КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

РОЗРОБКА СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З NFT У МЕРЕЖІ ERC721

Виконав студент 4-го курсу
Богдан Копанічук

_____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Лариса Катеринич

_____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем

«__» _____ 2021 р.,
Протокол №_____
Завідувач кафедри
Олександр Провотар

Київ – 2021

РЕФЕРАТ

Обсяг роботи 41 сторінка, 2 ілюстрації, 2 таблиці, 16 джерел посилання.

КРИПТОГРАФІЯ, БЛОКЧЕЙН, НЕВЗАЄМОЗАМІННИЙ ТОКЕН, NFT, ERC-721, ERC-1155, ETHEREUM.

Об’єктом роботи є централізована система для взаємодії з невзаємозамінними токенами на основі блокчейну. Предметом роботи є система для взаємодії з NFT розроблена у вигляді клієнт-серверного веб-застосунку.

Метою даної роботи є створення системи для взаємодії з невзаємозамінними токенами у мережі ERC-721 та її імплементація у вигляді клієнт-серверного застосунку.

Методи розроблення: розробка та імплементація уніфікованої моделі невзаємозамінного токену, розробка інтерфейсу для взаємодії з моделлю. Інструменти розроблення: безкоштовний текстовий редактор Visual Studio Code, мова програмування Python, мова програмування JavaScript, мова розмітки гіпертексту HTML5, мова стилю сторінок CSS, відкритий Python-фреймворк для розробки веб-систем Django, відкритий JavaScript-фреймворк для створення інтерфейсів користувача VUE.js.

Результати роботи: В ході роботи було розглянуто поняття взаємозамінного та невзаємозамінного активу. Розглянуті нюанси пов’язані з цим, а також сформульована ідея невзаємозамінного цифрового активу.

Виконано огляд ідеї використання невзаємозамінних токенів для підтвердження права власності. Було визначено проблеми, пов’язані з реалізацією ідеї невзаємозамінного активу та способи їх вирішення за допомогою блокчейну, а саме за допомогою ідеї невзаємозамінного токену

для підтвердження прав володіння. Також виконано загальний огляд існуючих систем взаємодії з невзаємозамінними токенами. Проаналізовані існуючі стандарти невзаємозамінних токенів, також вимоги до створення невзаємозамінних токенів та їх метаданих. Виконана розробка та імплементація уніфікованої моделі невзаємозамінного токену. Розроблено інтерфейс для взаємодії з моделлю у вигляді клієнт-серверного застосунку. Як результат була розроблена централізована система для взаємодії з невзаємозамінними токенами.

Така система для взаємодії з невзаємозамінними токенами може бути використана як оптимальне L2 рішення для взаємодії з NFT на ранніх етапах розвитку блокчейну, тобто доки більшість децентралізованих систем буде сильно програвати у ефективності централізованим системам. Згодом система буде застосовуватись як універсальний хаб для створення невзаємозамінних токенів у різних мережах.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об'єкта розробки. На протязі усієї історії люди стикаються з проблемою підтвердження факту володіння активами та правами на їх використання. З часом люди навчились вирішувати цю проблему для речей реального світу за допомогою державного апарату та різних міжрівневих реєстрів, але для речей інтелектуальної власності вона й досі залишається відкритою. З розвитком інтернету також з’явився цілковито новий тип активів – цифровий актив. Оскільки такий нематеріальний актив зазвичай має фінансову цінність, а середовище де він існує виходить далеко за рамки державного апарату – проблема підтвердження права власності набуває нового масштабу. Ідеєю вирішення цієї проблеми є використання невзаємозамінних токенів.

Актуальність роботи та підстави для її використання. Довгий час ідея використання невзаємозамінних токенів для підтвердження права власності залишалася нездійсненною, оскільки до недавнього часу існували лише централізовані та псевдо-децентралізовані системи, які не можуть забезпечити унікальність і незмінність токенів. Але з появою технології блокчейну створення невзаємозамінних токенів стало цілком здійсненним завданням. А оскільки блокчейн перебуває тільки на ранньому етапі свого розвитку, немає одного єдиного стандарту невзаємозамінного токену і комісії за операції у блокчейні є занадто коштовними. Тому актуальним є створення централізованої системи, яка пропонує уніфіковану модель токену, інструменти для взаємодії з нею, а також можливість конвертації і експорту моделі у токен будь-якого блокчейну, але яка не матиме недоліків децентралізованої системи.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення централізованої системи для взаємодії з NFT у мережі ERC-721. Для досягнення цієї мети поставлено такі завдання:

- Дослідити існуючі системи взаємодії з невзаємозамінними токенами;
- Дослідити існуючі стандарти невзаємозамінних токенів;
- Дослідити вимоги до створення невзаємозамінних токенів та їх метаданих;
- Розробити та імплементувати уніфіковану модель невзаємозамінного токену;
- Розробити інтерфейс для взаємодії з моделлю у вигляді клієнт-серверного застосунку;

Об’єкт, методи й засоби розроблення. Об’єктом розробки є централізована система для взаємодії з невзаємозамінними токенами на основі блокчейну.

В якості інструменту розробки системи для взаємодії з невзаємозамінними токенами було обрано безкоштовний текстовий редактор, а саме Visual Studio Code - для написання коду на мові програмування Python та JavaScript. Головною перевагою Visual Studio Code є підтримка інструментів розробника для багатьох фреймворків. Для розробки системи також було використано мову розмітки гіпертексту HTML5, мову стилю сторінок CSS3, відкритий Python-фреймворк для розробки веб-систем Django, відкритий JavaScript-фреймворк для створення інтерфейсів користувача VUE.js, контейнерізатор додатків Docker.

