Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ДОСЛІДЖЕННЯ МЕТОДІВ СТВОРЕННЯ Й ЕКСПЛУАТАЦІЇ ДИНАМІЧНИХ NFT - СМАРТКОНТРАКТІВ

Текстова частина до кваліфікаційної роботи
за спеціальністю «Комп'ютерні науки» - 122

Керівник курсової роботи
Старший викладач
Гороховський К.С.
_____ (Підпис)
"___" _____ 2024 року
Виконав студент
КН-4 Куценко А.О.
"___" _____ 2024 року

Київ 2024

ЗМІСТ

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Старший викладач
_____ Гороховський К.С.
„_____” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу

студенту Куценку Андрію Олександровичу

факультету інформатики 4 курсу бакалаврської програми

ТЕМА: Дослідження методів створення й експлуатації динамічних NFT - смартконтрактів

Зміст ТЧ до курсової роботи:
Зміст
Індивідуальне завдання
Вступ
1. Дослідження
2. Реалізація власної пропозиції стандарту та смартконтракту на його основі
Висновок
Список використаної літератури

Дата видачі „___” _____ 2024 р. Керівник_____

(підпис)

Завдання отримав_____

(підпис)

Календарний план виконання курсової роботи

№	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання індивідуального завдання	24.10.2024	
2.	Дослідження матеріалів для курсової роботи	25.10.2024	
3.	Написання текстової частини	01.04.2024	
4.	Реалізація власного стандарту	01.05.2024	
5.	Надання роботи керівнику на перевірку	12.05.2024	
6.	Виправлення на основі перевірки	17.05.2024	
7.	Кінцева перевірка роботи	22.05.2024	
8.	Захист курсової роботи	30.05.2024	

Студент _____

Керівник _____

«____» _____ _____р.

ВСТУП

Впродовж останнього десятиліття відбувся стрімкий розвиток технології блокчейну, це переросло з просто звичайної криптовалюти в дещо більше та цікавіше. З появою єдиної децентралізованої віртуальної машини, розробленої Віталіком Бутеріном, під назвою Ethereum – світ блокчейну змінився і розкрив набагато більше можливостей як для розробників, так і для користувачів.

Стрімко почали з'являтися нові криптовалюти, де кожна мала свою поведінку, з'явилися децентралізовані додатки, котрі мали більшу захищеність та стійкість до атак, та з'явилися NFT, що свого часу підняло великий галас серед людей та викликало багато обговорень та дій в бік невідомого до цього людству механізму взаємодії з активами.

І як часто відбувається в світі, розробник або творець може придумати новітні технології, закласти глибокий сенс в свої винаходи або творчість, але користувач це сприйме зовсім по іншому і буде використовувати не так, як планувалося. Гарним прикладом слугуватимуть крокси, котрі задумувалися як засіб для зручного пересування яхтами, але світ сприйняв це як частину моди та просто зручного літнього взуття.

На жаль чи на щастя, NFT в свій час спіткала ж така доля, оскільки люди це сприйняли як просто актив над цифровою картинкою, через що з'явився тренд на їх купівлю. Через обмеженість кількості ціна почала дуже стрімко і штучно зростати, в результаті поширення використання NFT зазнало краху. Після чого у людей з'явилося розчарування, оскільки багато хто втратив кошти. Більшість навіть не хоче спробувати розібратися в тому, що це таке, оскільки у загалу з'являються погані асоціації, коли чують про NFT, через досвід інших.

Після закінчення тренду пройшло чимало часу і поширилися нові технології, як наприклад, динамічні NFT, про що взагалі більшість не знає, хоча це розширює вікно можливостей цифрових активів в багато разів і може використовуватися в багатьох сферах.

Це в майбутньому може стати як і невід'ємною частиною нашого життя, до чого краще бути готовим заздалегідь, так і може взагалі зникнути, не розкривши свій потенціал. Але це все неможливо дізнатися, не спробувавши і не показавши це іншим.

Тому головною метою цієї роботи буде дослідити та показати процес як створення, так і експлуатації новітніх технологій з назвою “Динамічні NFT-смартконтракти”.

1. ДОСЛІДЖЕННЯ

Ця частина присвячена дослідженню того, які бувають NFT, як і де вони застосовуються та зберігаються, розгляду сценаріїв їх використання в сучасному цифровому світі, методів їх створення та експлуатації, огляду EIP та реальних кейсів їх реалізації на блокчейні.

1.1 Визначення

Далі наведений перелік термінів та їх роз'яснення, котрі будуть використовуватися далі в цій науковій роботі:

1.1.1 NFT, смартконтракти

Смартконтракт – це комп’ютерна програма або протокол транзакцій, який призначений для автоматичного виконання, контролю або документування подій і дій відповідно до умов контракту чи угоди. Цілями розумних контрактів є зменшення потреби в довірених посередниках, арбітражних витрат і збитків від шахрайства, а також зменшення зловмисних і випадкових дій. [1]

Іншими словами – це код програми, котрий люди додають до блокчейну, і будь-хто може викликати його функціонал для взаємодії з ним або з іншими програмами через нього. Для кращого розуміння треба уточнити, що блокчейн побудований по принципу singleton, тим самим всі контракти зберігаються на одній віртуальній машині (копії якої зберігають майнери) і вони безперешкодно можуть взаємодіяти один з одним,просто знаючи розташування іншого контракту (його публічний ключ), на кшталт взаємодії одного класу з іншим всередині вашого додатку.

Невзаємозамінний токен (далі NFT) є типом криптотокена на блокчейні, який представляє унікальний об’єкт. NFT може бути як повністю цифровим активом, так і токенізованою версією на реальний актив. Відмінно від криптовалют, які є взаємозамінними, кожен NFT має унікальні ідентифікатори і не може бути обміняний на рівнозначний об'єкт безпосередньо, що робить його не подібним до традиційних криптовалют.

Для кращого розуміння різниці між NFT та криптовалютою можна навести систему онлайн банку та фізичних грошей. Онлайн банк по своїй функціональності схожий на криптовалюти, де за користувачем зберігається баланс відносно його унікального номеру (наприклад, паспортні дані), а ~~в~~–у випадку з фізичними купюрами – користувач (людина) має унікальні банкноти, кожна з яких має свій номер та свою цінність, а не просто абстрактну суму. Так само відбувається з криптовалютою, оскільки за приватним ключем користувача зберігається просто сума балансу, а NFT має унікальні ідентифікатори на кожну одиницю токену, тому вони і є невзаємозамінними, оскільки один токен не має такої ж цінності як інший, тобто вони схожі на фізичні купюри тим, що мають унікальні номери, але у них немає номіналу. І говорячи про цінність NFT, не потрібно думати про їх грошову характеристику, оскільки ці токени можуть описувати що завгодно, наприклад вас як особу і бути вашим паспортом, що продати не можна.

Зазвичай NFT створюють на блокчейні Ethereum за допомогою стандарту ERC-721 або ERC-1155.

1.1.2 Динамічні NFT

Динамічні невзаємозамінні токени (далі dNFT) - це нове покоління незамінних токенів, які включають програмовані та інтерактивні функції, що дозволяє створювати цифрові елементи, які можуть змінюватися та розвиватися з часом. [2]

dNFT використовують смартконтракти для адаптації своїх метаданих або характеристик у реальному часі, що і дає їм можливість бути динамічними.

Статичні NFT:

- зберігають атрибути незмінними та постійно записаними на блокчейн;
- ігнорування-джерел поза мережею;
- не реагують на події реального світу;
- більш надійні і безпечні завдяки незмінним метаданим;
- ефективніші з точки зору вартості, обсягу зберігання та вимог до пропускної здатності;
- використання лише одного файлу метаданих, що полегшує перевірку.

Динамічні NFT:

- атрибути NFT можуть змінюватися залежно від певних обставин;
- підключення до джерел даних у ланцюжку та за його межами, токенізуючи предмети, що прогресують у реальному світі;
- менший рівень безпеки через змінні метадані та можливу атаку;
- вимагають більше витрат на оновлення метаданих;
- наявність кількох файлів метаданих ускладнює їх перевірку;
- більше залучення та більше занурення. [3]

Тому, дивлячись на різницю між цими двома типів токенів, ми бачимо, з одного боку, більшу можливість додавання функціоналу та взаємодії з навколишнім середовищем поза блокчейном, але, з іншого боку, з'являється більше ускладнень та небезпек зі зростанням функціональності.

1.1.3 Оракули

Оракули в блокчейні – це сервіси, котрі забезпечують смартконтракти надійними зовнішніми даними, які вони не можуть отримати самостійно. Вони виступають як провідники між реальним світом і блокчейном, дозволяючи

смартконтрактам взаємодіяти з зовнішніми API, базами даних та іншими веб-сервісами.

Механізм дії оракулів:

1) запит даних: смартконтракт dNFT має потребу в зовнішніх даних (наприклад, поточний курс валют, погода, результати спортивних матчів), і він робить запит до оракула;

2) взаємодія з оракулом: оракул отримує запит від смартконтракту та звертається до зовнішнього джерела, що може бути веб-серсвісом, API, або іншою системою для отримання інформації;

3) перевірка та передача даних: після отримання даних оракул починає перевірку та верифікацію даних, після чого повертає їх смартконтракту;

4) оновлення dNFT: смартконтракт після отримання даних від оракула використовує їх для оновлення власного стану, властивостей, метаданих;

Проблеми:

- надійність даних: одна з найголовніших проблем є залежність від зовнішніх джерел, а отже вони можуть бути схильні до помилок, маніпуляцій, нестабільності доступу, що вплине на поведінку dNFT, тому треба відповідально відноситися до джерел;

- централізація: багато оракулів є централізованими, тому це може створити ризик відмови або централізованих атак;

З'являється відповідне питання: чому ж смартконтракти не можуть напряму звернутися до API умовного сервісу і отримати дані. Основна відповідь на це – консенсус. Блокчейн вимагає єдиного стану в усіх вузлах мережі, тому обчислення контракту повинно мати однакові вхідні дані на всіх вузлах, що звичайне API не може забезпечити, оскільки під час розповсюдження задачі обчислення по мережі вхідні дані можуть змінитися і контракт видасть різний результат, через що він буде відмінений.

Тому наразі оракули є основним інструментом для смартконтрактів dNFT, щоб проявляти свою властивість – динамічність (інші будуть розглянуті як приклад далі).

Одним з найбільш розповсюджених децентралізованих оракулів є Chainlink.

1.1.4 Метадані NFT

В контексті NFT, метадані – це інформація, яка описує характеристики NFT, такі як власник, історія транзакцій, художній зміст або інші властивості, які відносяться до цього токена. Метадані важливі для NFT, оскільки вони дозволяють взаємодіяти з токеном в більш багатому і детальному контексті, надаючи інформацію, яка не зберігається безпосередньо на блокчейні.

