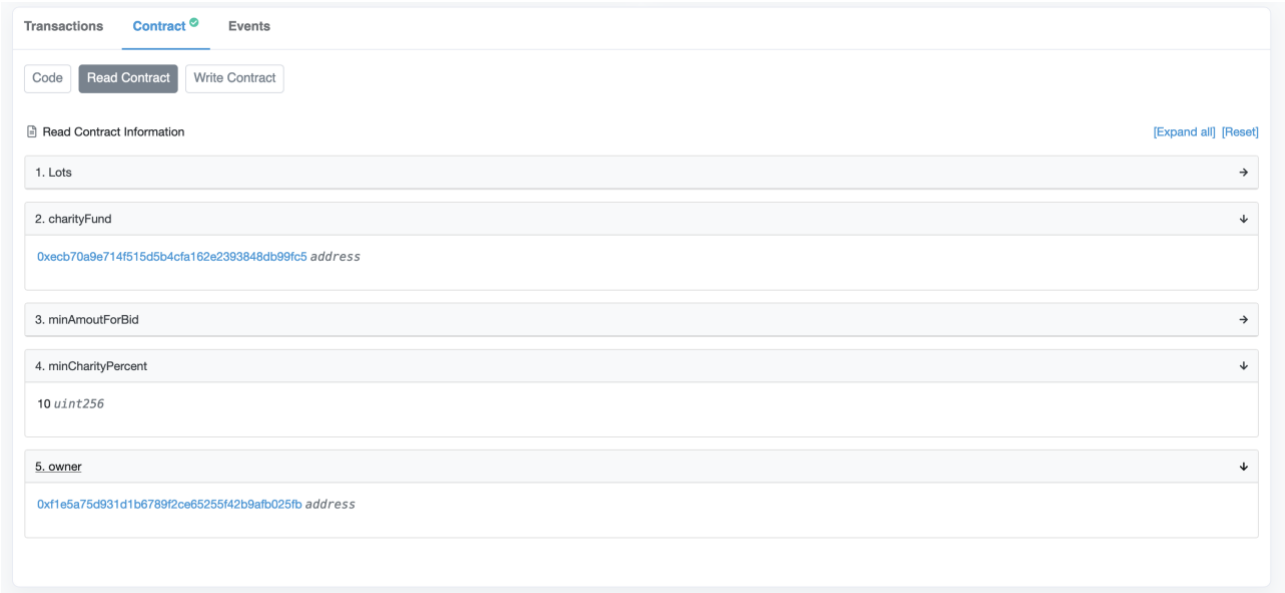


Рисунок 3.17 Методи GET запитів контракту CharityNftAuction

Рисунок 3.18 Методи POST запитів контракту CharityNftAuction

Так за читання контракту (використання методів GET) плата не стягується, а за POST запити необхідно оплачувати комісію транзакції. Потрібно дослідити вартість використання кожного з write методів усіх створених смарт-контрактів, чим ми і займемося в наступному етапі тестування та аналізі розробленої системи.

3.4 Тестування та аналіз розробленої системи

Для тестування створимо всі необхідні сутності: декілька NFT, благодійну організацію та новий лот аукціону. Проведемо декілька торгів NFT лотів, з використанням всіх можливих методів та оцінимо середню вартість використання кожного виклику.

3.4.1 Вартість використання

Так як, вартість проведення транзакції залежить саме від кількості використаного газу, який напряму прив'язаний до токену блокчейна в якому було розгорнуто смарт-контракт (ether в нашому випадку), то вартість використання змінюється в залежності від курсу цього токену. На таблиці 3.2 продемонстровано собівартість розгортання та використання кожного методу відповідних смарт-контрактів.