Можливі сфери застосування. Централізована система для взаємодії з невзаємозамінними токенами може бути використана як оптимальне L2 рішення для взаємодії з NFT на ранніх етапах розвитку блокчейну, тобто доки більшість децентралізованих систем буде сильно програвати у ефективності централізованим системам. Згодом система буде застосовуватись як універсальний хаб для створення невзаємозамінних токенів у різних мережах.

РОЗДІЛ 1 - ПОНЯТТЯ НЕВЗАЄМОЗАМІННОГО АКТИВУ

Більшість дискусій щодо невзаємозамінних активів починаються з введення ідеї замінності, яка визначається як «здатність замінити або бути заміненим іншим ідентичним предметом». Щоб краще зрозуміти, що може становити невзаємозамінних актив, просто подумайте про більшість своїх речей. Стілець, на якому ви сидите, ваш телефон, ноутбук. Всі вони можуть підпадати під категорію невзаємозамінних речей.

1.1 Відносність поняття взаємозамінності

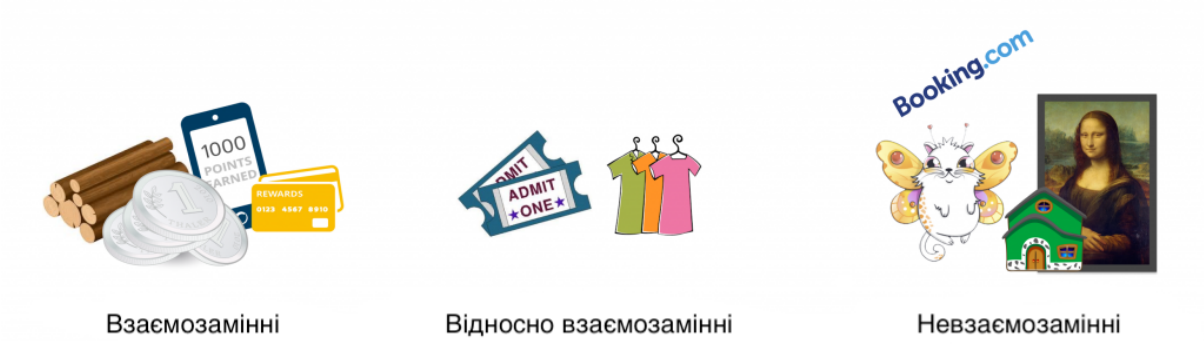

Рис №1 Приклад різних типів активів

Виявляється, що зі взаємозамінними активами насправді також не все так просто. Валюта - класичний приклад взаємозамінного активу. П'ять гривень - це завжди п'ять гривень, незалежно від серійного номера на конкретній п’яти гривневій купюрі або від того, чи лежать ці п'ять гривень на вашому банківському рахунку. Можливість замінити п’яти гривневу купюру іншою п’яти гривневою купюрою (чи п'ятьма купюрами) - ось що робить валюту взаємозамінною.

Зверніть увагу, що взаємозамінність відносна, вона дійсно застосовна тільки при порівнянні декількох речей. Розглянемо авіаквитки бізнес-класу, економ-класу і першого класу. Кожен квиток взаємозамінний в межах свого

класу, але ви не можете чесно обміняти квиток першого класу на квиток бізнес-класу. Навіть стілець, на якому ви сидите, взаємозамінний із стільцем тієї ж моделі, якщо тільки у вас немає особливої прихильності до вашого конкретного стільця.

Цікаво, що взаємозамінність також може бути суб'єктивною. Повернемося до прикладу з авіаквитками: людина, якій важливо сидіти на місці біля вікна, а не у проході, може не вважати два квитки економ-класу взаємозамінними. Так само рідкісна монета може коштувати 1 копійку для звичайної людини, але значно більше для колекціонера монет. Деякі з цих нюансів стають важливими при представленні цих предметів у блокчейні.

1.2 Невзаємозамінні цифрові активи

Як до появи криптовалют у нас були цифрові валюти (наприклад, бали авіакомпаній, внутрішньоігрові валюти), так і з моменту появи Інтернету у нас були недовговічні цифрові активи. Доменні імена, квитки на заходи, внутрішньоігрові предмети, навіть облікові записи в соціальних мережах, таких як Twitter або Facebook, - усе це невзаємозамінні цифрові активи; вони просто розрізняються по можливості торгівлі, ліквідності і сумісності. І багато з них неймовірно цінні: Тільки в 2018 році компанія Epic Games заробила 2,4 млрд доларів на продажі костюмів у своїй free - to - play грі Fortnite, ринок квитків на заходи, за прогнозами, досягне 68 млрд доларів в 2025 році, а ринок доменних імен продовжує демонструвати стрімкий ріст.

Отже, ясно, що у нас вже є безліч цифрових речей. Але в якій мірі ми "володіємо" цими цифровими речами? Якщо цифрове володіння означає, що річ належить тільки вам, а не комусь іншому, тоді ви в деякому розумінні володієте нею. Але якщо цифрова власність більше схожа на власність у фізичному світі (свобода володіння і передачі на невизначений

термін), то у випадку з цифровими активами це не завжди так. Швидше, ви володієте цими активами в певних контекстах, що може полегшити або прискладнити їх переміщення. Спробуйте продати речі для онлайн гри на відкритих торгових майданчиках і ви зрозумієте, наскільки складно переміщати цифрові активи від однієї людини до іншої.