Метадані NFT зазвичай представлені у форматі JSON. Ось приклад метаданих для NFT, який представляє цифрове зображення:

```json
{
  "name": "Sunset Artwork",
  "description": "A digital painting of a sunset.",
  "image": "https://example.com/path/to/image.jpg",
  "artist": "John Doe",
  "date": "2024-05-01",
  "attributes": [
    {
      "trait_type": "Color",
      "value": "Orange"
    },
    {
      "trait_type": "Mood",
      "value": "Calm"
    }
  ]
}
```

Рисунок 1.1 – JSON приклад метаданих

Цей приклад JSON описує основні характеристики токена, такі як назву, опис, посилання на зображення, інформацію про автора, дату створення та додаткові атрибути, які можуть визначати колір або настрій NFT.

Метадані можуть зберігатися різними способами:

- на блокчейні: забезпечує безпеку даних, але при цьому є дуже дорогим процесом через обмежену пропускну спроможність та високу вартість транзакцій;

- децентралізовані файлові системи: найчастіше використовуються децентралізовані файлові системи, такі як IPFS (InterPlanetary File System), що знижує ризик відмови;

- традиційні сервери: інколи розробники використовують звичайні хмарні сховища та веб-сервери для зберігання, але такий метод є небезпечним через високий ризик відмови та централізованість.

У контексті динамічних NFT, метадані можуть змінюватися відповідно до зазначених в смартконтракті тригерів або у відповідь на зовнішні події, наприклад, отримані через оракули.

1.1.5 IPFS

IPFS розшифровується як міжпланетна файлова система. У простих термінах, IPFS — це відкритий одноранговий розподілений гіпермедіа-протокол, що діє як універсальна файлова система для обчислювальних пристроїв.

Замість традиційного завантаження файлів з централізованих серверів, в IPFS запити на файли спрямовуються до однорангових вузлів у мережі, які вказують локацію потрібного файлу. Це знижує навантаження на окремі сервери та підвищує ефективність розподілу великих даних. [4]

Поведінка IPFS схожа на торрент, але не є повністю таким. Ключова різниця полягає в тому, що в торренті у кожного файла є рій однорангових зв'язків, в той час IPFS має один рій повністю для всіх даних.

Таким чином, IPFS пропонує стабільну, надійну та ефективну альтернативу традиційним методам веб-зберігання, зокрема в ареалі технологій блокчейн та NFT.

1.2 Сучасний стан dNFT

Технології блокчейну мають настільки стрімкий темп розвитку, що технології не встигають устаткуватися і досягти стандартизації, як вже з'являються більш новаторські ідеї, котрі перебирають увагу на себе. dNFT є одним з таким прикладів.

1.2.1 Історія виникнення NFT

Одним з перших проєктів, що використовував ідею NFT, був "Colored Coins" ще у 2012-2013 роках. Цей проєкт був реалізований на блокчейні Bitcoin і дозволяв "забарвлювати" невелику кількість біткоїнів для представлення активів поза мережею, таких як акції, нерухомість або навіть інтелектуальну власність.

Справжній прорив у розвитку NFT стався із запуском Ethereum, блокчейну з підтримкою смартконтрактів, що забезпечував значно більше можливостей для реалізації складних проєктів. У 2017 році був представлений стандарт ERC-721, перший стандарт для створення невзаємозамінних токенів на Ethereum. ERC-721 дозволив розробникам створювати унікальні цифрові активи, кожен з яких має свої ідентифікатори та властивості.

Популярність NFT стрімко зросла з появою гри CryptoKitties в кінці 2017 року. У цій грі користувачі могли купувати, збирати, розводити та продавати віртуальних котів, кожен з яких був унікальним NFT. CryptoKitties стали настільки популярними, що в певний момент спричинили значне навантаження на мережу Ethereum.

Між 2020 та 2021 роками ринок NFT зазнав значного зростання: оборот торгівлі NFT у 2021 році сягнув понад 17 мільярдів доларів, що перевищило показники попереднього року на 21 000%, коли цей показник становив лише 82 мільйони доларів. NFT були популярними як спекулятивні інвестиції, проте викликали збільшену критику через енергетичні витрати та вуглецеві викиди, асоційовані з верифікацією блокчейн транзакцій, а також через їхнє використання в арт-шахрайствах.

У травні 2022 року The Wall Street Journal заявив, що ринок NFT "зазнав падіння". Щоденні продажі NFT знизились на 92% з вересня 2021 року, а кількість активних гаманців на ринку NFT скоротилася на 88% з листопада 2021 року. Зростання процентних ставок негативно вплинуло на спекулятивні інвестиції на фінансових ринках, причому NFT виявилися серед найбільш ризикованих активів.[5]

1.2.2 Поява dNFT

Перші динамічні NFT з'явилися в 2018 році, незабаром після буму CryptoKitties. Ці ранні dNFT були в основному простими експериментами, які використовували смартконтракти для зміни метаданих NFT у відповідь на прості події, такі як зміна часу або кількість транзакцій.

Приклади перших популярних dNFT:

- CryptoCelebrities: цей проект дозволяє користувачам купувати NFT з зображеннями відомих людей. Атрибути NFT, такі як вік та популярність зображеної людини, автоматично оновлювались з часом;

- Dinoverse: ця гра на основі блокчейну дозволяє користувачам розводити та збирати віртуальних динозаврів. Динозаври dNFT могли еволюціонувати та розвивати нові характеристики залежно від їхнього середовища та поведінки;

- Ether Cards: цей проект створив колекційні карткові ігри на основі блокчейну з dNFT, які могли змінити свої характеристики залежно від результатів ігрових матчів.

У 2021 році популярність dNFT почала стрімко зростати. Це було обумовлено декількома факторами:

- популяризація DeFi: децентралізовані фінанси стали популярною сферою криптовалютного ринку, і dNFT почали використовуватися для створення децентралізованих фінансових інструментів, таких як позики та деривативи;

- нові інструменти для створення: з'явилися нові інструменти, такі як Chainlink, котрі полегшили dNFT доступ до зовнішніх даних, що дозволило їм реагувати на складніші події, ніж просто зміна часу, транзакцій чи чогось подібного;

- розвиток геймінгу: dNFT почали використовуватися в іграх на основі блокчейну, що заохотило більшу кількість користувачів, оскільки багато хто знаходить в цьому можливість заробітку;

- метавсесвіти: dNFT стали масивною часткою метавсесвітів, оскільки вони можуть описувати активи користувача та динамічно змінювати свої властивості, як активи в реальному всесвіті.

1.2.3 Проєкти з використанням dNFT

Наразі є безліч проєктів з використанням dNFT найрізноманітніших сферах, починаючи з онлайн ігор, закінчуючи сертифікацією володіння реальними предметами.

Ігри:

- Axie Infinity: у цій грі, dNFT представлені у формі "Axies" — міфічних істот, які гравці можуть тренувати, розводити та брати участь у боях. Відмінність dNFT в Axie Infinity полягає в тому, що характеристики і навички цих істот можуть змінюватись відповідно до їхнього досвіду та взаємодій з іншими гравцями.

- Sorare: ця фентезі-футбольна гра на основі блокчейну дозволяє користувачам збирати, торгувати та грати з цифровими картками гравців, які є dNFT. Картки Sorare мають цінність, залежно від реальних виступів гравців на полі.

- Gods Unchained: ця карткова гра на основі блокчейну дозволяє користувачам володіти, торгувати та грати з унікальними картками, які є dNFT. Картки Gods Unchained можуть мати різну рідкісність та силу, що робить їх цінними колекційними предметами. Маючи певний набір карток, можна створити свою унікальну колоду для свого стилю гри.

Метавсесвіти:

- Decentraland: цей децентралізований метавсесвіт дозволяє користувачам володіти, розробляти та монетизувати віртуальну землю та активи. Земля та активи в Decentraland – це dNFT, які можуть мати різну цінність залежно від їх розташування, рідкості та корисності і відповідно змінювати свої характеристики з часом;

- The Sandbox: має ідентичний функціонал. Багато світових провідних компаній вже зробили інвестиції і купили великі віртуальні ділянки

Рисунок 1.2 – Мапа куплених ділянок на The Sandbox

- Cryptovoxels: має ідентичний функціонал.

Онлайн платформи для взаємодії dNFT:

- Rarible та OpenSea: дозволяють користувачам купувати, продавати та створювати dNFT. Підтримують широкий спектр dNFT, включаючи мистецтво, колекційні предмети, ігрові активи та віртуальну землю;

- Foundation: спеціалізується на високоякісному цифровому мистецтві. Foundation підтримує лише унікальні та рідкісні dNFT, які створюються відомими художниками;

- Art Blocks: платформа для створення та торгівлі генеративним мистецтвом у формі dNFT. Кожен арт-об'єкт унікальний і створюється динамічно на основі алгоритму, заданого художником. У 2024 році Art Blocks запровадили нову функцію, що дозволяє власникам впливати на остаточний

вигляд арт-об'єкту, змінюючи вихідні параметри генерації, що робить кожен твір ще більш персоналізованим і динамічним.

Реальні активи:

- RealT: Цей проєкт використовує dNFT для токенізації реальної нерухомості. Кожен dNFT представляє частку власності в нерухомості, що дозволяє інвесторам володіти та торгувати нерухомістю без необхідності володіти фізичним активом. RealT використовує смартконтракти для забезпечення безпеки та прозорості володіння dNFT. Смартконтракти автоматизують процес передачі власності та гарантують, що кожен dNFT представляє чітко визначену частку власності. RealT має партнерські відносини з декількома компаніями з нерухомості, що дозволяє їм пропонувати dNFT для широкого спектру нерухомості;
- Civic: проєкт використовує dNFT для створення децентралізованої системи ідентифікації особистості. Кожен dNFT представляє цифрову ідентичність користувача, яка може використовуватися для доступу до послуг та ресурсів. Civic має партнерські відносини з декількома компаніями, які надають послуги, що вимагають ідентифікації особистості, такі як банки та фінтех-компанії.

Але проєкти RealT та Civic поки що перебувають на ранній стадії розвитку і тому є експериментальними.

1.3 Технологічні основи динамічних NFT

Стандарти токенів (ERC) формують детальний набір правил, який описує процес створення токенів та їхньої поведінки у системі блокчейну. Зокрема, ERC 721 є стандартом для NFT, що дозволяє кожному токену бути унікальним та мати різну вартість порівняно з іншими токенами в одному смарт-контракті, наприклад через такі характеристики як вік, рідкість або візуальні особливості.

