COMPUTER SYSTEMS AND NETWORKS

Vol. 4, No. 1, 2022

DOI: https://doi.org/10.23939/csn2022.01.047

УДК 004.75/.62

ДЕЦЕНТРАЛІЗОВАНИЙ ПРОГРАМНИЙ СЕРВІС СМАРТ-КОНТРАКТУ З ВИКОРИСТАННЯМ НЕВЗАЄМОЗАМІННИХ ТОКЕНІВ БЛОКЧЕЙНУ ETHEREUM

А. О. Ігнатович, А. В. Янчинський

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин
E-mail: andrii.yanchynskyi.mkisp.2021@lpnu.ua; anatolii.o.ihnatovych@lpnu.ua

У роботі проаналізовано децентралізовані технології блокчейну Ethereum, за принципами яких запропоновано рішення децентралізованого програмного сервісу з використанням смарт-контракту, реалізованого на спеціалізованій мові програмування Solidity. Цей контракт виступає у якості одного з елементів сервер-клієнтського додатку і є серверною частиною для обробки методів взаємодії із блокчейном Ethereum. Методи містять набір різноманітних функцій, зокрема й для взаємодії із колекцією невзаємозамінних токенів. Метадані, які описують цифровий продукт (невзаємозамінні токени) напряму пов'язані із блокчейном та знаходяться на децентралізованому сховищі задля їх якомога більш надійного та довговічного існування. Блокчейн Ethereum окрім інструментів для розробки надає стабільну підтримку розробникам та користувачам завдяки популярній парадигмі децентралізації, а актуальність та постійний розвиток технологій ведуть до зацікавленості користувачів у продуктах, створених на їхній основі. Невзаємозамінні токени можна використовувати у якості криптовалютної одиниці, а сам децентралізований додаток – як платформу для збору коштів задля певної мети.

Досліджено методи створення децентралізованих програмних сервісів з використанням смарт-контрактів невзаємозамінних токенів. Обґрунтовано вибір основних вузлів децентралізованого додатку. Запропоновано деталізовану функціональну схему роботи базових методів смарт-контракту разом зі схемою, яка описує всі функціональні вузли децентралізованого додатку загалом. Також продемонстровано результати взаємодії клієнтської частини програмного сервісу зі смарт-контрактом блокчейну Ethereum.

Ключові слова: блокчейн, Ethereum, DApps, децентралізований додаток, децентралізоване сховище, IPFS, смарт-контракт, невзаємозамінний токен, NFT, цифровий продукт, Solidity, криптовалюта.

Вступ

Кожен новий реалізований концепт покликаний зробити людське життя кращим, принести користь або ж бути створеним заради прагматичних власних чи суспільних цілей. Сьогодні технології блокчейну розвиваються у неймовірному темпі, і з'являється все більше можливостей для розроблення різноманітних речей на їх основі. Децентралізовані додатки (скор. англ. DApps) не є виключенням, і займають у сучасному світі технологій все більше місця, оскільки за їхнім концептом ховається майбутнє нового уявлення про глобальну мережу та прозорий користувацький досвід. Децентралізовані програми – програми або ж цифрові додатки, чи програмні сервіси, які працюють у блокчейні або в одноранговій мережі комп'ютерів. Ці програми значною мірою

децентралізовані, і не можуть контролюватися одним органом. Хоча, подібно до інших програмних додатків щодо візуальних аспектів, вони підтримують P2P (Peer-to-Peer) архітектуру, що робить їх різними. Децентралізовані додатки мають певні основні характеристики, які визначають спосіб їхньої роботи. По-перше, вони мають відкритий вихідний код, що означає, що кожна зміна, внесена до децентралізованої програми, спочатку приймається консенсусом великої кількості користувачів. Тому кодова база програми доступна для оцінки всім користувачам. Крім того, вони мають особливість – забезпечують децентралізоване зберігання, яке використовує децентралізовані блоки для зберігання даних. Вони зберігаються в мережі блокчейну, і їх виконання також відбувається в ній (зазвичай Ethereum). Подібні програми створюються для різних областей, таких як фінанси, ігри, соціальні мережі, благодійні платформи чи навіть для прозорого блокчейн-голосування. В принципі, можливо поєднати технології децентралізованих додатків та NFT (англ. non-fungible token, в перекладі з англ. – невзаємозамінний токен) з будь-якою сферою сучасного життя, і блокчейн Ethereum є поширеним вибором платформи для їх створення.

Невзаємозамінний токен – це тип криптовалюти, отриманий за допомогою смарт-контрактів Ethereum та інших блокчейнів. NFT було вперше запропоновано в Ethereum Improvement Proposals (EIP)-721 і далі розвинуто в EIP-1155. NFT відрізняється від класичних криптовалют, таких як біткойн, своїми властивостями [1]. Біткойн – стандартна монета, у якій усі монети еквівалентні та нерозрізнені. Тим часом NFT є унікальним, і його не можна обмінювати подібним на подібне (еквівалентно, незамінно), що робить його придатним для ідентифікації чогось або когось унікальним способом. Якщо говорити конкретно, використовуючи NFT на смарт-контрактах, творець може легко довести існування та право власності на цифрові активи [1] у вигляді відео, зображень, мистецтва, квитків на події тощо.

У цій науково-технічній роботі буде розглянуто один із варіантів взаємодії з блокчейном Ethereum згідно з концепцією децентралізованого Інтернету, а саме – створення децентралізованого додатку з використанням смарт-контракту в межах блокчейну в якості серверної частини. Оскільки зараз робота зі смарт-контрактами є новою, і спеціалістів у ній не так багато, варто дослідити її одними з перших, щоб у майбутньому використати набуті знання для створення більш складних речей.