Саме тут на допомогу приходить блокчейн – він забезпечує координаційний рівень для цифрових активів, надаючи користувачам право власності і управління.

РОЗДІЛ 2 – НЕВЗАЄМОЗАМІННІ ТОКЕНИ НА ОСНОВІ БЛОКЧЕЙНУ

З появою блокчейну ідея використання невзаємозамінних токенів для підтвердження права власності стала цілком виконуваною. Як і фізичні гроші, криптовалюти є взаємозамінними, тобто ними можна торгувати або обмінювати одну на іншу. Наприклад, один біткойн завжди дорівнює за вартістю іншому біткойну. Подібним чином одна одиниця ефіру завжди дорівнює іншій одиниці. Ця характеристика взаємозамінності робить криптовалюти придатними для використання як безпечний засіб транзакцій у цифровій економіці.

Невзаємозамінні токени - це еволюція щодо порівняно простої концепції криптовалют. Сучасні фінансові системи складаються із складних торгових та позичкових систем для різних типів активів, починаючи від нерухомості і закінчуючи договорами позики і творами мистецтва. Забезпечуючи цифрове представлення фізичних активів, NFT є кроком вперед у відновленні цієї інфраструктури.

Невзаємозамінні токени зміщують крипто-парадигму, роблячи кожен токен унікальним і незамінним, тим самим унеможливлюючи, щоб один незамінний токен дорівнював іншому. Вони є цифровими поданнями активів і їх уподібнюють цифровим паспортам, оскільки кожен токен містить унікальну непередавану ідентичність, щоб відрізняти його від інших токенів. Вони також є розширюваними, тобто ви можете поєднувати один NFT з іншим, щоб «виробити» третій, унікальний невзаємозамінний токен.

Блокчейн додає декілька унікальних властивостей до невзаємозаміннимих активів, які міняють стосунки користувачів і розробників з цими активами. Давайте детально розберемо ці властивості.

2.1 Стандартизація

Традиційні цифрові активи - від квитків на заходи до доменних імен - не мають єдиного представлення у цифровому світі. Гра, швидше за все, представляє свої внутрішньоігрові колекційні предмети абсолютно інакше, ніж система продажу квитків на заходи. Представляючи неігрові токени в публічних блокчейнах, розробники можуть створити загальні, багаторазово використовувані, успадковувані стандарти, що відносяться до усіх невзаємозамінних токенів. Вони включають такі базові примітиви, як володіння, передача і простий контроль доступу. Додаткові стандарти (специфікації для відображення NFT, наприклад) можуть бути накладені згори для додаткового відображення усередині додатків.

Blockchain Standards ERC20, ERC721, ERC1155, IBC
Internet Standards TCP/IP, HTTP, HTML/CSS, REST
Content Standards File formats, HTML/CSS, JSON

Таблиця №1 Приклади стандартів цифрового світу

Це аналогія іншим будівельним блокам цифрового світу, таким як формат файлів JPEG або PNG для зображень, HTTP для запитів між комп'ютерами і HTML / CSS для відображення контенту в Інтернеті. Блокчейн додає згори шар, який дає розробникам абсолютно новий набір примітивів, на основі яких вони можуть будувати додатки

2.2 Сумісність

Стандарти невзаємозамінних токенів дозволяють токенам легко переміщатися в декількох екосистемах. Коли розробник запускає новий проект NFT, ці NFT негайно стають доступними для перегляду в десятках різних постачальників гаманців, торгуються на торгових майданчиках і, останнім часом, відображаються у віртуальних світах. Це можливо, тому що відкриті стандарти надають ясний, погоджений, надійний і дозволений API для читання і запису даних.

2.3 Можливість торгівлі

Найпривабливішою функцією, що забезпечується сумісністю, є вільна торгівля на відкритих торгових майданчиках. Уперше користувачі можуть переміщати предмети за межі свого первинного середовища на ринок, де вони можуть скористатися складними можливостями торгівлі, такими як аукціони, торги, комплектація і можливість продажу у будь-якій валюті, наприклад, в стабільних монетах і валютах, специфічних для конкретного застосування.

Для розробників ігор можливість торгівлі активами є переходом від закритої економіки до відкритої, вільно-ринкової економіки. Розробникам ігор більше не треба управляти усіма елементами своєї економіки - від

постачань ресурсів до ціноутворення і контролю за рухом капіталу. Замість цього вони можуть дозволити вільним ринкам зробити усю важку роботу.

2.4 Ліквідність

Миттєва можливість торгівлі невзаємозамінними токенами приведе до підвищення ліквідності. Торгові майданчики NFT можуть обслуговувати самі різні аудиторії - від трейдерів до початківців, що дозволяє ширше розкрити активи ширшому колу покупців. Точно так, як і бум ICO в 2017 році породив новий клас активів, рухомий миттєво ліквідними токенами, невзаємозамінні токени розширюють ринок унікальних цифрових активів.

2.5 Незмінність і доказовий дефіцит

Смарт-контракти дозволяють розробникам встановлювати жорсткі обмеження на постачання невзаємозамінних токенів і забезпечувати постійні властивості, які не можуть бути змінені після випуску NFT. Наприклад, розробник може програмно забезпечити, щоб можна було створити тільки певну кількість певних рідкісних елементів, зберігаючи при цьому необмежену кількість загальніших елементів. Розробники також можуть забезпечити, щоб певні властивості не мінялися з часом, шляхом кодування їх в ланцюжку. Це особливо цікаво для мистецтва, яке багато в чому покладається на доказову рідкість оригінального

2.6 Програмованість

Звичайно, як і традиційні цифрові активи, NFT повністю програмовані. В CryptoKitties (як приклад NFT проекту) вбудовані

механізми розведення прямо в контракт, який представляє цифрових кішок. Багато з сьогоднішніх невзаємозамінних токенів має складнішу механіку, таку як кування, створення, погашення, випадкова генерація і т. д. Простір дизайну повно можливостей.