Таблиця 3.2

Контракт	Метод	Мінімум, ETH	Максимум, ETH	Середнє, ETH	Середня вартість, $
CharityOrganization	donate	0,0000591	0,00007823	0,00006108	0,073296
CharityFund	addOrganization	0,00260459	0,00341144	0,00295992	3,551904
CharityFund	deleteOrganization	0,00008775	0,00011494	0,00009055	0,10866
NFTAuction	createLot	0,00053334	0,00063432	0,00055037	0,660444
NFTAuction	closeLot	0,00011583	0,00015783	0,00009528	0,114336
NFTAuction	makeBid	0,00032355	0,0003848	0,0002886	0,34632
NFTAuction	withdrawRefund	0,00011583	0,00015783	0,00007892	0,094704
NFT	mintTo	0,00028304	0,00031042	0,00030111	0,361332
NFT	setApprovalForAll	0,00007392	0,00008107	0,00007864	0,094368
NFT	safeTransferFrom	0,00391323	0,00595974	0,00435172	5,222064
Розгортання					
NFT	-	0,00615016	0,00767297	0,0066574	7,98888
CharityFund	-	0,0032945	0,00391825	0,00380071	4,560852
NFTAuction	-	0,00391323	0,00501556	0,00402483	4,829796
CharityOrganization	-	0,00260459	0,00341144	0,00295992	3,551904

Кожен метод який змінює стан смарт-контакту бере комісію за проведення транзакції. Та користувач може підвищувати або зменшувати комісію, що буде прямо впливати на швидкість проведення цієї транзакції, тобто чим більша комісія, тим швидше пройде наша транзакція. Тому у таблиці 3.2 є стовбці з мінімальною, максимальною та середньою комісією, яка розрахована в ETH. Ether — це токен валюти, що використовується Ethereum для оплати комісій за транзакції. Середня вартість використання методу в долларах США дорівнює середній вартості проведення в ETH помножений на актуальний курс цього токена.

На момент написання цієї дипломної роботи курс 1 ETH = 1000$, а максимальний курс за всю історію був майже 5000$. Тому зараз розгортання та використання таких смарт-контрактів є незатратним, наприклад, щоб створити новий лот необхідно заплатити лише 66 центів, а щоб зробити ставку – 35 центів. Проте в майбутньому для економії доцільно створити та розгорнути подібні смарт-контракти на дешевших блокчейнах, таких як: Solana, Cardano або Harmony.

3.4.2 Аналіз прозорості системи

Транзакція — це пакет даних, підписаний зовнішнім обліковим записом. Він містить відправника транзакції, одержувача транзакції, кількість надісланого ефіру, необов’язкове поле даних, ліміт газу та поле ціни на газ. Кожного разу, коли поточний код облікового запису контракту виконує код операції «виклику», генерується повідомлення, яке надсилається контракту одержувача або зовнішньому обліковому запису. І транзакції, і виклики можуть використовуватися для створення нових контрактів, для виклику функцій контракту або для передачі ефіру в контракт або на зовнішній обліковий запис. Тому будь-які маніпуляції зі смарт-контрактом записуються, та добавляються в історію транзакцій, як показано на рисунку 3.19.