Смарт-контракт дійсно можна описати як серверну частину клієнт-серверного вебдодатку, власне як обробку реалізованого алгоритмічного функціоналу за допомогою об’єктноорієнтованої мови програмування Solidity, оскільки після його розробки смарт-контракт завантажується на блокчейн у вигляді байт-коду. Під час роботи буде розглянуто реалізовані методи для взаємодії з випадково згенерованою колекцію NFT, методи розробки для досягнення кращого результату. Якщо ж говорити про доцільність, то будь-який концепт децентралізованого додатку є актуальним як мінімум для власного інтересу та користувачів, які будуть зацікавлені у його використанні. Повертаючись до NFT, доцільно створити вебмагазин з можливістю покупок цифрових продуктів. Ідея не є новою в межах децентралізованих технологій, тому необхідно підійти до вирішення завдання з власним баченням концепту та додати певні нові методи взаємодії користувача з проєктом, окрім цього, подібні проєкти можна використати в якості платформи для збору коштів на благодійність. Тим не менш, має місце суб’єктивна думка, що в нашому швидкоплинному суспільстві варто популяризувати ідеї концепції Web3 задля створення справжньої незалежної мережі Інтернет.

Враховуючи викладене вище, можна вважати, що дослідження щодо створення децентралізованих програмних сервісів смарт-контрактів є важливими та актуальними.

Огляд літературних джерел

Сьогодні використовується відносно велика кількість різних децентралізованих програмних систем. Беручи до уваги емпіричні дослідження [2] та надану на сьогодні інформацію із вебсервісу State of the DApps [2], це більше чотирьох тисяч децентралізованих додатків на всіх існуючих відкритих блокчейнах. Програмні системи реалізовують різну концептуальну ідею чи виконують

певний програмний алгоритм із застосуванням Ethereum [3] смарт-контрактів [3], базованих на Javascript-подібній об'єктоорієнтованій мові програмування Solidity [3], або ж використовують інші програмні засоби, коли йдеться про відкриті блокчейни. Більшість із них так чи інакше пов'язані з токенізацією процесів [1], зокрема з унікальними (невзаємозамінними) токенами [1] стандарту ERC-721 та ERC-1155 [1]. Важливою частиною екосистеми блокчейну Ethereum є вебсервіс Etherscan [4] для вивчення та деталізації даних, які знаходяться в блокчейні. Цей тип вебсайтів відомий як Block Explorer, оскільки вони дозволяють досліджувати вміст блоків. Блок містить дані всіх транзакцій, які були виконанні протягом виділеного часу. Провідник блоків [4] дає змогу переглядати події, які були випущені під час виконання смарт-контракту, а також такі речі, як сума, яку було сплачено за газ [4], кількість транзакцій ефіру, і найголовніше – можливість взаємодії зі смарт-контрактами. Тобто, для зручності розробки смарт-контракту в мережі блокчейну існують інструменти [3, 4], які дозволяють перевірити кожну функцію за допомогою звернення до самого контракту через спеціальний сервіс.

Завдяки концепту Turing-complete [3] в смарт-контрактах розробники можуть впроваджувати нескінченні програми в них, що призводить до поломки вузлів, які їх виконують. Щоб уникнути цього, Ethereum використовує газовий механізм [3, 5]: кожен код операції в байт-коді коштуватиме деяку кількість ефіру (ETH, криптовалюта, яку випускає Ethereum), подібний механізм використовують інші відкриті блокчейни. За допомогою цього механізму Ethereum об'єднує запити контрактів і транзакції ETH у спільні транзакції [3]. Запитувач повинен заплатити газ за виконання, але лише вузол, який генерує блок, і містить сам запит, отримає плату за виконання. Розгортання смарт-контракту розглядається як транзакція і також вимагає оплати. При створенні децентралізованого програмного сервісу слід брати до уваги оптимізацію [5] газового механізму в блокчейні Ethereum. Якість та кількість програмного коду смарт-контракту впливає на результат швидкості обробки даних транзакції [5] та суми, яку сплатить користувач за її виконання. Існують зручні засоби для написання [2, 6] і оптимізації програмного коду смарт-контракту, одним з них є середовище розробки Remix [6], яке дозволяє емулювати транзакції в його межах.

Задля постійного доступу до контенту (даних) цифрового продукту, пов'язаного із невзаємозамінними токенами смарт-контракту, в довгостроковій перспективі існують децентралізовані сховища. Найпопулярнішим можна вважати IPFS (The Interplanetary File System) [7] – це набір підпротоколів і проєкт, керований ProtocolLabs.IPFS, що має на меті підвищити ефективність Інтернету та зробити мережу більш децентралізованою [7] і стійкою.

Для побудови клієнтської частини децентралізованого додатку доцільно застосовувати криптогаманець MetaMask [8], оскільки підтверджена у ньому адреса користувача грає роль даних для автентифікації у програмних блокчейн-сервісах. Для обробки даних запитами з клієнтської частини необхідний кластерний міст [9], цю можливість надає API (Application Programming Interface) Infura [9], яка слугує зв'язком між програмами та деякими методами смарт-контракту на блокчейні Ethereum.

Постановка завдання

Метою дослідження є аналіз децентралізованих блокчейн-технологій, схематична демонстрація створення децентралізованого додатку задля обґрунтування методів створення сервер-клієнтського додатку за децентралізованою концепцію, де серверна частина розробляється як смарт-контракт, який знаходиться на блокчейні Ethereum, описуючи методи для взаємодії вебсайту із контрактом, зокрема і невзаємозамінними токенами.

Результати дослідження

Основним функціональним завданням запропонованого децентралізованого програмного сервісу буде можливість придбання невзаємозамінних токенів з можливістю їх подальшої взаємодії зі смарт-контрактом додатку, включаючи різні реалізовані методи якими може скористатися

користувач в межах функцій серверної частини розміщеної в блокчейн-мережі Ethereum. Клієнтська частина має забезпечувати можливість відображення поточних даних цифрову власність із зручним інтерфейсом взаємодії, для цього необхідно реалізувати інтеграцію криптогаманця MetaMask в якості авторизації користувача [8]. Розроблення такого програмного сервісу доцільно виконувати із використанням концепції децентралізованих додатків із використанням смарт-контрактів у блокчейні Ethereum [3,4]. Вибір цієї концепції обумовлений наступними чинниками:

Для створення смарт-контракту для власного децентралізованого додатку доцільно вибрати блокчейн Ethereum та об'єктно-орієнтовану мову програмування Solidity, посилаючись на наведені вище дані про популярність самого блокчейну та найбільший розвиток технологій для розробки.