РОЗДІЛ 3 – СТАНДАРТИЗАЦІЯ НЕВЗАЄМОЗАМІННИХ ТОКЕНІВ

Стандарти токенів - це частина того, що робить невзаємозамінні токени потужними. Вони дають розробникам гарантію, що активи поводитимуться певним чином, і точно описують, як взаємодіяти з базовою функціональністю активів. Усі NFT мають змінну uint256 з ім'ям tokenId, тому для будь-якого контракту адреса uint256 tokenId має бути глобально унікальною. Зараз існує декілька поширених стандартів для невзаємозамінних токенів, давайте детально розберемо ці стандарти.

3.1 ERC-721

ERC-721- стандарт токенів у мережі смарт-контрактів Ethereum. Він був розроблений у вересні 2017 року і уперше був використаний у проекті CryptoKitties, став першим стандартом для представлення невзаємозамінних цифрових активів. ERC-721 - це успадкований стандарт смарт-контрактов Solidity, що означає, що розробники можуть легко створювати нові контракти, сумісні з ERC-721, імпортуючи їх з бібліотеки OpenZeppelin (тут у нас є корисне керівництво по створенню вашого першого контракту ERC-721). ERC-721 насправді відносно простий: він забезпечує зіставлення унікальних ідентифікаторів (кожен з яких представляє окремий актив) з адресами, які представляють власника цього ідентифікатора. ERC-721 також надає дозволений спосіб передання цих активів з використанням методу

```
interface ERC721 {
  function ownerOf(uint256 _tokenId) external view returns (address);
  function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
}
```

Якщо замислитися, ці два методи – це все, що треба для представлення невзаємозамінного токену: спосіб перевірити, хто чим володіє, і спосіб переміщати речі. Є декілька інших особливостей стандарту (деякі з яких виявляються дуже важливими для торгових майданчиків NFT), але ядро ERC-721 досить просте.

Список методів стандарту ERC-721:

- function balanceOf(address _owner) external view returns (uint256);
- function ownerOf(uint256 _tokenId) external view returns (address);
- function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable;
- function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;
- function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
- function approve(address _approved, uint256 _tokenId) external payable;
- function setApprovalForAll(address _operator, bool _approved) external;
- function getApproved(uint256 _tokenId) external view returns (address);
- function isApprovedForAll(address _owner, address _operator) external view returns (bool);

Список подій стандарту ERC-721:

- event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId);

- event Approval(address indexed _owner, address indexed _approved, uint256 indexed _tokenId);
- event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);

3.2 ERC-1155

ERC-1155, уперше розроблений командою Enjin у 2018 році, привносить ідею напівзамінюваності до світу невзаємозамінних токенів. У стандарті ERC-1155 ідентифікатори представляють не окремі активи, а класи активів. Наприклад, ідентифікатор може бути "мечами", а в гаманці може зберігатися 1000 таких мечів. В цьому випадку метод balanceOf поверне кількість мечів, що належать гаманцю, і користувач може передати будь-яку кількість цих мечів, викликавши transferFrom з ідентифікатором "меч".

```
interface ERC1155 {
  function balanceOf(address _owner, uint256 _id) external view returns (address);
  function transferFrom(address _from, address _to, uint256 _id, uint256 quantity) exter
nal payable;
}
```

Однією з переваг цього стандарту є ефективність: з ERC721, якщо користувач хотів передати 1000 мечів, йому треба було б змінити стан смарт-контракта (шляхом виклику методу transferFrom) для 1000 унікальних токенов. При використанні ERC-1155 розробникові досить викликати transferFrom з кількістю 1000 і виконати одну операцію передачі. Ця підвищена ефективність, звичайно ж, пов'язана з втратою інформації : ми більше не можемо простежити історію окремого меча.

Стандарт ERC-1155 надає розширений набір функцій ERC-721, що означає, що актив ERC-721 може бути побудований з використанням ERC-1155 (у вас просто буде окремий ідентифікатор і кількість 1 для кожного активу). Завдяки цим перевагам останнім часом ми стали свідками зростаючого впровадження стандарту ERC1155.

Таблиця №2 Анатомія стандартів ERC20, ERC721 і ERC1155. ERC20 зіставляє адреси з сумами, ERC721 зіставляє унікальні ідентифікатори з власниками, а ERC1155 має вкладене зіставлення ідентифікаторів з власниками і сумами.

Список методів стандарту ERC-1155:

- function safeTransferFrom(address _from, address _to, uint256 _id, uint256 _value, bytes calldata _data) external;
- function safeBatchTransferFrom(address _from, address _to, uint256[] calldata _ids, uint256[] calldata _values, bytes calldata _data) external;
- function balanceOf(address _owner, uint256 _id) external view returns (uint256);
- function balanceOfBatch(address[] calldata _owners, uint256[] calldata _ids) external view returns (uint256[] memory);
- function setApprovalForAll(address _operator, bool _approved) external;
- function isApprovedForAll(address _owner, address _operator) external view returns (bool);

Список подій стандарту ERC-1155:

- event TransferSingle(address indexed _operator, address indexed _from, address indexed _to, uint256 _id, uint256 _value);
- event TransferBatch(address indexed _operator, address indexed _from, address indexed _to, uint256[] _ids, uint256[] _values);
- event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);
- event URI(string _value, uint256 indexed _id);

3.3 Стандарти, відмінні від Ethereum

Тоді як в Ethereum відбуваються більшість подій, в інших мережах з’являється ще декілька стандартів невзаємозамінних токенів. DGoods, піонером якого є команда Mythical Games, націлений на створення багатофункціонального межцепочечного стандарту, починаючи з EOS. Проект Cosmos також розробляє модуль NFT, який може бути використаний як частину Cosmos SDK.