ERC 721 забезпечує основні можливості для відстеження та передавання NFT. Стандарт ERC-1155 розширює можливості ERC-721, додаючи функціональність для обробки множинних токенів, таких як збереження даних про численні предмети, атомарні обміни, торгівлю групами елементів та одночасні передачі і комбінації кількох елементів у одній транзакції. Ідентифікатор маркера ERC 721 є унікальним незмінним індексом, прив'язаним до єдиного контракту з налаштуваннями, які застосовуються до всіх маркерів у колекції. На відміну від нього, стандарт ERC 1155 дозволяє кожному ідентифікатору маркера представляти новий настроюваний тип маркера з власними користувацькими метаданими, постачанням та іншими властивостями. ERC 1155 також сприяє збереженню кількох елементів в одному смарт-контракті, зменшуючи необхідний обсяг даних для ідентифікації різних токенів.

Щоб розробити динамічний NFT, програмісти смарт-контрактів мають врахувати певні обмеження та функції, які дозволяють змінювати метадані NFT в залежності від конкретних обставин. Смарт-контракти можуть збирати дані з зовнішніх джерел та оновлювати метадані на основі нової інформації. Завдяки цим контрактам, можна оновлювати, змінювати і розширювати NFT з часом. Вони також визначають, чи потрібно змінювати незамінний токен на основі внутрішніх та зовнішніх даних. Динамічні NFT надають змогу зберігати унікальні ідентифікатори, а також дозволяють оновлення метаданих. Програмні алгоритми автоматично змінюють метадані і властивості елементів за заданих умов. NFT включає постійні елементи, які не можуть бути змінені, такі як ідентифікатор токена і адреса контракту, які зберігаються у блокчейні.

Метадані NFT описують основні характеристики, включаючи назву, опис та інші важливі властивості. Криптографічно захищений ідентифікатор ресурсу (URI), який включено до NFT, вказує на місцезнаходження метаданих. URI може посилатися на веб-сайт, підтримуваний творцем NFT, вторинний сервіс, адресу IPFS або інше сховище даних. Метадані часто включають посилання на

зображення та інші цифрові активи, які надають вартість NFT. Вони можуть бути «оновлені» за допомогою функції _setTokenURI(). Метадані динамічного NFT можуть змінюватися на основі часу, взаємодії з іншими NFT та зовнішніх даних, отриманих з джерел, які відслідковують реальні дані, такі як вітряні флюгери, результати спортивних змагань або виборів.

Як уже згадувалося, автоматичні зміни в динамічному NFT вбудовані в його смарт-контракт. Розробник має закодувати ці зміни. Оскільки багато динамічних NFT залежать від змін у реальному світі для оновлення метаданих, використання рішень оракулу буде ефективним для створення таких dNFT. Оракули надають позамережні або обчислювальні послуги для введення даних, які ініціюють оновлення, передбачені в смарт-контрактах, тому точність даних оракулів є критичною. Компрометація оракулів може загрожувати всім протоколам та смарт-контрактам у блокчейн-мережах, з якими вони взаємодіють. [3]

1.3.1 Процес оновлення метаданих

1. Отримання URI зі смарт-контракту:

На цьому етапі смарт-контракт використовується для вилучення універсального ідентифікатора ресурсів (URI), який зазвичай вказує на місцезнаходження метаданих NFT.

2. Ініціація функції оновлення метаданих:

Функція у смарт-контракті активується для початку процесу оновлення метаданих, готуючи систему до прийняття нових метаданих.

3. Завантаження нових метаданих виконавцем та отримання нового ID контенту:

Спеціально призначена особа або автоматизована система завантажує оновлені метадані до сховища даних. Після завантаження метаданих

генерується новий ідентифікатор контенту, який буде використовуватись для їх ідентифікації.

4. Той самий виконавець з високим рівнем дозволів замінює старий JSON метаданих на новий:

Той самий виконавець, який має високий рівень доступу, замінює старий файл JSON, який містить метадані, на новий файл з оновленою інформацією.

5. Опціонально: може бути ініційована нова подія для оновлення історії змін для публічних цілей:

Як опція, може бути запущена подія, яка реєструє історію оновлень метаданих, що може бути використано для звітності або публічного доступу до історії змін метаданих.

1.3.2 Стандарти динамічних NFT

Застосування стандартів, зокрема таких як ERC (Ethereum Request for Comments), є вирішальним для інтеграції, сумісності, безпеки та ширшої адаптації dNFT. Ці стандарти не лише сприяють технічній стабільності та надійності токенів, але й відіграють важливу роль у визначенні того, яким чином метадані взаємодіють з різними блокчейнами та додатками.

Використовуючи стандартизацію, ми можемо і спростити роботу собі, як розробнику, оскільки не придумуємо функціонал з нуля, а наслідуємо та модернізуємо заздалегідь реалізований, так і зробити кращу інтеграцію в спільноту блокчейну. Таким чином з нашим контрактом зможуть взаємодіяти стороні сервіси без додаткової інтеграції, оскільки ми гарантовано маємо функціонал певного ERC.

За рахунок активного розвитку спільноти впродовж останніх років і залучення великих інвестицій в сферу блокчейну йде велика кількість пропозицій реалізації певного функціоналу або потреб, тому з’являється все

більше нових стандартів, що дозволяє під час створення власного продукту або блокчейн додатку знайти саме той, котрий найкраще підійде під потреби.

Отже давайте розглянемо можливі варіанти стандартів для створення власних динамічних NFT.

1.3.2.1 ERC-1155

ERC-1155 – це стандарт токенів на блокчейні Ethereum, який розширює можливості ERC-721, стандартного формату для NFT. На відміну від ERC-721, який дозволяє створювати лише унікальні, невзаємозамінні токени (NFT), ERC-1155 дозволяє створювати як невзаємозамінні, так і взаємозамінні токени в одному контракті. Це робить його більш універсальним стандартом для створення широкого спектру активів на блокчейні.

ERC-1155 був створений для вирішення деяких обмежень ERC-721. ERC-721 добре підходить для створення унікальних NFT, таких як цифрові твори мистецтва або колекційні предмети. Однак він не є гнучким для створення інших типів активів, таких як ігрові активи, віртуальні валюти або квитки на події.

ERC-1155 вирішує цю проблему, дозволяючи створювати токени, які можуть бути як унікальними, так і взаємозамінними. Це робить його більш універсальним стандартом для створення широкого спектру активів на блокчейні.

ERC-1155 має ряд функцій, які роблять його більш універсальним, ніж ERC-721. До цього функціоналу належать:

- підтримка як невзаємозамінних, так і взаємозамінних токенів: ERC-1155 дозволяє створювати токени, які можуть бути як унікальними, так і взаємозамінними. Це робить його більш гнучким для створення широкого спектру активів на блокчейні;

- підтримка кількох типів активів: ERC-1155 дозволяє створювати токени, які представляють кілька типів активів в одному контракті. Це робить його більш ефективним для створення складних активів, таких як ігрові активи або віртуальні світи;

- підтримка пакетних транзакцій: ERC-1155 дозволяє переміщати кілька токенів різного типу в одній транзакції. Це робить його більш ефективним для переміщення великої кількості активів;

- підтримка оновлення метаданих: ERC-1155 дозволяє оновлювати метадані токенів після їх створення. Це робить його більш корисним для створення динамічних активів, які змінюються з часом.

ERC-1155 надає функцію _setURI(), яка дозволяє власнику контракту оновлювати URI (Uniform Resource Identifier) токена. URI – це посилання на місце розташування метаданих токена. Оновлюючи URI, власник контракту може змінити інформацію, що міститься в метаданих, таких як опис токена, його властивості або зображення.

Основні функції:

- balanceOf(address owner, uint256 id): повертає кількість токенів типу id, якими володіє адреса owner;

- balanceOfBatch(address[] owners, uint256[] ids): повертає баланси декількох адрес для декількох типів токенів;

- uri(uint256 id): повертає URI для метаданих токена типу id;

- setURI(uint256 id, string uri): встановлює URI для метаданих токена типу id;

- safeTransferFrom(address from, address to, uint256 id, uint256 amount, bytes data): передає amount токенів типу id з адреси from на адресу to;

- safeBatchTransferFrom(address from, address to, uint256[] ids, uint256[] amounts, bytes data): передає декілька типів токенів з адреси from на адресу to в партіях. [10]

Але при цьому важливо зазначити, що не всі токени, котрі реалізовують ERC-1155, мають можливість оновлення метаданих, оскільки це залежить від реалізації контракту розробниками.

1.3.2.2 ERC-721TL

ERC-721TL – це стандарт токенів на блокчейні Ethereum, який розширює можливості ERC-721, стандартного формату для NFT. ERC-721TL розроблений компанією Transient Labs і пропонує ряд нових функцій, які не доступні в ERC-721.

Основні функції ERC-721TL:

- підтримка оновлюваних метаданих: ERC-721TL дозволяє оновлювати метадані NFT після їх створення. Це дозволяє підтримувати концепцію динамічних NFT;
- підтримка пакетних транзакцій: ERC-721TL дозволяє переміщати кілька NFT в одній транзакції. Це робить його більш ефективним та дешевим для переміщення великої кількості NFT;
- підтримка роялті: ERC-721TL дозволяє творцям NFT стягувати роялті з кожного продажу NFT. Це може допомогти творцям NFT отримувати стійкий дохід від своїх творів.

Переваги використання ERC-721TL:

- динамічність: ERC-721TL дозволяє створювати NFT, які можуть змінюватися з часом. Це відкриває нові можливості для використання NFT, таких як створення ігрових активів, які еволюціонують з часом, або NFT, які представляють реальні активи, які змінюються в цінності;
- більш ефективні транзакції: ERC-721TL дозволяє переміщати кілька NFT в одній транзакції. Це робить його більш ефективним для переміщення великої кількості NFT;

- підтримка творців: ERC-721TL дозволяє творцям NFT стягувати роялті з кожного продажу NFT. Це може допомогти творцям NFT отримувати стійкий дохід від своїх творів.

ERC-721TL все ще є новим стандартом, і він не так широко використовується, як ERC-721. Однак він пропонує ряд гарних нових функцій, які роблять його більш привабливим для творців NFT та колекціонерів. [6]

1.3.2.3 ERC-6551

ERC-6551 – це стандарт токенів на блокчейні Ethereum, який розширює можливості ERC-721, стандартного формату для NFT. ERC-6551 спеціально розроблений для динамічних NFT, які можуть змінюватися та еволюціонувати з часом.