Децентралізовані додатки мають все ширше застосування. Під децентралізованим [2] додатком розуміють програми, які здебільшого або повністю децентралізовані. Варто розглянути всі можливі аспекти [2] конкретної програми, які можуть бути децентралізованими:

- серверне програмне забезпечення (логіка програми);
- інтерфейсне програмне забезпечення;
- зберігання даних;
- комунікація;
- резолюція імен.

Кожен із цих аспектів може бути частково централізованим або дещо децентралізованим. Наприклад, інтерфейс можна розробити як вебпрограму, яка працює на централізованому сервері, або як мобільний додаток, який працює на певному пристрої. Сервер і сховище можуть бути на приватних серверах чи власних базах даних, або ж можна використовувати смарт-контракт і P2P-сховище [2]. Створення децентралізованого додатку має багато більше переваг порівняно з типовою централізованою архітектурою. В першу чергу – стійкість, оскільки бізнес-логіка контролюється смарт-контрактом, серверна частина DApp буде повністю розташована та керована на певному блокчейні. На відміну від програм, розгорнутих на централізованому сервері, децентралізований додаток не матиме простоїв, і буде залишатися доступним, поки платформа працює. Також варто згадати такий показник, як прозорість [2]: розташування на блокчейні дозволяє кожному передивитися вихідний код і бути впевненим у функціоналі та намірах розробників; більше того, будь-яка взаємодія з DApp зберігатиметься назавжди на блокчейні.

Смарт-контракти є важливою частиною DApps, і їх можна визначити як комп'ютерні протоколи, які цифровим способом сприяють, перевіряють і забезпечують виконання контрактів, укладених між двома або більше сторонами в блокчейні [3, 4]. По-перше, програмний код смарт-контракту буде записаний і перевірений на блокчейні, завдяки чому контракт буде захищеним від втручання. По-друге, виконання смарт-контракту здійснюється серед анонімних окремих вузлів без централізованого контролю та координації сторонніх органів. По-третє, смарт-контракт, як і інтелектуальний агент [3], може мати власні криптовалюти або інші цифрові активи та передавати їх, коли спрацьовують попередньо визначені умови. Існує декілька відкритих блокчейнів, які надають можливість створювати власні контракти, але уперше вони з'явилися в мережі блокчейну Ethereum. Один із декількох стандартів смарт-контрактів блокчейну Ethereum надає можливість використання невзаємозамінних токенів. ERC-721 [1, 2, 3] був першим стандартом для NFT, розробленим у Ethereum, з безкоштовним розповсюдженням та відкритим вихідним кодом. Стандарт являється інтерфейсом, який смарт-контракт повинен реалізувати, щоб мати можливість передавати та керувати NFT. Кожен токен ERC-721 має унікальні властивості та інший ідентифікатор токена. Токени ERC-721 містять інформацію про власника, список затверджених адрес, функцію передачі, яка реалізує передачу токенів від власника до покупця та інші корисні функції. У ERC-721 смарт-контракти можуть групувати токени з однаковою конфігурацією, і кожен токен має різні властивості, тому ERC-721 не підтримує взаємозамінні токени.

Задля зв'язку клієнтської частини програмного сервісу із серверною використовується кластерний міст. Щоб клієнтська частина мала зв'язок зі смарт-контрактом, необхідно реалізувати

зчитування функцій контракту клієнтом у доступному вигляді та реалізувати міст, який якраз буде створювати зв'язок із блокчейном. Для цього необхідно використати API Infura – це платформа, яка дозволяє DApp швидко отримати доступ до Ethereum без необхідності запускати вузли Ethereum локально [9]. Infura є провайдером Web3, за яким стоїть кластер вузлів API зі збалансованим навантаженням [9]. Перевага використання в тому, що ніколи не доведеться турбуватися про збій підключених вузлів.

Для реалізація клієнтської частини децентралізованого програмного сервісу, який буде використовувати смарт-контракт блокчейну Ethereum в якості серверної частини, найбільш доцільно використати фреймворк Next.js об'єктно-орієнтованої прототипної мови програмування JavaScript, який дозволяє створювати надшвидкі та надзвичайно зручні статичні вебсайти, а також вебдодатки за допомогою бібліотеки React. Завдяки автоматичній статичній оптимізації цього фреймворку «статичний» і «динамічний» концепт сайтів стали одним цілим, ця функція дозволяє Next.js створювати гібридні програми, які містять як відтворені на стороні сервера, так і статично згенеровані сторінки. Також розробники даного фреймворку надають хмарну платформу Vercel для гібридних сайтів і безсерверних функцій, це дає можливість розміщувати вебсайти та вебсервіси, які миттєво розгортаються, автоматично масштабуються та не потребують нагляду, і все без конфігурації, тобто оптимальне рішення для проєкту, в якого серверна частина знаходиться на блокчейні.

В якості даних цифрових продуктів може бути що завгодно: віртуальні предмети колекціонування, зображення, квитки, музика, певні дані, і в деяких випадках – фізичні предмети. Все залежить від того, яка мета була поставлена при розробці повноцінного децентралізованого додатку. Оптимальним рішенням є створення колекції зображень, кожному з яких будуть відповідати атрибути, які характеризують їх окремо, додаючи унікальності та цінності для користувача. Цю колекцію можна використати як мотивацію для збору коштів на благодійність або задля розвитку проєкту. Насправді, ідея із використанням зображень досить розповсюджена, але її можна виконати по-різному в межах реалізації взаємодії з цими зображеннями.