РОЗДІЛ 4 – МЕТАДАНІ НЕВЗАЄМОЗАМІННИХ ТОКЕНІВ

Як уже згадувалося, метод ownerOf дозволяє знайти власника NFT. Наприклад, запросивши ownerOf (1500718) в смарт-контракті CryptoKitties, ми можемо побачити, що власником CryptoKitty #1500718 на момент написання є обліковий запис з адресою 0x6452. Це можна перевірити, відвідавши сайт CryptoKitties.co.

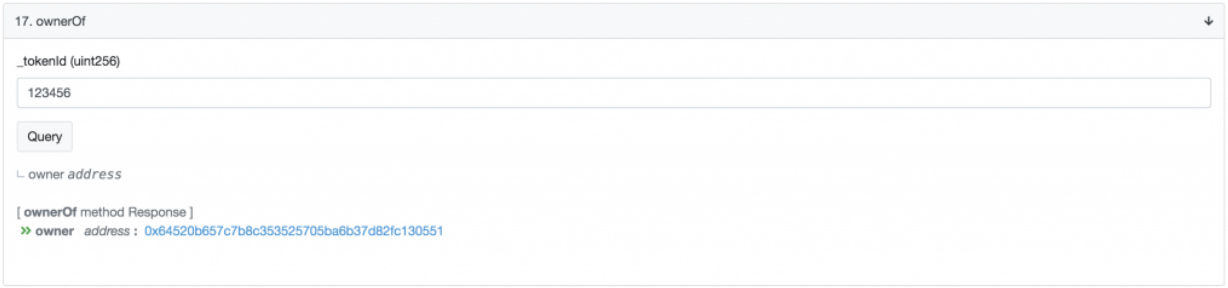

Рис №2 Приклад токену з сайту CryptoKitties.co

Але як при відображенні на сайті CryptoKitties з'ясовують, як виглядає CryptoKitty #1500718? А як щодо його назви і унікальних атрибутів? Тут на допомогу приходять метадані. Метадані надають описову інформацію для конкретного ідентифікатора токена. У разі CryptoKittty метадані - це ім'я кішки, зображення кішки, опис і будь-які додаткові характеристики (що називаються "атрибутами" у разі CryptoKitties). У разі квитка на захід метадані можуть включати дату заходу і тип квитка, а також ім'я і опис. Метадані для вищезгаданого кота можуть виглядати приблизно так:

```
{
  "name": "Duke Khanplum",
  "image": "https://storage.googleapis.com/ck-kitty-image/0x06012c8cf97bead5deae237070f9
587f8e7a266d/1500718.png",
  "description": "Heya. My name is Duke Khanplum, but I've always believed I'm King Henr
y VIII reincarnated."
}
```

Виникає питання, як і де зберігати ці дані, щоб до них могли отримати доступ додатки, які працюють з невзаємозамінними токенами.

4.1 On-chain або off-chain

Перше рішення для розробників – як зберігати метадані: в мережі або поза нею. Тобто ви зберігаєте метадані безпосередньо в смарт-контракті, представляючому токен, або розміщуєте їх окремо?

4.1.1 On-chain метадані

Переваги зберігання метаданих в мережі полягають в наступному:

1) вони постійно зберігаються разом з токеном, зберігаючись упродовж життєвого циклу будь-якого конкретного застосунку,

2) вони можуть змінюватися відповідно до логіки ланцюжка.

Пункт №1 важливий, якщо передбачається, що активи матимуть довготривалу цінність далеко за межами їх первинного середовища. Наприклад, передбачається, що витвір цифрового мистецтва збережеться на протязі віків, незалежно від того, чи існує оригінальний веб-сайт, на якому воно було створене. Тому важливо, щоб метадані зберігалися упродовж усього життєвого циклу ідентифікатора токена.

Крім того, логіка мережі зможе взаємодіяти з метаданими. Наприклад, у випадку з CryptoKitties "покоління" CryptoKitty впливає на те, як швидко CryptoKitty може розмножуватися, а розмноження відбувається в мережі (кішки більш високого покоління розмножуються повільніше). Тому логіка усередині смарт-контракта має бути здатна прочитувати метадані з його внутрішнього стану.

4.1.2 Off-chain метадані

Незважаючи на ці переваги, більшість проектів зберігають свої метадані поза мережею просто із-за поточних обмежень зберігання блокчейна Ethereum. Таким чином, стандарт ERC721 включає метод tokenURI, який розробники можуть реалізувати, щоб повідомити додаткам, де знайти метадані для цього елементу.

```
function tokenURI(uint256 _tokenId) public view returns (string)
```

Метод tokenURI повертає загальнодоступний URL. Він, у свою чергу, повертає словник даних JSON, щось подібне до прикладу словника для CryptoKitty вище. Ці метадані повинні відповідати офіційному стандарту метаданих ERC721, щоб їх могли використати інші застосунки.