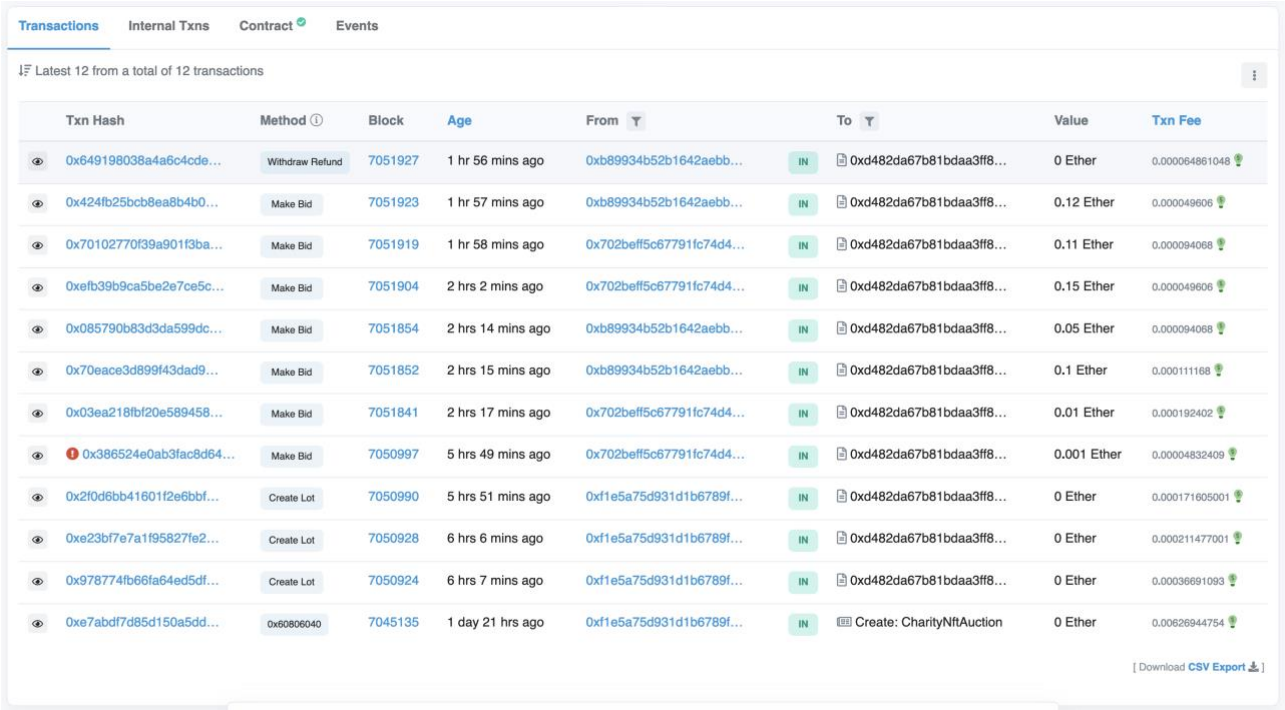


Рисунок 3.20 Історія транзакцій смарт-контракту

Також записуються всі задекларовані типи подій в смарт-контракті, які теж не змінні та можна переглянути будь-якому користувачеві, як показано на рисунку 3.20.