Для збереження даних невзаємозамінних токенів, пов'язаних зі смарт-контрактом, існує безліч класичних методів: віддалені або локальні серверні бази даних, різноманітні централізовані сервіси, хмарні сховища. Цього недостатньо для збереження контенту на довготривалий період, який вираховується роками, оскільки взаємодія додатка з цим контентом не буде вважатися повністю децентралізованою. Зокрема, використання локальної чи віддаленої бази даних може бути необхідним на певній стадії розгортання проєкту. Задля постійного доступу до контенту у довгостроковій перспективі існують децентралізовані сховища [7]. Як було згадано вище, одним із перших та найбільш популярний є IPFS. IPFS використовує адресацію [7] на основі вмісту, де він адресується не через розташування, а через власне сам вміст. Те, як IPFS зберігає та адресує дані за допомогою властивостей дедуплікації, забезпечує ефективне зберігання даних, через протокол можна децентралізовано зберігати файли та обмінюватися ними, підвищуючи стійкість до цензури для їх вмісту. IPFS можна використовувати для розгортання вебсайтів, створюючи розподілену мережу [7]. Він використовується як служба зберігання, що доповнює блокчейни, забезпечуючи роботу різних додатків поверх протоколу.

Усі зазначені вище технології та концепції складаються в одну суцільну картину, яка відображає принцип роботи децентралізованого програмного сервісу в блокчейні Ethereum.

Наведені особливості програмного сервісу є важливими аргументами на користь того, що вони концептуально можуть бути застосовані при побудові децентралізованого додатку смарт-контракту невзаємозамінних токенів для реалізації платформи для збору коштів або ж інших корисних цілей за допомогою унікальних цифрових криптовалютних одиниць, які власне описує смарт-контракт.

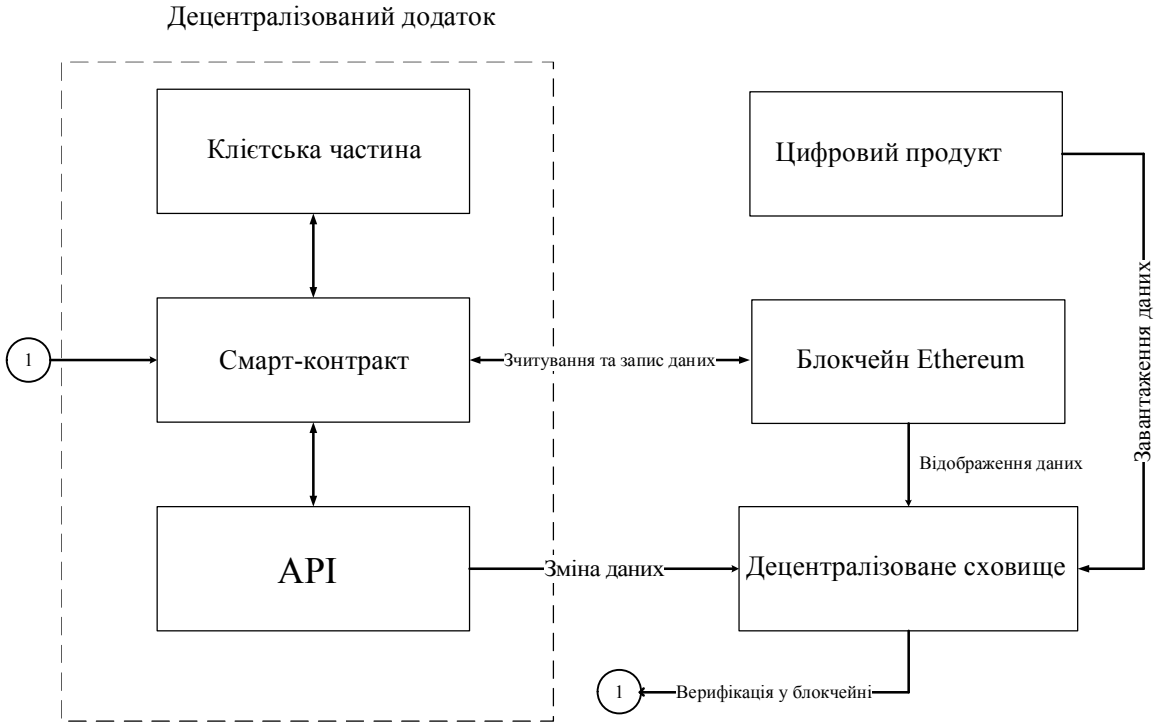

Рис. 1. Структурна схема роботи децентралізованого програмного сервісу смарт-контракту із використанням невзаємозамінних токенів у блокчейні Ethereum

Принципи створення смарт-контракту невзаємозамінних токенів децентралізованого програмного сервісу

Перш ніж приступити до розробки смарт-контракту, потрібно визначатися з інтегрованим середовищем. Найбільш зручним можна вважати Remix IDE через його можливості відлагодження методів контракту в межах самого середовища, оскільки воно має підтримку емуляції [6] виконання транзакцій блокчейну. Середовище не потребує налаштування, забезпечує швидкий цикл розробки та має багатий набір плагінів [9] з інтуїтивно зрозумілим графічним інтерфейсом. IDE доступне в двох варіантах (вебпрограма або настільна програма) і як розширення VSCode [9].

Ethereum є першим блокчейном, що надає Turing-complete [3] мову програмування для розробки смарт-контрактів. На сьогодні, Solidity є Javascript-подібною об'єктно-орієнтованою мовою програмування, яка найбільш популярна в спільноті Ethereum. Смарт-контракт розробляється мовою Solidity, компілюється в байт-код [3], а потім розгортається в блокчейні. Щоб прийняти скомпільовані коди, Ethereum вводить новий тип користувачів, які називають контрактивними. Власник контракту має ті ж функції, що й окремий користувач, але має також певні привілеї над контролем всього смарт-контракту. Усі дані власника є відкритими, в тому числі його транзакції та байт-код смарт-контракту, це дає можливість користувачу самому оцінити якісь реалізації контракту [3] або звернутися до когось з проханням провести дослідження реалізованого смарт-контракту.

При розробці смарт-контракту варто брати до уваги, що кількість написаного тексту програми та її логіка впливає на вартість комісійної плати (газу) для завантаження контракту в блокчейн Ethereum, оскільки в момент інтеграції контракту текст програми перетворюється в байт-код, який є мовою асемблера, що складається з кількох кодів операцій (інструкцій низького рівня). Найважливіше те, що в блокчейні зберігається лише байт-код смарт-контракту [3, 5]. Крім цього, логіка програмного коду розповсюджується на вартість транзакцій взаємодії користувача та власника із самим контрактом.