Основні функції ERC-6551:

- підтримка оновлення атрибутів: ERC-6551 дозволяє оновлювати атрибути NFT після їх створення. Це робить його більш корисним для створення динамічних NFT, які можуть змінюватися з часом;

- здатність створювати NFT з обмеженим терміном дії: ERC-6551 дозволяє створювати NFT з обмеженим терміном дії. Це може бути корисно для створення NFT, які представляють квитки на події або інші активи з обмеженим терміном дії;

- можливість встановлювати залежності між NFT: ERC-6551 дозволяє встановлювати залежності між NFT. Це може бути корисно для створення складних ігрових активів або інших активів, які взаємодіють один з одним.

Переваги використання ERC-6551:

- більш гнучкі NFT: ERC-6551 дозволяє створювати NFT з обмеженим терміном дії та встановлювати залежності між NFT. Це робить його більш гнучким стандартом для створення широкого спектру активів;

- більш універсальні NFT: ERC-6551 може використовуватися для створення динамічних NFT, які неможливо створити за допомогою ERC-721.

ERC-6551 все ще є новим стандартом, і він не так широко використовується, як ERC-1155. Однак він має ряд удосконалень, котрі можуть слугувати причиною вибору цього стандарту.

1.3.2.4 ERC-2981

ERC-2981 – це пропозиція стандарту токена Ethereum, яка прагне вирішити проблему роялті для невзаємозамінних токенів. Коли NFT продається на вторинному ринку, первинний творець зазвичай не отримує роялті від цього продажу. ERC-2981 пропонує стандартний спосіб для NFT-проектів вказувати роялті, які повинні виплачуватися творцям при перепродажу їхніх NFT.

ERC-2981 визначає два методи для вказівки роялті:

- за замовчуванням роялті: Цей метод встановлює єдину ставку роялті, яка застосовується до всіх NFT, випущених проектом;

- роялті на рівні токена: Цей метод дозволяє проектам вказувати різні ставки роялті для різних NFT.

В обох випадках ставка роялті виражається як відсоток від ціни продажу. Наприклад, ставка роялті 10% означає, що творець отримає 10% від кожного продажу свого NFT.

Основні характеристики ERC-2981:

- сумісність з ERC-721 та ERC-1155: ERC-2981 сумісний з існуючими стандартами NFT, ERC-721 та ERC-1155, що робить його простим для впровадження в існуючі NFT-проекти;

- стандартизована структура роялті: ERC-2981 визначає стандартизовану структуру даних для роялті, що робить інформацію про роялті легкодоступною та зрозумілою;

- підтримка декількох творців: ERC-2981 дозволяє вказувати роялті для декількох творців, що робить його придатним для спільних проектів;

- розширення: ERC-2981 можна розширити за допомогою додаткових функцій, таких як динамічні ставки роялті або підтримка різних типів розподілу роялті.

Функції ERC-2981:

- royaltyInfo(uint256 tokenId, address sender): повертає структуру даних з інформацією про роялті для NFT з tokenId, що використовується при продажу з адреси sender;

- setRoyaltyInfo(uint256 tokenId, address receiver, uint96 royaltyFraction): встановлює інформацію про роялті для NFT з tokenId, де receiver – це адреса, яка отримуватиме роялті, а royaltyFraction – це відсоток від ціни продажу, який буде виплачуватися.

ERC-2981 має кілька переваг, зокрема:

- справедливість для творців: ERC-2981 допомагає забезпечити те, щоб творці NFT отримували справедливу компенсацію за свою роботу, навіть після того, як їхні NFT були продані;

- прозорість: ERC-2981 робить інформацію про роялті доступною для всіх, що полегшує покупцям NFT розуміння того, скільки вони платять творцям;

- ефективність: ERC-2981 пропонує стандартизований спосіб вказівки роялті, що полегшує ринкам NFT підтримку виплати роялті.

ERC-2981 все ще є відносно новим стандартом, і його ще не широко підтримують. Однак зростає кількість NFT-проектів, які впроваджують цей стандарт. Очікується, що ERC-2981 відіграватиме все більшу роль у забезпеченні справедливої компенсації творцям NFT у міру зростання популярності NFT. [7]

1.3.2.5 ERC-4907

ERC-4907 – це пропозиція стандарту токена Ethereum, яка розширює функціональність NFT. Стандартні NFT зазвичай містять лише статичну інформацію, таку як ідентифікатор та посилання на мультимедійний контент. Це робить його корисною основою для створення динамічних NFT, які можуть реагувати на зовнішні події або дії користувачів.

Основні характеристики ERC-4907:

- сумісність з ERC-721: ERC-4907 ґрунтується на стандарті ERC-721, що робить його сумісним з існуючими інструментами та платформами NFT.

- нові ролі: ERC-4907 вводить дві нові ролі: "власник" та "користувач"; Власник NFT може надати користувачеві дозвіл на використання NFT протягом певного періоду часу;

- оренда NFT: ERC-4907 дозволяє власникам NFT орендувати їх користувачам на певний час. Користувачі мають доступ до NFT протягом цього періоду, але не володіють ним;

- розширення: ERC-4907 можна розширити за допомогою додаткових функцій, таких як контроль доступу, роялті або динамічні умови оренди.

Функції ERC-4907:

- userOf(uint256 tokenId): Повертає адресу користувача, який має доступ до NFT з tokenId;

- approveUser(uint256 tokenId, address user): надає користувачеві user дозвіл на використання NFT з tokenId;

- setUser(uint256 tokenId, address user): встановлює користувача user для NFT з tokenId;

- getEndTime(uint256 tokenId): повертає час закінчення оренди NFT з tokenId;

- setEndTime(uint256 tokenId, uint256 endTime): встановлює час закінчення оренди для NFT з tokenId.

ERC-4907 відкриває двері для створення різних типів динамічних NFT, зокрема:

- NFT з ігровими елементами: дані, що зберігаються в NFT, можуть представляти статистику персонажа, рівень розвитку або інші ігрові атрибути, які можуть змінюватися під час гри;

- NFT з прив'язкою до реального світу: дані NFT можуть містити інформацію про фізичний актив, що він представляє, наприклад, дані датчиків або інформацію про місцезнаходження. Ці дані можуть оновлюватися в режимі реального часу;

- NFT з прогресивним контентом: дані NFT можуть використовуватися для керування доступом до розблоковуваного контенту. Наприклад, NFT може надати доступ до певного рівня відеогри лише після того, як користувач досягне певного ігрового етапу, що відображається в даних NFT.

Переваги:

- дозволяє створювати більш інтерактивні та захоплюючі NFT;

- сприяє розвитку нових сфер застосування NFT за межами цифрових колекційних предметів;

- забезпечує більшу гнучкість для розробників NFT.

Обмеження:

- обмежена ємність сховища даних (32 байти);

- необхідність додаткової розробки для використання динамічних можливостей;

- стандарт все ще перебуває на ранній стадії розвитку.

2. РЕАЛІЗАЦІЯ ВЛАСНОЇ ПРОПОЗИЦІЇ СТАНДАРТУ ТА СМАРТКОНТРАКТУ НА ЙОГО ОСНОВІ

Стандарти виникають в тих випадках, коли у людства є якась потреба, котра часто повторюється, наприклад, починаючи з команд в армії, щоб поведінка кожного солдата була схожою та очікуваною, закінчуючи типізацією отворів кріплення модулів на МКС, щоб кожна країна виробник нового модуля могла легко інтегруватися в середовище космічної станції.

Власне схоже відбулося в блокчейні, щоб різні “виробники”, котрі реалізовують схожий функціонал своїх контрактів, мали спільні точки, щоб іншим користувачам платформи легше було взаємодіяти з цими контрактами і вони знали, що в конкретному контракті є цей функціонал.

Початково це здається схожим на патерн стратегії, але так вважати – це велика помилка, оскільки в даному патерні взаємодія з об’єктом відбувається тільки за визначеними функціями через інтерфейс. А у випадку блокчейну Ethereum стандартом вважається набором мінімально потрібного функціоналу. Тобто, якщо хоч одна функція, котра визначена в стандарті, не буде присутня в реалізованому контракті, то, відповідно, і не можна сказати, що ця реалізація відповідає тому чи іншому erc. Але водночас можна доповнювати функціонал, котрий так само буде доступний для зовнішнього користувача, тому це надає можливість реалізовувати декілька стандаратів одночасно.

2.1 Опис ідеї

Можна придумати стандарт для багатьох речей, але це не матиме сенсу, якщо це одноразовий випадок застосування певного функціоналу, а не повторюваний багато разів.

За основу ідеї було взято ERC-1155, оскільки цей напрям NFT доволі росте через популяризацію метавсесвітів, ігор тощо, але при цьому включити в нього філософію динамічних невзаємозамінних токенів.

На мою думку, ніщо краще не закріплюється в віртуальному світі, як те, чому можна знайти аналогію в реальному. Тому після аналізу різних дій, котрі люди часто роблять повсякденно, було виокремлено те, що зараз є частиною нашої буденності.

Це онлайн маркетплейси. Для багатьох купівля в інтернеті стала більш звичною ніж купівля в фізичних магазинах, тому я вирішив інтегрувати базові характеристики предметів в маркетплесах в характеристики токенів. Отже, як на мене, для предмету, який продається, основними характеристиками є:

- чи доступний цей предмет зараз для продажу;
- вартість купівлі цього предмету;
- скільки можна продати цього предмету.

Якщо у реальному світі з часом у предмету, який продається, можуть змінитися властивості,то, відповідно, і предмет, який проданий у цифровому світі, може змінювати свої властивості (наприклад, ламатися, модернізуватися тощо). При цьому, як і в реальному світі, можна обмінювати один предмет на інший в якійсь пропорції. Тому враховуючи таку поведінку, пропонується створити такий функціонал в інтерфейсі:

- дізнатися кількість токенів token_id у користувача account;
- дізнатися кількість токенів [token_id] у користувачів [account];
- надати/забрати можливість розпоряджатися токенами адресу operator;
- дізнатися чи має право адрес operator розпоряджатися токенами;
- перевести токени token_id з користувача from до користувача to в кількості amount;
- перевести токени [token_id] з користувача [from] до користувача [to] в кількості [amount];

- встановити базовий uri;

- дізнатися базовий uri;

- встановити кастомний uri для token_id;

- встановити кастомний uri для [token_id];

- прибрати кастомний uri для token_id;

- прибрати кастомний uri для [token_id];

- дізнатися uri для token_id;

- дізнатися uri для [token_id];

- встановити обмеження кількості випуску токенів для token_id;

- встановити обмеження кількості випуску токенів для [token_id];

- прибрати обмеження кількості випуску токенів для token_id;

- прибрати обмеження кількості випуску токенів для [token_id];

- дізнатися обмеження кількості випуску токенів для token_id;

- дізнатися обмеження кількості випуску токенів для [token_id];

- встановити заборону(bool) для купівлі токена token_id;

- встановити заборону(bool) для купівлі токена [token_id];

- дізнатися про заборону купівля токена token_id;

- дізнатися про заборону купівля токена [token_id];

- встановити ціну купівлі токену по token_id, price (число з плаваючою точкою);

- встановити ціну купівлі токену по [token_id], [price];

- прибрати ціну купівлі токену по token_id;

- прибрати ціну купівлі токену по [token_id];

- дізнатися ціну купівлі токену по token_id;

- купити токену по token_id;

- замінити токен з token_id в кількості amount на токен new_token_id в кількості new_amount;

- випустити токени з ціною, дозволом на купівлю та обмеженням в кількості;

- випустити токени з ціною, дозволом на купівлю та обмеженням в кількості;

дізнатися кількість випущених токенів з token_id;

- дізнатися кількість випущених токенів з [token_id];

- вивести гроші з контракту.