4.1.3 Off-chain рішення для зберігання

Якщо ви зберігаєте метадані поза мережею, у вас є декілька варіантів:

- **Централізовані сервери** - найпростіший спосіб зберігати метадані де-небудь на централізованому сервері або в хмарному сховищі, наприклад AWS. Звичайно, у цього є недоліки:
    - 1) розробник може змінювати метадані по своєму бажанню
    - 2) якщо проект переходить в автономний режим, метадані можуть зникнути з початкового джерела. Щоб пом'якшити проблему №2, тепер існує декілька сервісів, які кэшують метадані на своїх власних серверах, щоб гарантувати ефективне обслуговування користувачів, навіть якщо початкове рішення для хостингу вийде з ладу

- **IPFS** - все більше число розробників, особливо у сфері цифрового мистецтва, використовують InterPlanetary File System (IPFS) для зберігання метаданих поза мережею. IPFS - це однорангова система зберігання файлів, яка дозволяє розміщувати контент на різних комп'ютерах, так що файл реплицируется у багатьох різних місцях. Це гарантує, що
    - A) метадані незмінні, оскільки вони однозначно адресуються хешем файлу
    - B) до тих пір, поки є вузли, що бажають розмістити дані, дані зберігатимуться з часом. Тепер існують служби, такі як Pinata, які спрощують цей процес для розробників, управляючи інфраструктурою для розгортання і управління вузлами IPFS, а довгождана мережа Filecoin (теоретично) додасть шар поверх IPFS, щоб стимулювати вузли до розміщення

РОЗДІЛ 5 – СТВОРЕННЯ ЦЕНТРАЛІЗОВАНОЇ СИСТЕМИ ДЛЯ ВЗАЄМОДІЇ З НЕВЗАЄМОЗАМІННИМИ ТОКЕНАМИ У МЕРЕЖІ ERC-721

Метою цього розділу є створення централізованої системи для взаємодії з невзаємозамінними токенами, що за потреби можуть бути викладені у мережу Ethereum у форматі токена ERC-721. Система повинна мати інструменти для взаємодії з токенами, а саме інструменти створення, інструменти опису та інструменти передачі токену. Важливою вимогою є пряма сумісність формату токену нашої системи з токеном формату ERC-721. Для реалізації системи потрібно розробити та імплементувати уніфіковану модель невзаємозамінного токену, яка б забезпечила таку пряму сумісність. Система буде складатись з двох частин – клієнтського застосунку на серверу для роботи цього застосунку.

5.1 Розробка та імплементація уніфікованої моделі невзаємозамінного токену

Щоб краще зрозуміти, що повинна мати наша модель для підтримання сумісності с токеном у мережі ERC-721 звернемося до розділу №3.

Ми знаємо, що будь який невзаємозамінний токен у мережі Ethereum повинен мати два обов’язкових атрибути – адреса Ethereum гаманця та унікальний tockenId. Спираючись на цю інформацію можемо побудувати базову модель токену:

```
class BaseToken(Model):
    tokenId = models.CharField
    address = models.CharField(/^0x[a-fA-F0-9]{40}$/)
```

Тепер в нас є базова модель токену, але щоб такий токен мав якусь цінність – цього не достатньо. Згідно розділу №4 ми знаємо щоб токен міг дійсно називатися NFT він повинен також мати метадані. Структура цих метаданих нам вже відома, отже можемо побудувати їх модель:

```
class Metadata(Model):
    title = models.CharField
    type = models.CharField
    name = models.CharField(Identifies the asset to which this NFT
    represents)
    description = models.CharField(Describes the asset to which this
    NFT represents)
    content = models.CharField(A URI pointing to a resource with
    mime type image/* representing the asset to which this NFT
    represents. Consider making any images at a width between 320
    and 1080 pixels and aspect ratio between 1.91:1 and 4:5 inclusive)
```

Тепер ми маємо всі необхідні нам моделі для створення токену сумісного з токеном у мережі ERC-721. Наступний етап – адаптація цих моделей для роботи у нашій системі.

5.2 Розробка серверної частини системи

Для розробки серверної частини системи будо обрано мову Python, а у якості фреймворку веб-додатків буде слугувати Django та Django-Rest-

Framework. Спочатку треба створити моделі даних для роботи нашої системи. Початковою моделлю нашої системи буде User – саме він буде оперувати системними інструментами та ініціювати будь яку дію у системі. В якості базової моделі будемо використовувати модель запропоновану фреймворком Django – AbstractUser, але доповнимо її. Саме до цієї моделі ми будемо підв’язувати токени у нашій системі і виконувати аутентифікацію.

```
class CustomUser(AbstractUser):
    """
    A User model that uses `email` as it's default identifier instead of
    username.
    """

    username = None
    email = models.EmailField(_('email address'), unique=True)
    bio = models.TextField()
    gender = models.CharField(
        max_length=140,
        null=True,
        choices=(
            ('Male', 'Male'),
            ('Female', 'Female'),
            ('Other', 'Other')
        )
    )
    birth_date = models.DateField(null=True, blank=True)
    pro = models.BooleanField(default=False)
```

Далі треба адаптувати та імплентувати розроблені раніш моделі невзаємозамінних токенів для роботи у нашій системі. Ми визначили, що будь яких токен буде прив'язаний до користувача – як до творця. А метадані вже будуть прив'язані лише тільки до одного унікального токену, так само як і токен до метаданих. Тоді наші моделі будут виглядати ось так:

```
class BaseToken(models.Model):
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    tokenId = models.CharField(max_length=200, unique=True, null=True,
blank=True)
    address = models.CharField(max_length=42, null=True, blank=True)

    def __str__(self):
        return self.tokenId
```