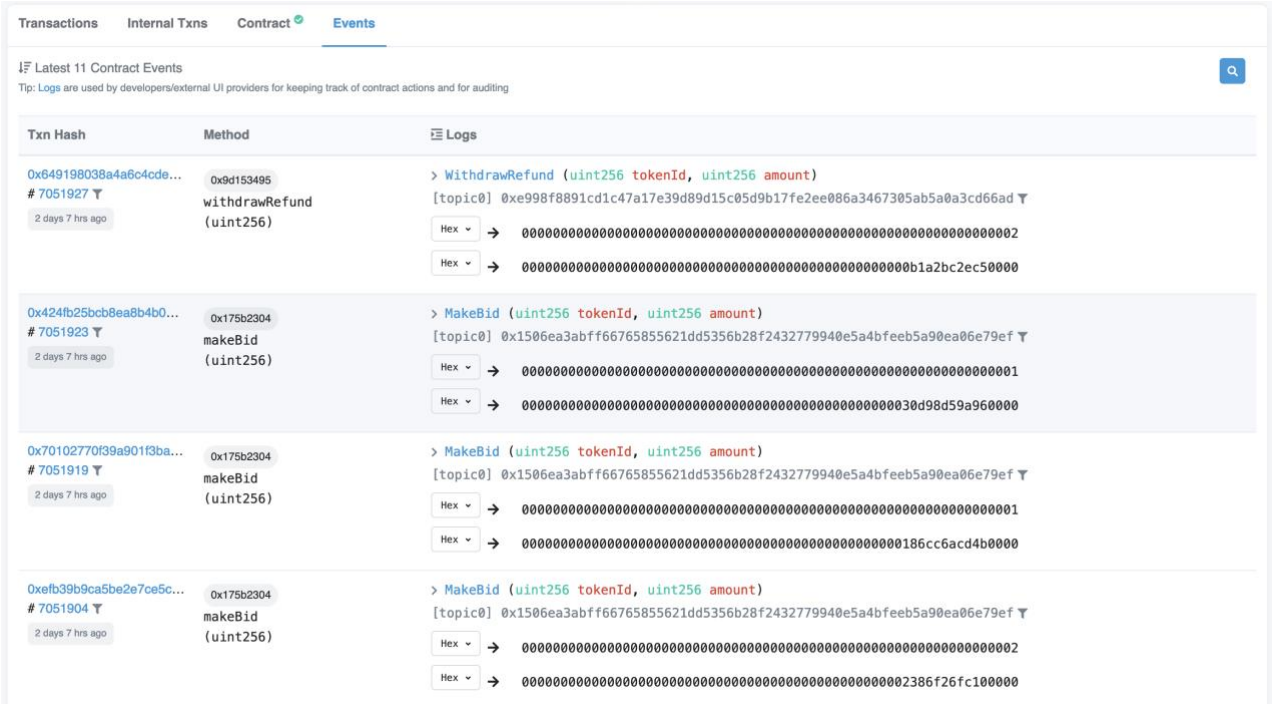


Рисунок 3.20 Історія подій смарт-контракту

Події – це інтерфейси, які дозволяють використовувати засоби реєстрації EVM. Ці інтерфейси використовуються програмами для підписки на події контракту та для моніторингу стану контракту без безпосередньої взаємодії з ним. Коли подія викликається, її аргументи зберігаються в структурі даних журналу транзакцій блокчейну, яка недоступна безпосередньо з контракту. Хеш підпису події використовується для ідентифікації і називається темою. Щоб дозволити пошук конкретних значень аргументів теми, до трьох параметрів події можуть отримати атрибут «індексований».

Висновки до розділу 3

Отже, у результаті виконання третього розділу було розроблено архітектуру та вибрано стек технологій для реалізації системи. Після чого було створено основні чотири смарт-контракти, які успішно були розгорнуті в тестовій мережі Ethereum Goerli.

Були проведені мануально всі функціоналі тестування системи та проаналізовано собівартість використання розробленої системи.

ВИСНОВОКИ

Результатом даної дипломної роботи є розроблена система з організації благодійних NFT аукціонів. Ця система вирішує задачу заохочення до благодійності та гарантує цілковиту прозорість та безпеку коштів.

Будь-який користувач системи може анонімно прийняти участь в аукціоні або напряму пожертвувати кошти вибраній благодійній організації. Після завершення аукціону NFT надсилається переможцю, а виручені кошти розділяються між власником лоту та благодійною організацією відповідно до налаштувань.

У ході виконання роботи було:

1. Проведено аналіз питання благодійності криптовалютного сектора та існуючих систем пожертвування та благодійних NFT аукціонів. Виділено сильні та слабкі сторони знайдених систем.
2. Проаналізовано безпечність та стабільність технології блокчейн та ознайомились з застереженнями під час реалізації смарт-контрактів.
3. Проаналізовано та обрано потужні та дієві технології реалізації системи.
4. В процесі розробки було створено діаграми прецедентів, для правильної побудови архітектури системи, використавши уніфіковану мову моделювання UML.
5. Спроектовано та реалізовано прозору систему для проведення благодійних NFT аукціонів із гнучким функціоналом.
6. Здійснено мануальне тестування, під час якої було проаналізовано безпечність та собівартість розробленої системи.