Таким чином, цей стандарт дозволить реалізувати функціонал маркетплейсу з динамічною зміною метаданих.

2.2 Інтерфейси

В цій роботі пропонується наступний інтерфейс для даного опису функцій:

```
  interface IEIP2014 {
      // Events
      event TransferSingle(address indexed operator, address indexed from,
address indexed to, uint id, uint value);
      event TransferBatch(address indexed operator, address indexed from,
address indexed to, uint[] ids, uint[] values);
      event ApprovalForAll(address indexed account, address indexed
operator, bool approved);
      event Issued(uint indexed tokenId, uint price, bool
purchasePermission, uint limit, string uri);
      event URI(string value, uint indexed id);

      // Token balances and approvals
      function balanceOf(address account, uint id) external view
returns(uint);
      function balanceOfBatch(address[] calldata accounts, uint[] calldata
ids) external view returns(uint[] memory);
      function setApprovalForAll(address operator, bool approved) external;
      function isApprovedForAll(address account, address operator) external
view returns(bool);
```

```
        // Token transfers
        function safeTransferFrom(address from, address to, uint id, uint
amount, bytes calldata data) external;
        function safeBatchTransferFrom(address from, address to, uint[]
calldata ids, uint[] calldata amounts, bytes calldata data) external;

        // URI management
        function setBaseURI(string calldata uri) external;
        function getBaseURI() external view returns(string memory);
        function setCustomURI(uint id, string calldata uri) external;
        function setCustomURIBatch(uint[] calldata ids, string[] calldata
uris) external;
        function removeCustomURI(uint id) external;
        function removeCustomURIBatch(uint[] calldata ids) external;
        function getURI(uint id) external view returns(string memory);
        function getURIBatch(uint[] calldata ids) external view
returns(string[] memory);

        // Token issuance limits
        function setTokenIssuanceLimit(uint id, uint limit) external;
        function setTokenIssuanceLimitBatch(uint[] calldata ids, uint[]
calldata limits) external;
        function removeTokenIssuanceLimit(uint id) external;
        function removeTokenIssuanceLimitBatch(uint[] calldata ids) external;
        function getTokenIssuanceLimit(uint id) external view returns(uint);
        function getTokenIssuanceLimitBatch(uint[] calldata ids) external view
returns(uint[] memory);

        // Purchase permissions
        function setPurchasePermit(uint id, bool permitted) external;
        function setPurchasePermitBatch(uint[] calldata ids, bool[] calldata
prohibitions) external;
        function getPurchasePermission(uint id) external view returns(bool);
        function getPurchasePermissionBatch(uint[] calldata ids) external view
returns(bool[] memory);

        // Token purchase price management
        function setPurchasePrice(uint id, uint price) external;
        function setPurchasePriceBatch(uint[] calldata ids, uint[] calldata
prices) external;
        function removePurchasePrice(uint id) external;
        function removePurchasePriceBatch(uint[] calldata ids) external;
        function getPurchasePrice(uint id) external view returns(uint);
```

```
        function getPurchasePriceBatch(uint[] calldata ids) external view
returns(uint[] memory);

        // Token transactions
        function purchaseToken(uint id) external payable;
        function upgradeToken(uint tokenId, uint newTokenId, uint amount, uint
newAmount, address account) external;
        function upgradeTokenBatch(uint[] memory tokenIds, uint[] memory
newTokenIds, uint[] memory amounts, uint[] memory newAmounts, address[]
memory accounts) external;
        function issueToken(uint price, bool purchasePermission, uint limit,
string calldata uri) external returns (uint);
        function issueTokenBatch(uint[] memory prices, bool[] memory
purchasePermissions, uint[] memory limits, string[] memory uris) external
returns (uint[] memory);
        function issueTokenCount(uint id) external view returns (uint);
        function issueTokenCountBatch(uint[] memory ids) external view returns
(uint[] memory);

        // Financial management
        function withdrawFunds() external;
    }
```

Також для безпечного запису переводу токенів з адреси на адресу нам потрібно робити перевірку, чи адреса отримувача є контрактом, і якщо так, то вона повинна імплементувати даний інтерфейс:

```
    interface IEIP2014Receiver {
        function onERC1155Received(
            address operator,
            address from,
            uint id,
            uint amount,
            bytes calldata data
        ) external returns(bytes4);

        function onERC2014BatchReceived(
            address operator,
            address from,
            uint[] calldata ids,
            uint[] calldata amounts,
            bytes calldata data
```

```
    ) external returns(bytes4);
}
```

2.3 Екземпляр контракту EIP-2014

Екземпляр контракту EIP-2014 буде використовувати інтерфейс IEIP-2014, щоб відповідати потрібному стандарту. Для початку розглянемо конструктор та загальні змінні контракту:

```
contract EIP2014 is IEIP2014{
    uint private _tokenCounter = 0;
    uint private _balance;
    mapping(uint => mapping(address => uint)) private _balances;
    mapping(address => mapping(address => bool)) private
_operatorApprovals;

    string private _baseURI;
    mapping(uint => string) private _tokenURIs;

    uint[] private _issuedTokens;
    mapping(uint => uint) private _countTokens;
    mapping(uint => uint) private _limits;
    mapping(uint => uint) private _purchasePrices;
    mapping(uint => bool) private _purchasePermissions;

    address private _owner;

    constructor(address owner) {
        _owner = owner;
        _balance = address(this).balance;
    }
```

Оскільки наш контракт є умовним маркетплейсом, то він буде накопичувати кошти і в кінці ми повинні мати право вивести їх, тому ми приймаємо адресу owner власника цього контракта задля майбутніх функцій, котрі може визивати власник і на цю адресу будуть виводитися накопичені кошти.

Створюється лічильник токенів _tokenCounter, щоб ми могли автоматично призначати новий індекс при створенні токену, а не пам’ятати, які створенні і визначати вільними.

_balance є балансом нашого контракту в мережі.

_balances є ідентичним за функціоналом, як в ERC-1155, він зберігає привязку айді токену та кількість цих токенів на кожному гаманці.

_operatorApprovals є списком адрес, котрі можуть робити операції переказу токенів замість власника.

_baseURI - базова URI, котра повертається, якщо не встановлене індивідуальне URI для токена.

_tokenURIs - індивідуальні URI токенів.

_issuedTokens - список айді випущених токенів.

_countTokens - кількість випущених токенів для кожного токену.

_limits - обмеження кількості випуску токенів(якщо вказаний 0, або не вказаний - кількість може бути необмежанна)

_purchasePrices - ціна продажу токену відносно айді.

_purchasePermissions - чи продається цей токен на цей момент.

За допомогою цих полів ми можемо реалізувати повністю весь функціонал інтерфейсу IEIP-2014, приклад якого ви можете знайти на цьому репозиторії https://github.com/RiperYT/ERC-2014.

2.4 Створення проєкту на контракті EIP-2014

Більшість проблем, які відбулися з ринком NFT, були пов'язані з тим, що сам по собі токен не мав жодної цінності і це була наприклад просто картинка, а люди їх купували з метою збагачення, бо через хвилю хайпу була ідея того, що потім вони зможуть продати подорожче. Тому схема була аналогічна акціям “МММ” Сергія Мавродія котрий просто побудував піраміду.

Тому через цей досвід зараз люди набагато обережніше купують NFT і хочуть мати з цього якусь користь, як наприклад токен Bored Ape Yacht Club, котрий дає власникам доступ до приватних вечірок та заходів по всьому світу.

2.4.1 Опис

Ідея створення донат-платформи для України з'явилася через ідею того, що потрібно людині, котра захоче купити токен, та надання їй можливість бути причетною до токену, враховуючи відсутність можливості організовувати, наприклад, вечірки або виробляти продукцію, щоб відправляти покупцям.

Основна мета заохочення користувача буде в тому, що купуючи позицію (токен), він спочатку отримує його собі на баланс таким, який він продається, але з часом буде мігрувати в індивідуальний неповторний токен.

Наведу приклад. Нехай у нас продається: FPV дрон 7 дюймів, FPV дрон 10 дюймів, DJI Mavic 3. І наприклад, якщо людина купила (задонатила) дрон 7 дюймів, вона отримує відповідну NFT, яка з часом, після його виготовлення змінить метадані на ті, де вже буде картинка цього дрону з першими 5 цифрами контракту, з якого відбулася купівля. Також в метадані будуть додані атрибути, куди та в яку бригаду відправляється саме цей дрон. Таким чином ми створюємо індивідуальний досвід з користувачем і показуємо особисту причетність його донату до допомоги Україні, що і буде основним заохоченням. Також серед продажних тез можна зазначити, що покупцеві може випасти можливість запису військовими відео влучаня з цього дрону, яке також буде серед метаданих токену.

2.4.2 Створення контракту

Тепер модифікуємо контракт написаний нами, перед цим додавши два поля та одну функцію:

```
  pragma solidity ^0.8.0;

  import "./ERC2014.sol";

  contract UADonate is EIP2014{
      string public name;
      string public symbol;

      constructor(string memory Name, string memory Symbol)
ERC2014(msg.sender)
      {
          name = Name;
          symbol = Symbol;
      }

      function uri(
          uint _tokenId
      ) public view returns (string memory) {
          return this.getURI(_tokenId);
      }
  }
```

Після цього ми додатково задаємо ім’я та символ нашого токену, щоб вони могли відображатися на платформах по типу Opensea.

Потім додано функцію function uri(uint _tokenId) заради того, щоб відповідати формату ERC-1155, котрий підтримують більшість платформ для взаємодії/перегляду з NFT, і щоб люди могли побачити візуалізацію свого токену не тільки в нашому майбутньому додатку.

2.4.3 Деплой контракту на блокчейн

Після написання контракту потрібно його запушити в мережу блокчейну для майбутнього користування. Для цього буде використовуватися Remix IDE та Metamask.