Із цього принципу серверна частина додатку має бути якомога більш лаконічна, без зайвого програмного коду. Під час дослідницької роботи розроблений додаток якраз наслідує цей принцип та виконує найбільш важливі функції, присутні для користувача та власника. Смарт-контракт стандарту ERC-721 реалізує можливість для користувача придбати цифровий продукт та взаємодіяти з ним різноманітним чином, наприклад, за бажанням користувач може змінити назву невзаємозамінного токену, звісно ж, якщо цей метод реалізований в самому смарт-контракті. Після транзакції введений текст назавжди лишиться в межах блокчейну та закріпиться за ідентифікатором токена, що придасть унікальності цифровому продукту.

Як було зазначено раніше, при відлагодженні варто користуватися можливостями оглядача блоків Etherscan, задля того, щоб бути впевненим у правильному виконанні всіх функцій. Використовуючи тестову мережу, можна уникнути плати за виконання транзакцій, але спершу необхідно завантажити контракт саме в неї, зокрема, весь проєкт може працювати в межах тестової мережі [5], але тоді токени в ньому, відповідно, не будуть мати вартості.

Базовий смарт-контракт стандарту ERC-721 децентралізованого програмного сервісу представляє собою визначену сукупність елементів, які об'єднані між собою у програмному коді, виконують окремі вузько спеціалізовані дії, орієнтовані на вирішення певних задач. Обов'язковим центральним елементом системи є можливість передавати та мати право власності на невзаємозамінний токен та узгодження кожної транзакції за сигнатурою адресу криптовалютного гаманця користувача.

Детальні особливості кожної частини смарт-контракту можна розділити на блоки. Стандарт ERC-721 визначає два різні підходи до даних про власність. По-перше, існує метод ownerOf, який повертає власника певного активу за допомогою ідентифікатора маркера сигнатури, інакше кажучи, відбувається запит зіставлення ідентифікаторів токенів із поточним власником, яких підтримує контракт. Крім того, існує метод balanceOf, який повертає загальну кількість NFT, що належать певному обліковому запису. Цей метод комбінується із функціями перерахування токенів, власне методом повернення їх загальної кількості та можливістю відслідковувати певний невзаємозамінний токен за унікальним ідентифікатором. Варто виділити окремо метадані, які задаються на початку розробки у виді назви та скороченого символу невзаємозамінних токенів, та після успішного завантаження контракту в блокчейн Ethereum необхідно вказати посилання на кожен атрибут токена, наприклад, IPFS-хеш або HTTP(S)-адресу, щоб зовнішня програма могла знайти більше інформації про токен за допомогою логічних операцій. Такі посилання – це дані про дані, або метадані, і вони придають візуальної або атрибутивної унікальності кожному токену.

Найпростіший спосіб ініціювати передачу токена – використовувати метод transferFrom [1, 3]. Цей метод дозволяє абоненту передавати заданий ідентифікований токен від його поточного власника на іншу адресу. Звичайно, відправник повідомлення повинен бути авторизований для виконання передачі – поточний власник токена завжди авторизований, але бувають також інші варіанти. Існує ризик, пов'язаний із використанням цього методу, припустімо, він використовується для передачі токена на певну адресу, а отримувачем є смарт-контракт. Потім NFT буде передано в смарт-контракт, і лише транзакція, що походить від смарт-контракту, може перенести токен на інший обліковий запис. Якщо смарт-контракт не підготовлений для цього, тобто, якщо він не має методу для ініціювання такої передачі, NFT буде втрачено назавжди. Щоб принаймні частково пом'якшити цей ризик, стандарт ERC-721 заохочує контракти, які можуть керувати невзаємозамінними токенами, впровадити маркерний інтерфейс. Щоб обмежити негарантовані транзакції, стандарт пропонує метод safeTransferFrom [1, 3]. Цей спосіб дуже схожий на звичайний переказ, за двома винятками. По-перше, передбачається перевірити, чи адреса отримання є смарт-контрактом (або, що не зовсім те саме, має ненульовий код). Якщо так, він спробує викликати метод onERC721Received [1, 3] цільового контракту, який має повернути визначену послідовність із чотирьох «магічних байтів». Якщо цільовий контракт не реалізує метод або метод існує, але повертає інше значення, передача не вдасться. По-друге, метод safeTransferFrom необов'язково приймає поле даних, яке може містити довільну послідовність байтів, яке передається до onERC721Received [3]

одержувача, цільовий контракт може, наприклад, реєструвати ці дані або виконувати деякі інші операції, як-от оновлення балансу та збереження переданих даних як посилання. Іншими словами, метод зручно використовувати в якості підтвердження даних отримувача.