Дана робота є готова та актуальною для використання. За допомогою цієї системи можна уже влаштовувати NFT аукціони та збирати кошти на благодійність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. **Бурков А.І.** Ethereum: работа с сетью, смарт-контракты и распределенные приложения/ Бурков Олексій Іванович – Москва: ЛітРес, 2022. – ISBN 978-504-28838-4-2
2. **Когут Ю.І.** Технології блокчейн та криптовалюта: ризики та кібербезпека/ Когут Юрий Иванович – Київ: Дакор, 2022. – 316с. – ISBN 978-617-95100-7-6
3. **Смарт Н.** Криптография/ Смарт Нігель – Москва: Техносфера, 2005. ISBN 5-94836-043-1
4. **Тапскотт Д.** Блокчейн-революція/ Дон Тапскотт, Алекс Тапскотт. – К.: Літопис, 2019. – 488 с. - ISBN 978-966-8853-58-6
5. **Прасти Н.** Блокчейн. Розробка додатків/ Нараян Прасті - Санкт-Петербург: БХВ-Петербург, 2018 - 256с - ISBN 978-597-75397-6-0
6. **Фролов О.В** Створення смарт-контрактів Solidity для блокчейну Ethereum. Практичний посібник/ Фролов Олександр В'ячеславович – Москва: ЛітРес, 2022. – ISBN 978-504-19292-5-1
7. **Лелу Л.** Блокчейн від А до Я. Все про технологію десятиліття/ Лоран Лелу – Москва: Ексмо, 2022 – ISBN 978-504-10071-7-1
8. **Н.Атці, М.Бартолетті, Т.Чімолі**: Огляд атак на смарт-контрати Ethereum. Retrieved from https://eprint.iacr.org/2016/1007.pdf
9. **Vitalik Buterin**. Thinking About Smart Contract Security, 2016. Retrieved from https://blog.ethereum.org/2016/06/19/thinking-smart-contract-security/.
10. **Vitalik Buterin.** Visions, part 1: The value of blockchain technology, 2015. Retrieved from https://blog.ethereum.org/2015/04/13/visions-part-1-the-value-ofblockchain-technology
11. **Vitalik Buterin.** 2016. Critical update re: Dao vulnerability. Blog. Retrieved from https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability
12. Ethereum Homestead Documentation - What is Ethereum, 2016. Retrieved from http://ethdocs.org/en/latest/introduction/what-is-ethereum.html
13. Guide to Solidity https://github.com/ethereum/wiki/wiki/BE-Solidity

ДОДАТОК А

Лістинг програми

NFT.sol – смарт-контракт для створення та керування тестовим NFT.

https://goerli.etherscan.io/address/0x3b2814744b5714326dfe70177fd9324ae6edc494

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/utils/Strings.sol";
import "@openzeppelin/contracts/utils/math/SafeMath.sol";

import "./common/meta-transactions/ContentMixin.sol";
import "./common/meta-transactions/NativeMetaTransaction.sol";

contract OwnableDelegateProxy {}

contract ProxyRegistry {
    mapping(address => OwnableDelegateProxy) public proxies;
}

/**
 * @title ERC721Tradable
 * ERC721Tradable - ERC721 contract that whitelists a trading address, and has minting functionality.
 */
abstract contract ERC721Tradable is ERC721URIStorage, ContextMixin, NativeMetaTransaction, Ownable {
    using SafeMath for uint256;
    using Counters for Counters.Counter;

    /**
     * We rely on the OZ Counter util to keep track of the next available ID.
     * We track the nextTokenId instead of the currentTokenId to save users on gas costs.
     * Read more about it here: https://shiny.mirror.xyz/OUampBbIz9ebEicfGnQf5At_ReMHlZy0tB4glb9xQ0E
     */
    Counters.Counter private _nextTokenId;
    address proxyRegistryAddress;

    constructor(
        string memory _name,
        string memory _symbol,
        address _proxyRegistryAddress
    ) ERC721(_name, _symbol) {
        proxyRegistryAddress = _proxyRegistryAddress;
        // nextTokenId is initialized to 1, since starting at 0 leads to higher gas cost for the first minter
        _nextTokenId.increment();
        _initializeEIP712(_name);
    }

    function mintTo(address _to, string memory _tokenURI) public onlyOwner {
        uint256 currentTokenId = _nextTokenId.current();
        _nextTokenId.increment();
        _safeMint(_to, currentTokenId);
        _setTokenURI(currentTokenId, _tokenURI);
    }

    function totalSupply() public view returns (uint256) {
        return _nextTokenId.current() - 1;
    }
```