Після того як ми завантажимо наш код в Remix IDE, необхідно зробити його компіляцію.

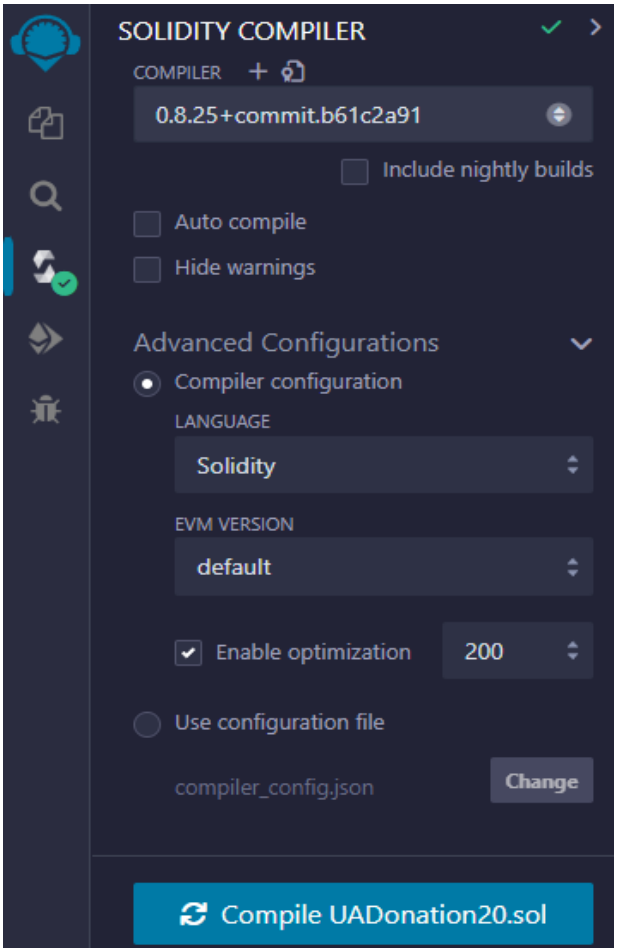

Рисунок 2.1 - Компіляція контракту в Remix IDE

Для цього вибираємо потрібну мову та компілятор, в даному випадку використовується Solidity 0.8.25. Також додатково ввімкнено оптимізацію задля зменшення вартості газу, котрий піде на деплой в мережу (але при цьому незначно виросте ціна майбутніх операцій).

Після того як наш код буде скомпільовано, можна приступати до власне деплою.

Спочатку треба вибрати мережу, в яку необхідно задеплоїти наш контракт. Для цього необхідно відкрити Metamask та перемкнути на потібну мережу:

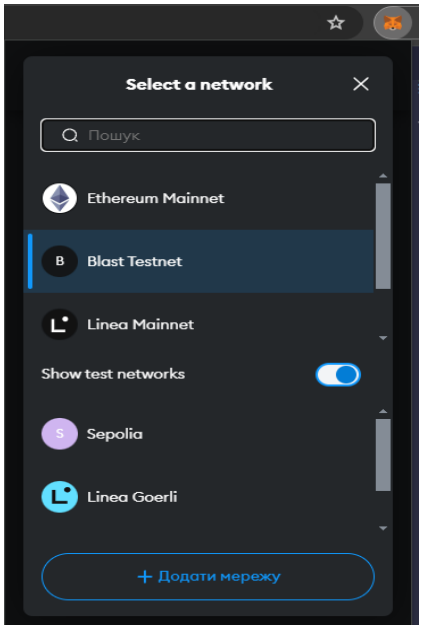

Рисунок 2.2 - Вибір мережі в Metamask

Тепер, перейшовши до вкладки Deploy в Remix IDE, можна ввести відповідні дані для нашого конструктора та задеплоїти наш контракт:

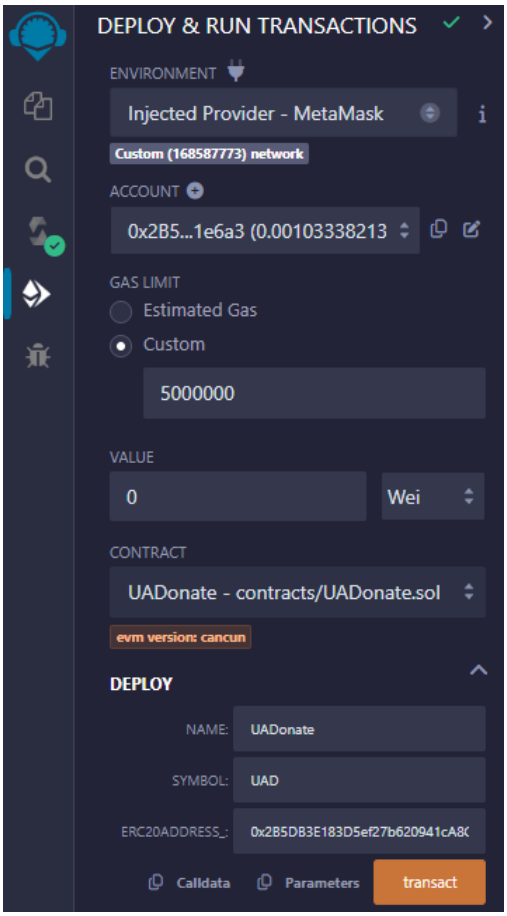

Рисунок 2.3 - Деплой контракту в Remix IDE

Після цього підтверджуємо нашу транзакцію в Metamask:

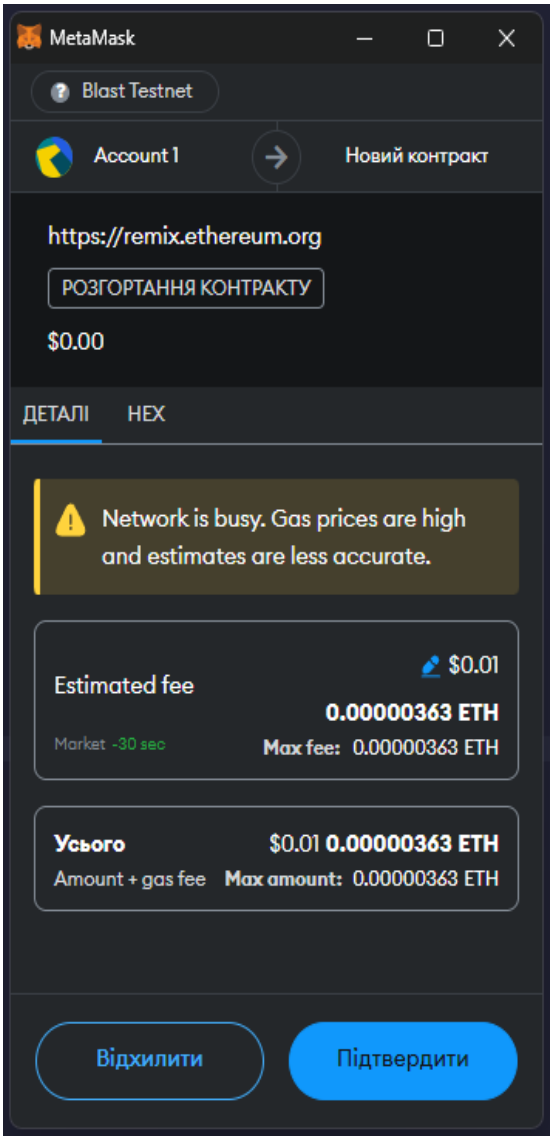

Рисунок 2.4 - Підтвердження транзакції деплою в Metamask

У результаті будь-хто може взаємодіяти з цим контрактом в мережі Blast Testnet за адресою 0xa21a94d4afCB9C06c49D4241DD49462a82FB1202.

Останнім кроком верифікуємо наш контракт:

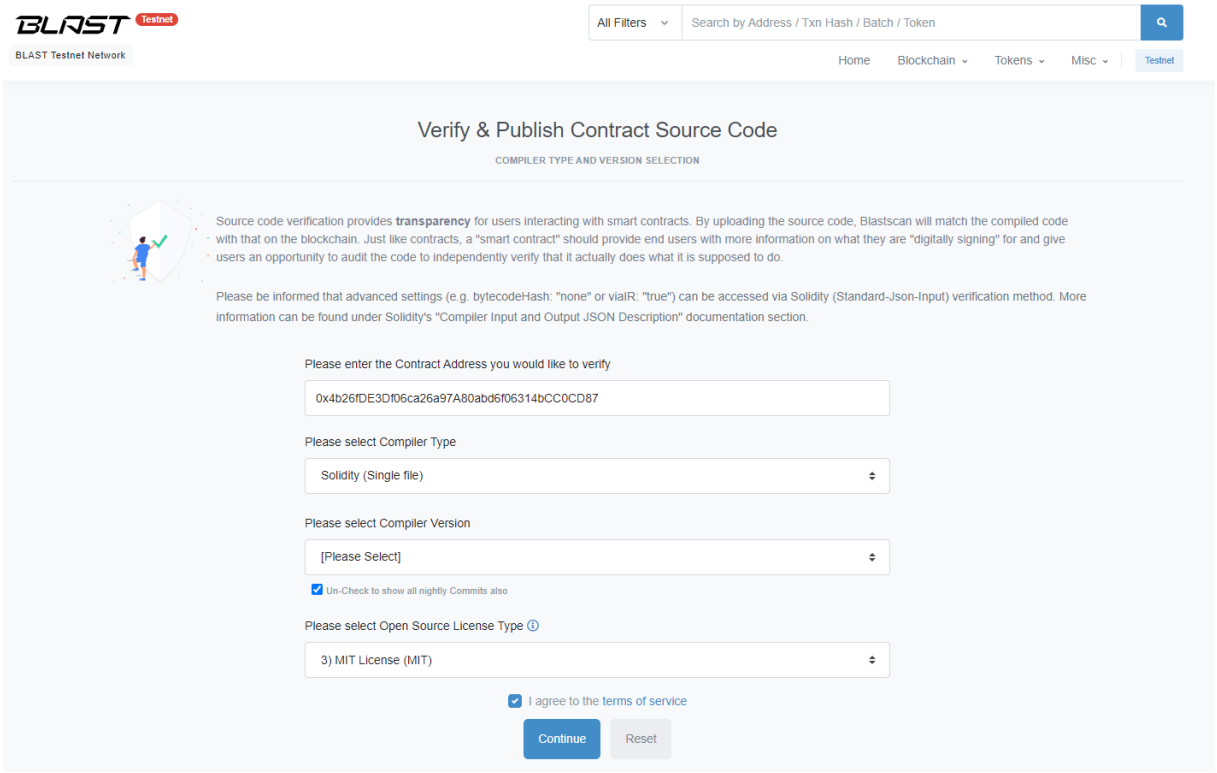

Рис 2.5 Верифікація контракту

З’являється можливість зручно працювати з контрактом через сканер:

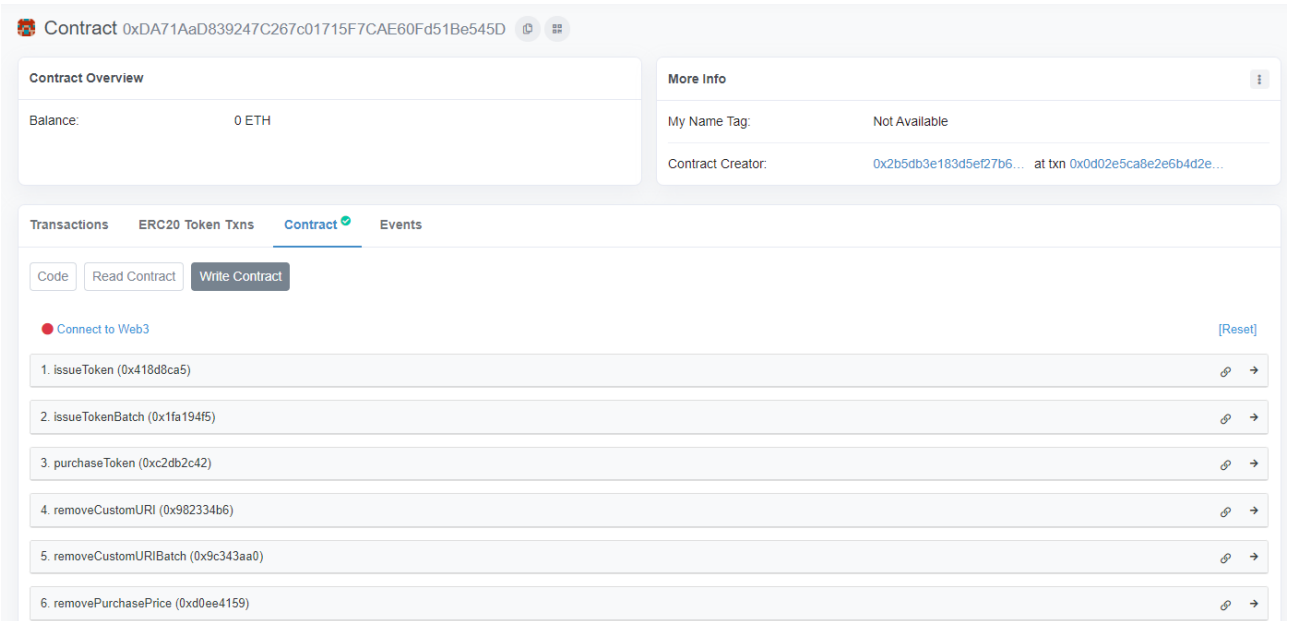

Рис. 2.6 Інтерфейс взаємодії з контрактом через сканер

2.4.4 Створення метаданих та випуск токену

Метадані будемо зберігати в децентралізованому сховищі задля їх безпечності та доступності. Для цього я буду використовувати платформу Filebase.

Першим кроком необхідно створити bucket, куди будуть завантажуватися наші файли:

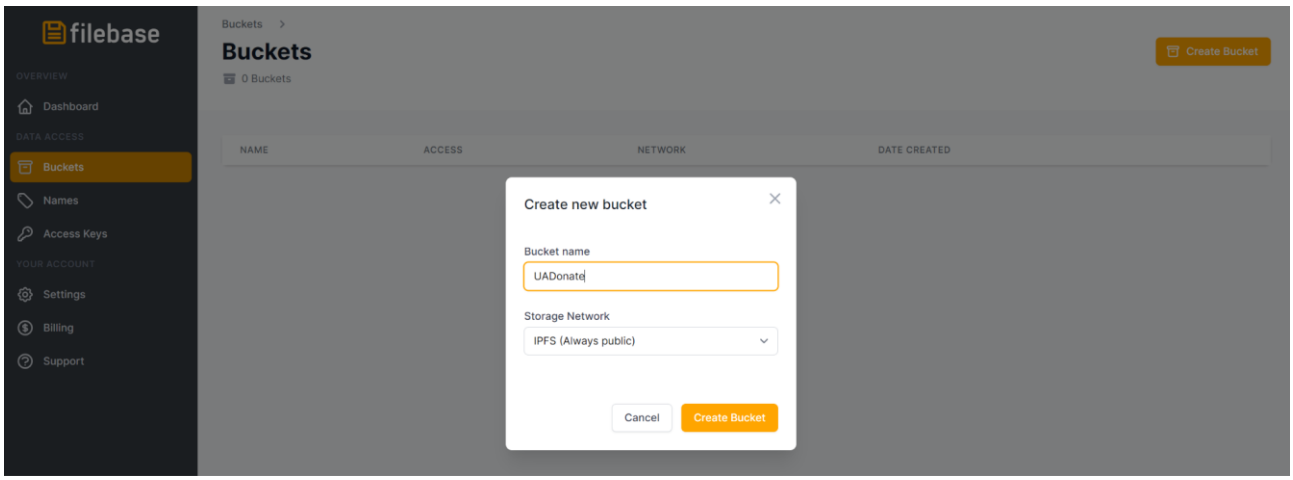

Рис. 2.6 Створення bucket в Filebase

Тепер потрібно знайти відповідні малюнки до токенів, які ми хочемо створити, завантажити їх, отримавши адресу в мережі ipfs, та створити метадані в форматі json, де вкажемо адресу відповідного зображення. Я буду створювати два токени: один для 10-дюймового дрона та один для 7-и дюймового. Метадані за стандартом для Opensea виглядатимуть так:

```
{
  "description": "This 10-inch FPV drone w
  "image": "ipfs://QmPRdoRFBkhaXmG9novZBhg
  "name": "10 inch FPV drone",
  "attributes": [
    {
      "trait_type": "Size",
      "value": "10 inch"
    }
  ]
}
```

Рис. 2.7 Метадані для NFT

Тепер завантажуємо в bucket два зображення, беремо їх адресу, вставляємо в метадані і завантажуємо метадані в bucket. Після проведення цих операцій, мій bucket виглядає ось так:

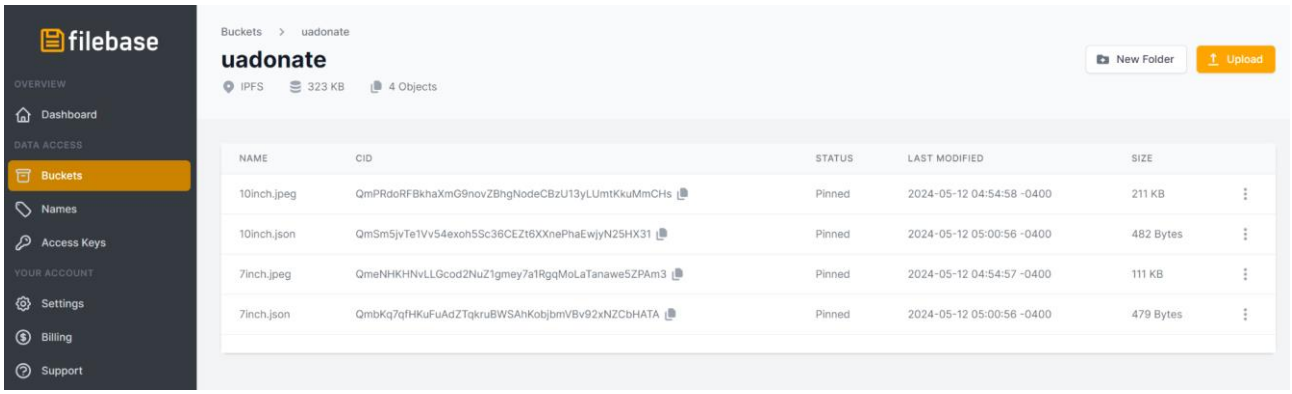

Рис. 2.8 Завантажені метадані в мережу ipfs

Знаючи ідентифікатори метаданих в мережі IPFS, можемо випускати наші токени:

Рис. 2.9 Випуск токену

Отже було випущено токен з необмеженою кількістю продажів, миттєвим доступом для купівлі та можливістю для будь-кого його придбати.

2.4.5 Динамічність

Після того як людина задонатила на дрон, починається його фізичне виробництво, про що інформується людина. Робимо абгрейд із загального токену дрону на його індивідуальний. Спочатку створимо новий, де буде заборона на продаж та ліміт в одну штуку. Після чого необхідно конвертувати 1 токен загального дрона на 1 токен індивідуального. Для цього викликаємо таку функцію:

```
    function upgradeToken(
        uint tokenId,
        uint newTokenId,
        uint amount,
        uint newAmount,
        address account
    ) onlyOwner public {
        require(_isTokenIssued(tokenId), "Token not issued");
        require(_isTokenIssued(newTokenId), "New token not issued");
        require(_balances[tokenId][account] >= amount);

        uint limit = _limits[newTokenId];
        uint availableTokens = (limit == 0 ? type(uint256).max : limit) -
_countTokens[newTokenId];
        require(availableTokens >= newAmount);

        _balances[tokenId][account] -= amount;
        _balances[newTokenId][account] += newAmount;

        _countTokens[tokenId] -= amount;
        _countTokens[newTokenId] += newAmount;
    }
```

Цей код дозволяє зменшити кількість одного токена та збільшити кількість іншого для користувача.

Після цього в силу вступає динамічність наших токенів, і ми можемо оновлювати метадані цього індивідуального токену (дрону, який в процесі виробництва) за допомогою функції:

```
    function setCustomURI(
        uint id,
        string calldata uri
    ) onlyOwner external {
        require(_isTokenIssued(id));
        _tokenURIs[id] = uri;
    }
```

Це забещпечує індивідуальний підхід до користувача і кожен може відслідковувати те, куди йде його донат.