та

```
class Metadata(models.Model):
    token = models.OneToOneField(BaseToken, on_delete=models.CASCADE,
primary_key=True)
    title = models.CharField(max_length=255, null=False, blank=False)
    type = models.CharField(max_length=255, null=False, blank=False)
    name = models.CharField(max_length=255, help_text="Identifies the asset to
which this NFT represents", null=True, blank=True)
    description = models.CharField(max_length=255, help_text="Describes the
asset to which this NFT represents", null=True, blank=True)
    content = models.CharField(max_length=255, help_text="A URI pointing to
a resource with mime type image/* representing the asset to which this NFT
```

```
represents. Consider making any images at a width between 320 and 1080
pixels and aspect ratio between 1.91:1 and 4:5 inclusive.", null=True,
blank=True)


    def __str__(self):
        return self.title
```

Наступним кроком є створення кінцевих точок для звернення з клієнтського додатку. Реалізація відбувається стандартними інструментами Django-Rest-Api, але треба відзначити чітку прив'язку до користувача при звернені до будь якої кінцевої точки. Це можна побачити на прикладі базового токену:

```
class BaseTokenViewSet(viewsets.ModelViewSet):
    """
    This viewset automatically provides `LIST`, `CREATE`, `RETRIEVE`,
    `UPDATE` and `DESTROY` actions.
    """

    serializer_class = serializers.BaseTokenSerializer
    queryset = models.BaseToken.objects.all()
    def get_queryset(self):
        """
        This view should return a list of all the purchases
        for the currently authenticated user.
        """
        user = self.request.user
        return models.BaseToken.objects.filter(user=user)
```

Далі реалізуємо базову логіку та автентифікацію у системі. Наступним кроком є розробка клієнтського додатку.

5.3 Розробка клієнтського додатку

Для розробки клієнтської частини системи будо обрано мову JavaScript, а у якості фреймворку веб-додатків буде слугувати VUE.js та Nuxt.js. Спочатку треба створити базову сторінку та сторінки автентифікації. Для цього будемо використовувати токен автентифікації, що повертає нам сервер після надання даних користувача.

```
<script>
export default {
 data: () => ({
  userData: {
   email: 'email@email.com',
   password: 'long-password',
   showPassword: false
  }
 }),
 methods: {
  async logInUser (userData) {
   try {
    await this.$auth.loginWith('local', {
     data: userData
    })
    console.log('notification successful')
   } catch (error) {
    console.log('notification unsuccessful')
   }
```

```
    console.log(this.$auth.user)
   }
  }
}
</script>
```

Далі треба реалізувати базові інструменти взаємодії з токеном у системі, а саме:

- Інформаційну таблицю токенів користувача

```
<v-card style="margin-top: 50px">
    <v-card-title>
      <h2>List of tokens</h2>
    </v-card-title>
    <v-divider />
    <v-data-table
      v-model="selectedToken"
      :headers="headers"
      :items="tokens"
      :items-per-page="5"
      :single-select="true"
      item-key="pk"
      show-select
      dense
    />
</v-card>

async fetchTokens () {
    this.item.user = this.$auth.$state.user.id
```

```
    const response = await this.$axios.$get(
      '/api/basetoken/'
    )
    this.tokens = response
    console.log(this.tokens)
  },
}
```

- Форму для створення нових токенів

```
<v-form v-model="valid" @submit.prevent="submit">
        <v-container>
          <v-row>
            <v-col cols="12" md="6" >
              <v-text-field
                v-model="item.tokenId"
                :rules="tokenIdRules"
                :counter="42"
                label="Token ID"
                required
              />
            </v-col>
            <v-col ="12" md="6" >
              <v-text-field
                v-model="item.address"
                :rules="addressRules"
                :counter="42"
                label="Ethereum adress"
                required
              />
```

```
        </v-col>
      </v-row>
      <v-row style="padding: 10px">
        <v-btn class="mr-4" type="submit" :disabled="!valid" >
          submit
        </v-btn>
        <v-btn @click="clear">
          clear
        </v-btn>
      </v-row>
    </v-container>
  </v-form>
```

- Форму для надання метаданих для токену

```
<v-form @submit.prevent="submitMeta">
    <v-container>
      <v-row>
        <v-col cols="12" md="4" >
          <v-text-field v-model="metadata.title"
            :counter="20"
            label="Title"
            required
          />
        </v-col>
        <v-col cols="12" md="4" >
          <v-text-field
            v-model="metadata.type"
            :counter="20"
            label="Type"
            required
          />
        </v-col>
```

```
        <v-col cols="12" md="4" >
          <v-text-field
            v-model="metadata.name"
            :counter="20"
            label="Name"
            required
          />
        </v-col>
      </v-row>
      <v-row>
        <v-col cols="12" md="6" >
          <v-text-field v-model="metadata.description"
            :counter="120"
            label="Description"
            required
          />
        </v-col>
        <v-col cols="12" md="6" >
          <v-text-field v-model="metadata.content"
            :counter="120"
            label="Content"
            required
          />
        </v-col>
      </v-row>
      <v-row style="padding: 10px">
        <v-btn class="mr-4" ="submit" >
          submit
        </v-btn>
        <v-btn @click="clearMeta">
          clear
        </v-btn>
      </v-row>
    </v-container>
  </v-form>
```

- Форму передачі токену на іншу Ethereum адресу

```
<v-form v-model="valid2" @submit.prevent="transfer">
      <v-container>
       <v-row>
        <v-col>
         <v-text-field
          v-model="transferTo"
          :rules="addressRules"
          :counter="42"
          label="Ethereum adress"
          required
         />
        </v-col>
       </v-row>
       <v-row style="padding: 10px">
        <v-btn
         class="mr-4"
         type="submit"
         :disabled="!valid2"
        >
         submit
        </v-btn>
        <v-btn @click="clear">
         clear
        </v-btn>
       </v-row>
      </v-container>
     </v-form>
```

5.4 Результат роботи

На виході ми маємо централізований клієнт-серверний додаток, який оперує моделлю сумісною зі стандартом невзаємозамінного токену у мережі ERC-721 та надає інструменти для роботи користувача з цією моделюю. На прикладі працюючої програми ми бачимо переваги використання централізованої системи – а саме швидкість проведення операцій, а також мінімальну ціну виконання таких операцій. Система допоможе зекономити ресурси і звернутись до використання блокчейну у випадку реальної необхідності.