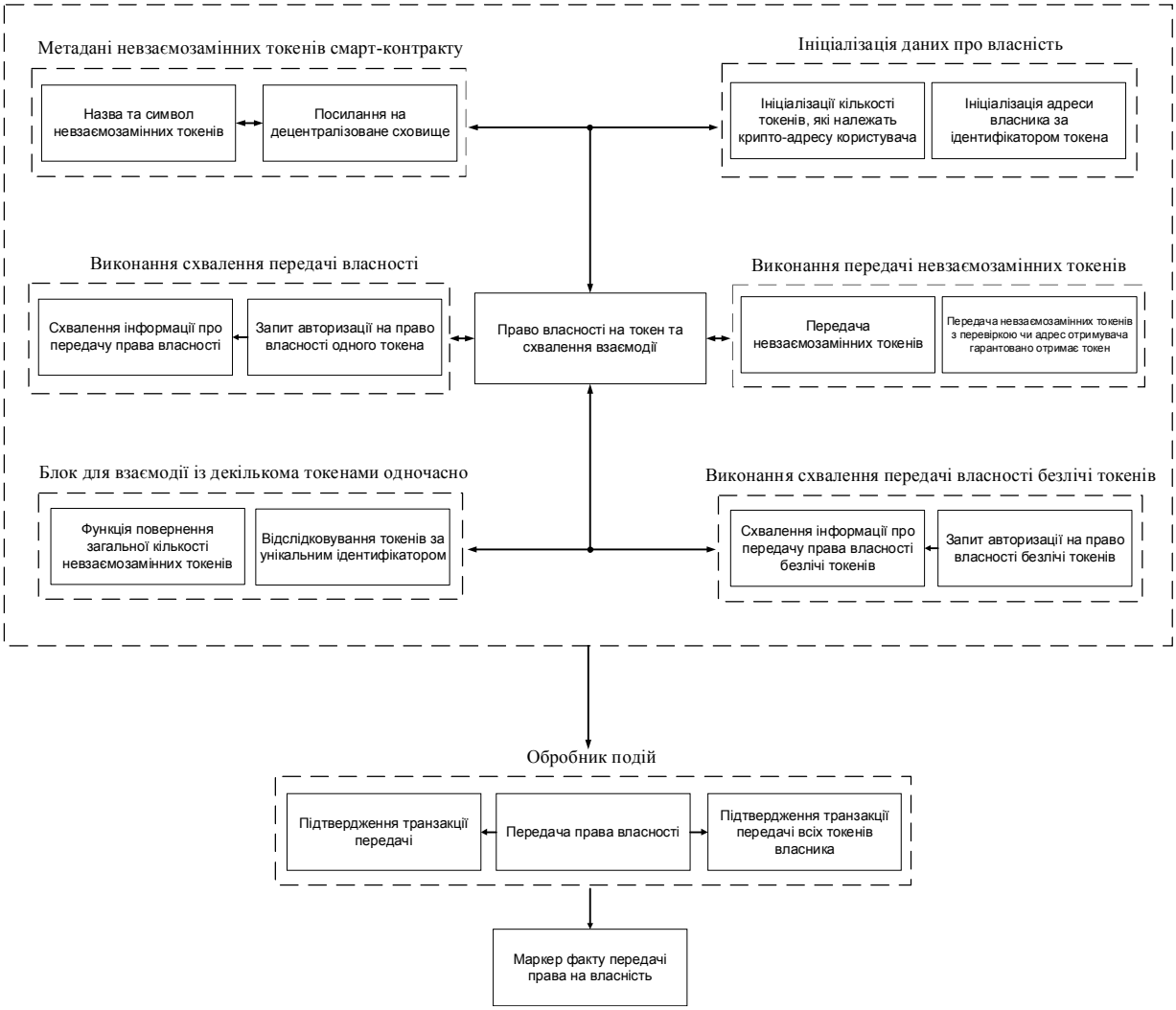

Рис. 2. Функціональна схема базового набору методів смарт-контракту невзаємозамінних токенів у блокчейні Ethereum

Власник невзаємозамінного токена завжди має право ініціювати переказ. На додаток до цього, також підтримується схема зняття криптовалютного балансу. Насправді існує метод схвалення, який власник NFT може використати, щоб дозволити комусь іншому передати невзаємозамінний токен. Схвалення також можна явно скасовувати або ж, якщо право власності на NFT змінюється воно скидається автоматично. На додаток до цього явного схвалення, яке стосується конкретного ідентифікатора невзаємозамінного токена, також можна зареєструвати іншу адресу як уповноважену здійснювати будь-які перекази від вашого імені, тобто як оператора. Після визначення оператор може передати будь-який токен, яким володіє користувач.

В кінці виконання певних методів стандарт визначає подій, які мають виникати, коли здійснюється передача, надається або скасовується схвалення, коли оператор найменовується або видаляється.

Нижче наведено власний варіант реалізації програмного сервісу смарт-контракту невзаємозамінних токенів у блокчейні Ethereum для експериментальної демонстрації ефективності запропонованих застосованих принципів створення.

Реалізація та експериментальна перевірка децентралізованого програмного сервісу

Посилаючись на опис смарт-контракту та методів розробки децентралізованого програмного сервісу, можна впевнено сказати, що концепт створення подібних програм є перспективним та ефективним для створення дійсно децентралізованого додатку.

Як вже було раніше зазначено у цьому, децентралізований додаток для взаємодії із смарт-контрактом невзаємозамінних токенів розділяється на дві частини: клієнтську та серверну. Принцип побудови децентралізованого додатку краще всього зображувати схематично. Використання додатку користувачем чи власником супроводжується перевіркою на право власності цифрового продукту, коли дані збігаються та верифікуються на стороні серверної частини, то надається можливість взаємодії із невзаємозамінними токенами. В випадку, коли цифровий продукт не належить нікому, то користувач виступає у ролі покупця, який має оцифрувати токен оплативши вказану в смарт-контракті суму та комісійну плату за одиницю невзаємозамінного токену, після чого передається право власності на адрес криптовалютного гаманця. Всі транзакції відбуваються із згоди консенсусного вузла блокчейну Ethereum. У випадку, коли відбувається взаємодія із смарт-контрактом, транзакція записує нові дані у блокчейн. Серверна частина взаємодії із блокчейном використовує кластерний міст Infura який дозволяє виконати операцію. Відповідно API додатку реагує на дії користувача та змінює метадані на децентралізованому сховищі.

Дані невзаємозамінних токенів, які закріплені посиланням на децентралізоване сховище мають знаходитися у ньому в ієрархії де відображення кожного токена закріплено за метаданими з певними властивостями. Ці файли необхідно закріпити за допомогою будь-якого хмарного напів-децентралізованого сховища, як-от Pinata Service [1]. Це надасть безперервний доступ до даних невзаємозамінних токенів, навіть якщо децентралізований вузол сховища IPFS буде виключений.