```
    /**
     * Override isApprovedForAll to whitelist user's OpenSea proxy accounts to enable gas-less listings.
     */
    function isApprovedForAll(address owner, address operator)
        override
        public
        view
        returns (bool)
    {
        // Whitelist OpenSea proxy contract for easy trading.
        ProxyRegistry proxyRegistry = ProxyRegistry(proxyRegistryAddress);
        if (address(proxyRegistry.proxies(owner)) == operator) {
            return true;
        }

        return super.isApprovedForAll(owner, operator);
    }

    /**
     * This is used instead of msg.sender as transactions won't be sent by the original token owner, but by OpenSea.
     */
    function _msgSender()
        internal
        override
        view
        returns (address sender)
    {
        return ContextMixin.msgSender();
    }
}
// SPDX-License-Identifier: MIT
pragma solidity 0.8.13;

import "./ERC721Tradable.sol";

contract NFT is ERC721Tradable {
    string private _baseTokenURI;

    string private _contractURI;

    constructor(
        string memory name,
        string memory symbol,
        string memory baseTokenUri,
        string memory contractUri,
        address _proxyRegistryAddress
    ) ERC721Tradable(name, symbol, _proxyRegistryAddress) {
        _baseTokenURI = baseTokenUri;
        _contractURI = contractUri;
    }

    function _baseURI() internal view override returns (string memory) {
        return _baseTokenURI;
    }

    function contractURI() public view returns (string memory) {
        return _contractURI;
    }
}
```

CharityOrganization.sol - смарт-контракт благодійної організації.

https://goerli.etherscan.io/address/0x442a967844d2cd75474d6b697f779aa946c24327

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

contract CharityOrganization is Ownable {
    string public organizationName;
    string public organizationDescription;
    address public organizationManager;
    uint256 public minimumDonation;
    uint256 public numberOfRequests;
    address[] public donators;
    mapping(address => bool) IsAlreadyDonator;
    mapping(address => address) tokenToPriceFeed;

    constructor(
        address _organizationManager,
        string memory _organizationName,
        string memory _organizationDescription,
        uint256 _minimumDonation
    ) {
        organizationManager = _organizationManager;
        organizationName = _organizationName;
        organizationDescription = _organizationDescription;
        minimumDonation = _minimumDonation;
    }

    function organizationDetails()
        public
        view
        returns (
            string memory,
            string memory,
            uint256
        )
    {
        return (organizationName, organizationDescription, minimumDonation);
    }

    function setPriceFeedContract(address _token, address _priceFeed)
        public
        onlyOwner
    {
        tokenToPriceFeed[_token] = _priceFeed;
    }

    // This function accepts the donation only in ether
    function donate(uint256 _amount) public payable {
        require(
            _amount >= minimumDonation,
            "Donation Amount is less than the minimum Amount"
        );
        // Check if the donator already donated to this organization?
        // If not then insert this donator's address in the "donators" list
        if (IsAlreadyDonator[msg.sender] == false) {
            IsAlreadyDonator[msg.sender] = true;
            donators.push(address(msg.sender));
        }
    }
```

```
    function getTokenValue(address _token)
        public
        view
        returns (uint256, uint256)
    {
        address priceFeedAddress = tokenToPriceFeed[_token];
        AggregatorV3Interface priceFeed = AggregatorV3Interface(
            priceFeedAddress
        );
        (, int256 price, , , ) = priceFeed.latestRoundData();
        uint256 decimals = uint256(priceFeed.decimals());
        return (uint256(price), decimals);
    }
}
```