ВИСНОВКИ

В даній роботі було проведено аналіз ідеї використання невзаємозамінних токенів для підтвердження права власності та виконано загальний огляд існуючих систем взаємодії з невзаємозамінними токенами. Проаналізовані існуючі стандарти невзаємозамінних токенів та вимоги до створення невзаємозамінних токенів та їх метаданих. Також виконана розробка та імплементація уніфікованої моделі невзаємозамінного токену. Розроблено інтерфейс для взаємодії з моделлю у вигляді клієнт-серверного застосунку. Метою даної роботи було створення системи для взаємодії з невзаємозамінними токенами у мережі ERC-721 та її імплементація у вигляді клієнт-серверного застосунку.

В ході роботи було розглянуто поняття взаємозамінного та невзаємозамінного активу. Розглянуті нюанси пов’язані з цим, а також сформульована ідея невзаємозамінного цифрового активу.

Було визначено проблеми, пов’язані з реалізацією ідеї невзаємозамінного активу та способи їх вирішення за допомогою блокчейну, а саме за допомогою ідеї невзаємозамінного токену для підтвердження прав володіння. Також сформульовано властивості блокчейну, що дозволяють реалізувати ідею невзаємозамінного токену. Це такі властивості, як стандартизація, сумісність, можливість торгівлі, ліквідність, незмінність і доказовий дефіцит, програмованість.

Також в ході роботи було розглянуто важливість стандартизації токенів для вирішення проблеми прав володіння, а також виконано огляд та порівняння найпопулярніших стандартів невзаємозамінних токенів, а саме стандарти ERC-721 та ERC-1155.

Було визначено важливість метаданих для реалізації невзаємозамінних токенів, а також сформульовані ідеї їх реалізації і зберігання.

За результатом роботи було розроблено та імплементовано уніфіковану модель невзаємозамінного токену. Розроблено інтерфейс для взаємодії з моделлю у вигляді клієнт-серверного застосунку. Як результат була розроблена централізована система для взаємодії з невзаємозамінними токенами

На виході ми маємо централізований клієнт-серверний додаток, який оперує моделлю сумісною зі стандартом невзаємозамінного токену у мережі ERC-721 та надає інструменти для роботи користувача з цією моделюю. На прикладі працюючої програми ми бачимо переваги використання централізованої системи – а саме швидкість проведення операцій, а також мінімальну ціну виконання таких операцій. Система допоможе зекономити ресурси і звернутись до використання блокчейну у випадку реальної необхідності.

СПИСОК ПОСИЛАНЬ

[1] "What are NFTs and why are they so popular?". MoneyWeek. Retrieved 2021-04-06.

[2] "CryptoKitties Mania Overwhelms Ethereum Network's Processing". Bloomberg.com. 2017-12-04. Retrieved 2021-04-07.

[3] De-Mattei, Shanti Escalante (2021-04-14). "Should You Worry About the Environmental Impact of Your NFTs?". Art News. Retrieved 2021-06-02.

[4] Garg, Priyeshu (2020-05-16). "Understanding Ethereum's Gas and Transaction Fees". Crypto Briefing.

[5] Chevet, Sylve (2018-05-10). "Blockchain Technology and Non-Fungible Tokens: Reshaping Value Chains in Creative Industries". Rochester, NY. SSRN 3212662.

[6] Di Liscia, Valentina (2021-04-05). "Does Carbon Offsetting Really Address the NFT Ecological Dilemma?". Hypoallergic. Retrieved 2021-04-21.

[7] Howson, Peter. "NFTs: why digital art has such a massive carbon footprint". The Conversation. Retrieved 2021-04-06.

[8] Cuen, Leigh (2021-03-21). "The debate about cryptocurrency and energy consumption". TechCrunch.

[9] Reyburn, Scott (2021-03-30). "Art's NFT Question: Next Frontier in Trading, or a New Form of Tulip?". The New York Times. ISSN 0362-4331. Retrieved 2021-05-03.

[10] "EIP-721: ERC-721 Non-Fungible Token Standard". Ethereum Improvement Proposals. Retrieved 2021-04-05.

[11] "EIP-1155: ERC-1155 Multi Token Standard". Ethereum Improvement Proposals. Retrieved 2021-04-05.

[12] VUE.js documentation - https://ru.vuejs.org/v2/guide/

[13] Django documentation - https://docs.djangoproject.com/en/3.2/

[14] "NFTs are both priceless and worthless". Engadget. 2021-03-11. Retrieved 2021-04-09.
[15] Samarbakhsh, Laleh (March 17, 2021). "What are NFTs and why are people paying millions for them?". The Conversation. Retrieved March 31, 2021.
[16] Boscovic, Dragan. "How nonfungible tokens work and where they get their value – a cryptocurrency expert explains NFTs". The Conversation. Retrieved 2021-04-08.