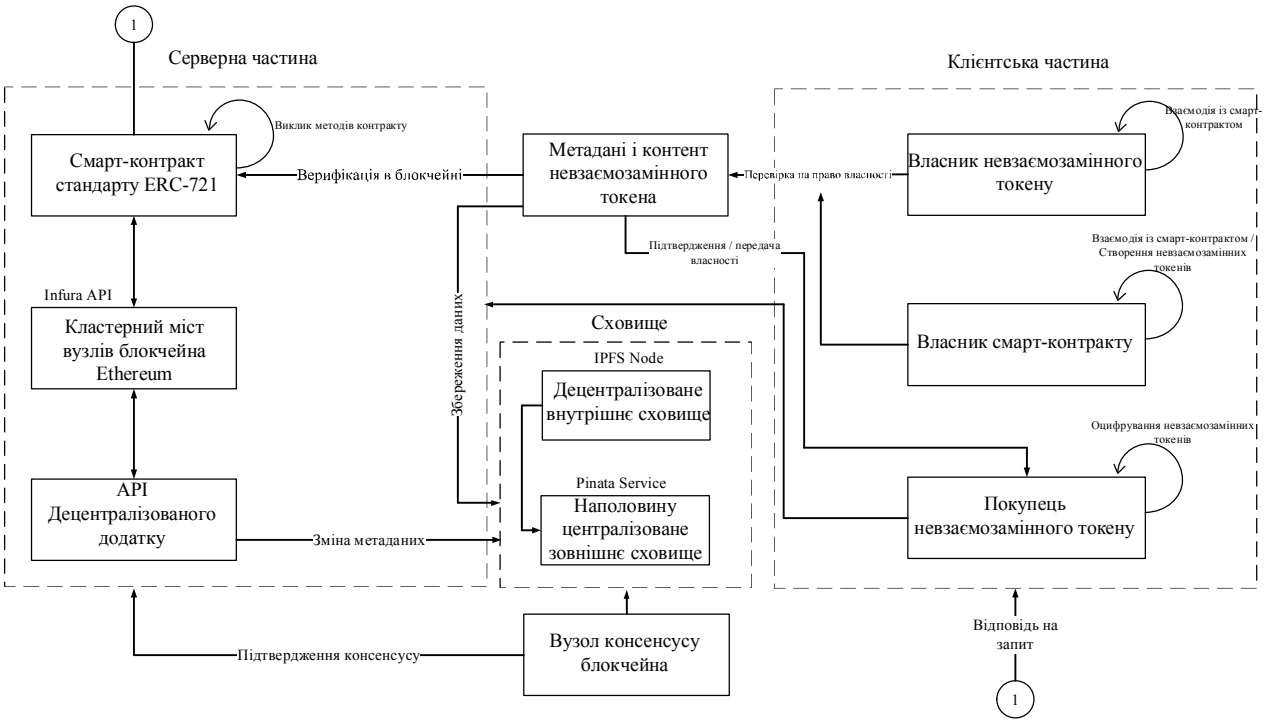

Рис. 3. Функціональна схема роботи децентралізованого програмного сервісу смарт-контракту із використанням невзаємозамінних токенів у блокчейні Ethereum

Клієнтська частина складається з гібридного вебінтерфейсу, який дозволяє користувачу дізнатися всю необхідну інформацію про цифровий продукт та взаємодіяти з ним після покупки за допомогою зв’язку із смарт-контрактом, оскільки контракт використовується як серверна частина

всього додатку. Незалежно від тематики та цілей децентралізованого додатку інтерфейс має бути приємним для ока та викликати довіру, тому варто надати як можна більше інформації та чітко сформулювати мету проєкту.

На відміну від логіки серверної частини, яка вимагає від розробника розуміння EVM (Ethereum Virtual Machine) [3] і нових мов, таких як Solidity, клієнтський інтерфейс DApp може використовувати стандартні вебтехнології [3] (HTML, CSS, JavaScript тощо). Це дозволяє традиційному веброзробнику використовувати знайомі інструменти, бібліотеки та фреймворки. Взаємодія з Ethereum, наприклад, підпис повідомлень, надсилання транзакцій і керування ключами, часто здійснюється через веббраузер за допомогою такого розширення, як MetaMask.

В якості перевірки працездатності програмного сервісу доцільно звернути увагу на вірне виконання транзакцій всіх реалізованих методів, використовуючи клієнтську частину, яка зв'язується зі смарт-контрактом блокчейну Ethereum. Для демонстрації викликатиметься метод серверної частини, який змінює імена невзаємозамінних токенів. Після введення даних і їх підтвердження викликається вікно криптогаманця. У разі неправильного формату імені чи відсутності в браузері розширення MetaMask користувач буде попереджений повідомленням про відповідну помилку.

Рис. 4. Інтерфейс взаємодії децентралізованого програмного сервісу із смарт-контрактом