CharityFund.sol – смарт-контакт благодійного фонду.

https://goerli.etherscan.io/address/0xecb70a9e714f515d5b4cfa162e2393848db99fc5

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

import "./CharityOrganization.sol";

contract CharityFund {
    CharityOrganization[] public organizations;
    address owner;
    uint256 numberOfRequests;
    constructor() {
        owner = msg.sender;
    }

    event organizationAdded(
        address indexed sender,
        CharityOrganization organization,
        string organizationName,
        string organizationDescription
    );
    event organizationRemoved(
        address indexed sender,
        CharityOrganization organization
    );


    function addOrganization(
        string memory _organizationName,
        string memory _organizationDescription,
        uint256 _minimumContribution
    ) public {
        CharityOrganization organization = new CharityOrganization(
            msg.sender,
            _organizationName,
            _organizationDescription,
            _minimumContribution
        );
        organizations.push(organization);
        emit organizationAdded(
            msg.sender,
            organization,
            _organizationName,
            _organizationDescription
        );
    }
```

```
    function deleteOrganization(
        CharityOrganization _organization
    ) public {
        for (uint i=0; i<organizations.length; i++) {
            if (organizations[i] == _organization) {
                organizations[i] = organizations[organizations.length - 1];
                organizations.pop();
            }
        }
        emit organizationRemoved(
            msg.sender,
            _organization
        );
    }
}
```

ChartityNFTAuction.sol – смарт-контракт благодійного NFT аукціону.
https://goerli.etherscan.io/address/0xd482da67b81bdaa3ff89e7f514eb474f177fc8f3

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.13;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/security/ReentrancyGuard.sol";
import "@openzeppelin/contracts/utils/Address.sol";

import "./NFT.sol";
import "./CharityFund.sol";
import "./CharityOrganization.sol";

contract CharityNftAuction is Ownable, ReentrancyGuard {
    NFT nft;

    CharityFund public charityFund;
    uint256 public minCharityPercent;
    mapping(uint256 => Lot) public Lots;

    struct Lot {
        uint64 minStep;
        uint256 currentBid;
        uint256 startTimestamp;
        uint256 endTimestamp;
        address higherBidder;
        uint256 charityPercent;
        bool exists;
        bool closed;
        address payable authorAddress;
        CharityOrganization charityOrganization;
        mapping(address => uint256) bids;
        address[] participants;
    }

    event CreateLot(
        uint256 tokenId,
        uint64 minStep,
        uint256 initialBid,
        uint256 startTimestamp,
        uint256 endTimestamp,
        uint256 charityPercent,
        address authorAddress,
        CharityOrganization charityOrganization
    );
```

```
event MakeBid(uint256 tokenId, uint256 amount);

event WithdrawRefund(uint256 tokenId, uint256 amount);

event CloseLot(uint256 tokenId, address recipient, uint256 amount);

event PurchaseItem(address recipient, uint256 tokenId);

constructor(NFT _nft, uint256 _minCharityPercent, CharityFund _charityFund) {
   nft =   nft;
   minCharityPercent = _minCharityPercent;
   charityFund = _charityFund;
}

function createLot(
   uint64 minStep,
   uint256 tokenId,
   uint256 initialBid,
   uint256 startTimestamp,
   uint256 endTimestamp,
   uint256 charityPercent,
   address payable authorAddress,
   CharityOrganization charityOrganization
) external onlyOwner {
   require(!Lots[tokenId].exists, "Lot already exists");
   require(charityPercent >= minCharityPercent, "Fees are too high");
   require(authorAddress != address(0), "Wrong author address");