2.4.6 Купівля та експлуатація

Після випуску нашого токену у нас з'являється можливість купівлі (якщо ми встановили відповідний дозвіл), тому викликаємо функцію purchaseToken, де вказуємо айді токену, який бажаємо купити, та суму головної валюти блокчейну, яку плануємо перевести:

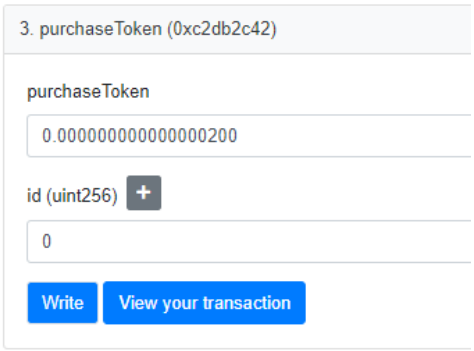

Рис 2.10 Купівля токену

В коді смартконтракту виконується запис івентів (логів), котрі потім дають можливість платформам для перегляду NFT відслідковувати токени на гаманцях. Однією з таких платформом є OpenSea. Логи нашої транзакції виглядають наступним чином:

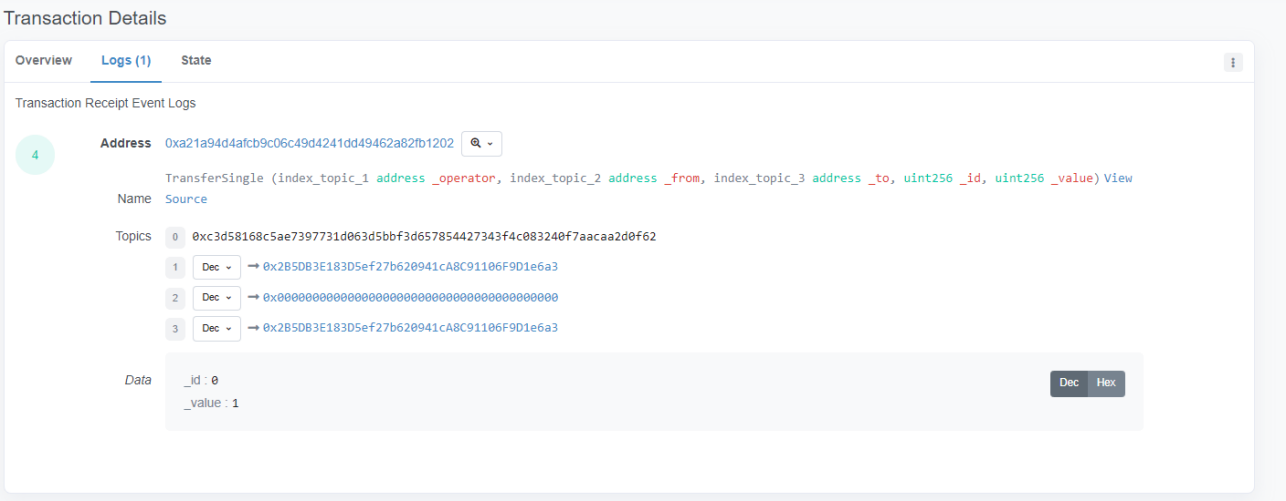

Рис 2.11 Логи транзакції купівлі токену

Оскільки контракт був створений такий чином, щоб в різних системах він міг розпізнаватися, як стандарт ERC-1155, то, відповідно, наш івент має також такі дані:

- оператор: хто визвав функцію;
- адреса відправника: при випуску використовується нульова адреса;
- адреса отримувача: покупець;
- айді токену;
- кількість токенів.

У результаті можемо переглянути наші токени через платформу OpenSea.

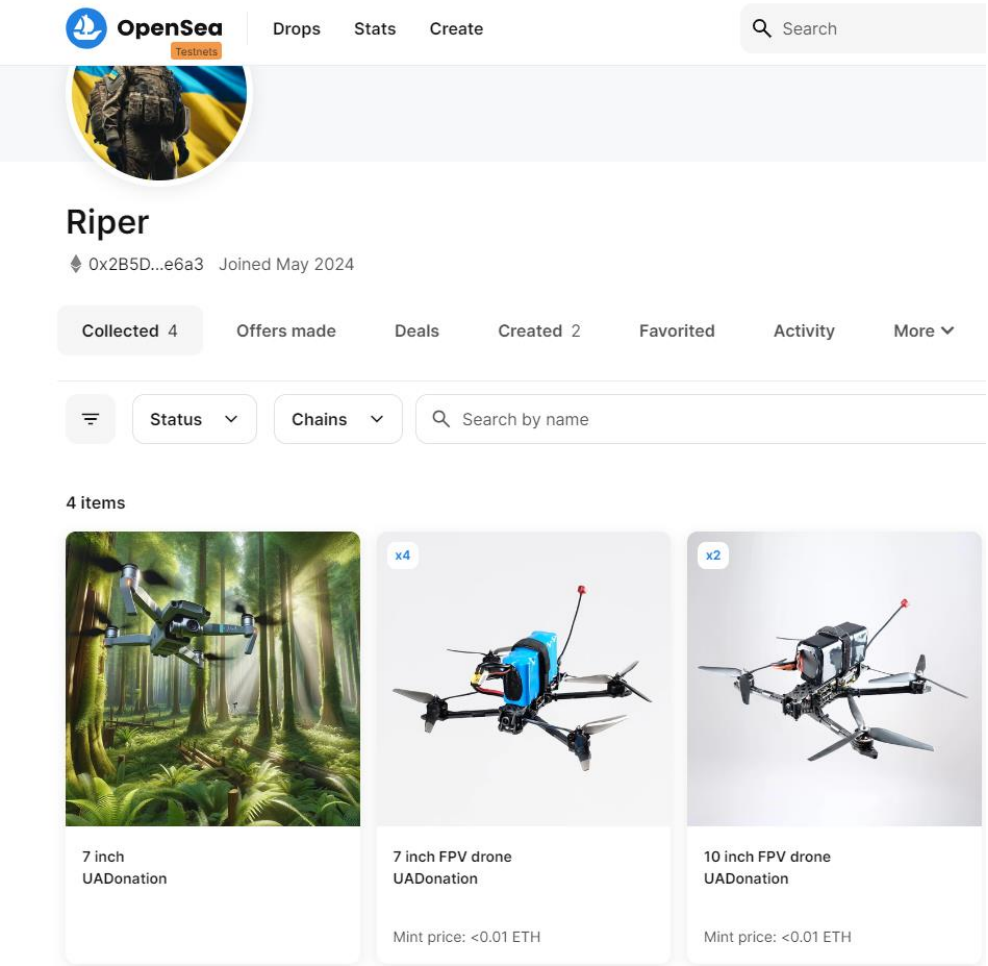

Рис. 2.12 Демонстрація акаунту на платформі OpenSea

Отже, через відповідність івентів нашого контракту івентам стандарту ERC-1155 ми бачимо наші токени, і будь-хто може переглянути їх за цим посиланням https://testnets.opensea.io/collection/uadonation.

Висновок

Під час написання цієї роботи було проведено дослідження про використання та експлуатацію динамічних невзаємозамінних токенів. Зараз цей напрям набуває все більшої популярності, оскільки довіра до NFT починає поступово повертатися після краху 2021 року. Але серед динамічних NFT стандартизація розвиваються повільніше, оскільки проєктів, які використовують цю технологію, ще не багато і немає потреб в загальному стандарті.

Динамічні NFT дозволяють нам створити індивідуальний підхід до користувача, змінюючи метадані його власності, тим самим впроваджуючи взаємодію з ним, що мотивує його на взаємний зв'язок з нашим продуктом. Наприклад, це може використовуватися як у сфері ігр, метавсесвітів тощо, так і у разі сертифікацією майна в реальному світі.

Кінцевим результатом роботи став розроблений новий стандарт для блокчейну Ethereum, котрий містить в собі ідею динамічності NFT і проводить аналогію з маркетплейсом в реальному світі. Таким чином, на основі цього стандарту був розроблений смартконтракт, котрий може використовуватися як донат-платформа для ЗСУ.

Отже, завдяки цій роботі вдалося дослідити на зараз дуже важливу тему динамічних невзаємозамінних токенів, яка починає розвиватися в сфері блокчейну і популяризуватися серед користувачів, та застосувати ці знання на проєкті, який в майбутньому може бути реалізований для повноцінного використання та допомозі ЗСУ.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. [Електронний ресурс] Smart contract
https://en.wikipedia.org/wiki/Smart_contract
(Перевірка на доступ до ресурсу - 05.05.2024)

2. [Електронний ресурс] What are Dynamic NFTs? A Comprehensive Guide
https://medium.com/@jackjill7659/what-are-dynamic-nfts-a-comprehensive-guide-fc9e3255f0eb
(Перевірка на доступ до ресурсу - 05.05.2024)

3. [Електронний ресурс] An In-depth Insight at Digital Ownership Through Dynamic NFTs
https://www.researchgate.net/publication/366127167_An_In-depth_Insight_at_Digital_Ownership_Through_Dynamic_NFTs
(Перевірка на доступ до ресурсу - 05.05.2024)

4. [Електронний ресурс] Що таке IPFS і для чого потрібний новий протокол?
https://medium.com/@sergey.gnatushok/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-ipfs-%D1%96-%D0%B4%D0%BB%D1%8F-%D1%87%D0%BE%D0%B3%D0%BE-%D0%BF%D0%BE%D1%82%D1%80%D1%96%D0%B1%D0%BD%D0%B8%D0%B9-%D0%BD%D0%BE%D0%B2%D0%B8%D0%B9-%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB-bba3d9acebd1
(Перевірка на доступ до ресурсу - 05.05.2024)

5. [Електронний ресурс] Невзаємозамінний токен
https://uk.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B2%D0%B7%D0%B0%D1

%94%D0%BC%D0%BE%D0%B7%D0%B0%D0%BC%D1%96%D0%BD%D0%BD%D0%B8%D0%B9_%D1%82%D0%BE%D0%BA%D0%B5%D0%BD
(Перевірка на доступ до ресурсу - 05.05.2024)

6. [Електронний ресурс] Introducing ERC721TL: The Ultimate Creator Contract https://medium.com/@TransientLabs/introducing-erc721tl-the-ultimate-creator-contract-67520106b15f
(Перевірка на доступ до ресурсу - 05.05.2024)

7. [Електронний ресурс] ERC-6551: Ushering in the Dawn of Dynamic NFTs https://medium.com/@pearliboy1/erc-6551-ushering-in-the-dawn-of-dynamic-nfts-811c08228e80
(Перевірка на доступ до ресурсу - 05.05.2024)

8. [Електронний ресурс] ERC-2981: NFT Royalty Standard https://eips.ethereum.org/EIPS/eip-2981
(Перевірка на доступ до ресурсу - 05.05.2024)

9. [Електронний ресурс] ERC-1948: Non-fungible Data Token https://eips.ethereum.org/EIPS/eip-1948
(Перевірка на доступ до ресурсу - 05.05.2024)

10. [Електронний ресурс] ERC-1155: Multi Token Standard https://eips.ethereum.org/EIPS/eip-1155
(Перевірка на доступ до ресурсу - 05.05.2024)