   require(address(charityOrganization) != address(0), "Wrong charity address");

   Lot storage newLot = Lots[tokenId];

   newLot.minStep = minStep;
   newLot.currentBid = initialBid;
   newLot.startTimestamp = startTimestamp;
   newLot.endTimestamp = endTimestamp;
   newLot.higherBidder = owner();
   newLot.charityPercent = charityPercent;
   newLot.exists = true;
   newLot.closed = false;
   newLot.authorAddress = authorAddress;
   newLot.charityOrganization = charityOrganization;

   emit CreateLot(
      tokenId,
      minStep,
      initialBid,
      startTimestamp,
      endTimestamp,
      charityPercent,
      authorAddress,
      charityOrganization
   );
}

modifier lotIsActive(uint256 tokenId) {
   require(
      Lots[tokenId].exists &&
      block.timestamp >= Lots[tokenId].startTimestamp &&
      block.timestamp < Lots[tokenId].endTimestamp &&
      !Lots[tokenId].closed,
      "Lot is not active"
   );
   _;
                                                                                                    }
```

```
function minAmoutForBid(uint256 tokenId) public view returns(uint256) {
    return Lots[tokenId].currentBid + Lots[tokenId].minStep - Lots[tokenId].bids[msg.sender];
}

function makeBid(uint256 tokenId) external payable lotIsActive(tokenId) {
    require(
        msg.value >= minAmoutForBid(tokenId),
        "Bid must be bigger than current bid"
    );
    if (Lots[tokenId].bids[msg.sender] == 0) {
        Lots[tokenId].participants.push(msg.sender);
    }

    Lots[tokenId].higherBidder = msg.sender;
    Lots[tokenId].currentBid = msg.value + Lots[tokenId].bids[msg.sender];
    Lots[tokenId].bids[msg.sender] = Lots[tokenId].currentBid;

    emit MakeBid(tokenId, Lots[tokenId].currentBid);
}

function closeLot(uint256 tokenId) external {
    require(Lots[tokenId].exists, "Lot does not exist");
    require(!Lots[tokenId].closed, "Lot already closed");
    require(
        block.timestamp >= Lots[tokenId].endTimestamp,
        "Lot is active"
    );
    require(
        msg.sender == Lots[tokenId].higherBidder || msg.sender == owner(),
        "You can not close Lot"
    );

    Lots[tokenId].closed = true;
    purchaseItem(Lots[tokenId].higherBidder, tokenId);

    for (uint i=0; i<Lots[tokenId].participants.length; i++) {
        address participant = Lots[tokenId].participants[i];
        uint refund = Lots[tokenId].bids[participant];
        if (refund > 0) {
            Address.sendValue(payable(participant), refund);
        }
    }

    delete Lots[tokenId];

    emit CloseLot(
        tokenId,
        Lots[tokenId].higherBidder,
        Lots[tokenId].currentBid
    );
}
```

```
    function withdrawRefund(uint256 tokenId)
        external
        nonReentrant
        lotIsActive(tokenId)
    {
        require(
            msg.sender != Lots[tokenId].higherBidder,
            "Withdraw is not available for the highest bid"
        );
        require(
            Lots[tokenId].bids[msg.sender] > 0,
            "No funds found for refund"
        );

        uint256 refund = Lots[tokenId].bids[msg.sender];
        Address.sendValue(payable(msg.sender), refund);
        Lots[tokenId].bids[msg.sender] = 0;

        emit WithdrawRefund(tokenId, refund);
    }

    function purchaseItem(address recipient, uint256 tokenId)
        internal
        nonReentrant
    {
        require(!Lots[tokenId].closed, "Lot is active!");
        nft.safeTransferFrom(Lots[tokenId].authorAddress, recipient, tokenId);

        uint256 charityAmount = (Lots[tokenId].currentBid * Lots[tokenId].charityPercent) / 100;
        uint256 authorAmount = Lots[tokenId].currentBid - charityAmount;

        Address.sendValue(Lots[tokenId].authorAddress, authorAmount);
        Address.sendValue(payable(address(Lots[tokenId].charityOrganization)), charityAmount);

        emit PurchaseItem(recipient, tokenId);
    }
}
```