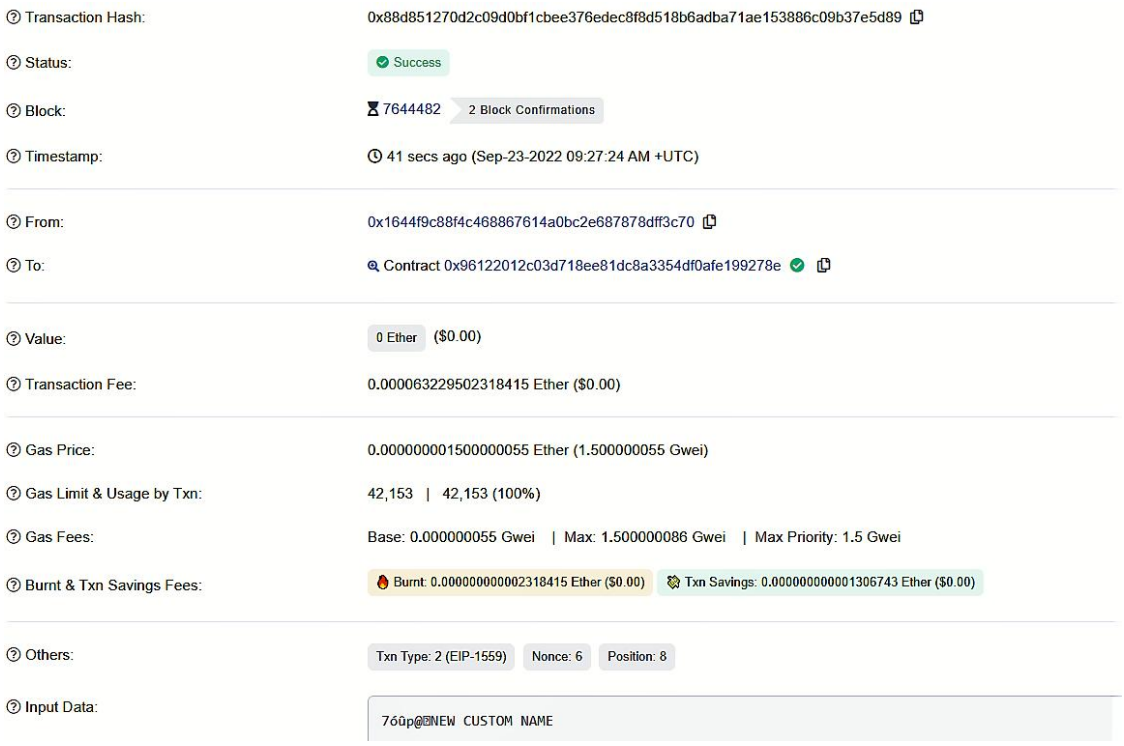

Рис. 5. Дані оброблення транзакції відображені на вебсервісі Etherscan

Дані виконання кожного обробленого запиту, як було вказано раніше, можна переглянути за допомогою вебсервісу Etherscan.

Проаналізувавши результати виконання методу смарт-контракту нижче, видно, що нове ім'я невзаємозамінного токена було успішно записане у виді байт-коду в блокчейн Ethereum. Це значить, що децентралізований програмний сервіс вірно обробив один із методів серверної частини, і кожен вузол додатку працездатний.

Якщо зробити висновки щодо перевірки програмного сервісу, яка була описана вище, можна впевнено вважати, що експериментально підтверджено ефективність та доцільність запропонованих рішень щодо принципів створення децентралізованих додатків невзаємозамінних токенів.

Висновки

У роботі наведено результати розроблення децентралізованого програмного сервісу смарт-контракту на блокчейні Ethereum, який реалізує методи для взаємодії користувача через клієнтську частину зі смарт-контрактом невзаємозамінних токенів. Проаналізовано концепцію децентралізованих блокчейн-технологій, за якими було розроблено децентралізований додаток. Виконано аналіз даних про сучасний стан та концепцію побудови подібних децентралізованих програмних сервісів.

Основним вузлом сервісу вибрано блокчейн Ethereum, на якому розміщено байт-код розробленого смарт-контракту з функціональними методами. В якості збереження даних цифрового продукту було обрано децентралізоване сховище IPFS з використанням напівдецентралізованого сервісу Pinata для безперервного доступу до метаданих невзаємозамінних токенів. Візуально результати опрацювання роботи методів смарт-контракту можна спостерігати за допомогою клієнтської частини, яка дозволяє придбати невзаємозамінний токен та надає можливість подальшої взаємодії із цифровим продуктом. Цей програмний сервіс може використовуватися в якості платформи для збору коштів задля різноманітних цілей через придбання невзаємозамінних токенів за фіксованою ціною у смарт-контракті, а сам токен-продукт в якості унікальної одиниці криптовалютного продукту.

На основі виконаної роботи можна зробити висновок, що запропонований децентралізований додаток реалізовує концепцію технології смарт-контрактів невзаємозамінних токенів блокчейну Ethereum, має актуальні функціональні та алгоритмічно-програмні рішення, а його основна ідея характеризується широкими функціональними можливостями, та при виконанні відповідних дослідницько-конструкторських робіт може бути адаптована для різноманітного практичного застосування.

Список літератури

1. Wang Q. Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges / Qin Wang, Rujia Li, Qi Wang, Shiping Chen // CSIRO Data61. 2021. P. 1–8. DOI: /10.48550/arXiv.2105.07447 (accessed: 22 September 2022).

2. Wu K. Empirical Study of Blockchain-based Decentralized Applications / Kaidong Wu // Key Lab of High-Confidence Software Technology, MoE. 2019. P. 3–4. DOI: /10.48550/arXiv.1902.04969 (accessed: 22 September 2022).

3. Andreas M. Antonopoulos Mastering Ethereum: Building Smart Contracts and DApps/ Andreas M. Antonopoulos, Dr. Gavin Wood. 2018. Pp. 127–128, 268–269. URL: https://dl.ebooksworld.ir/motoman/Mastering_Ethereum_Andreas.M.Antonopoulos.www.EBooksWorld.ir.pdf (accessed: 22 September 2022). (accessed: 22 September 2022).

4. Mota M. Ethereum Development With Go / Miguel Mota // 2021. P. 4. URL: https://goethereumbook.org/ethereum-development-with-go.pdf (accessed: 22 September 2022).

5. Laurent A. Transaction fees optimization in the Ethereum blockchain / Arnaud Laurent, Luce Brotcorne, Bernard Fortz // Blockchain: Research and Applications. 2022. № 3. P. 2–3. DOI: /10.1016/j.bcra.2022.100074 (accessed: 22 September 2022).

6. Remix IDE Documentation. [Electronic resource]. – URL: https://remixide.readthedocs.io/en/latest/ (accessed: 22 September 2022).

7. Daniel E. IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks / Erik Daniel, Florian Tschorsch// IEEE Communications Surveys & Tutorials. 2022. № 24. P. 5. DOI: /10.21203/rs.3.rs-951089/v1 (accessed: 22 September 2022).

8. MetaMask. [Electronic resource]. – URL: https://en.wikipedia.org/wiki/MetaMask (accessed: 22 September 2022).

9. Infura API Documentation. [Electronic resource]. – URL: https://docs.infura.io/infura/networks/ethereum (accessed: 22 September 2022).

DECENTRALIZED SMART CONTRACT SOFTWARE SERVICE USING ETHEREUM BLOCKCHAIN NON-FUNGIBLE TOKENS

A. Ihnatovych, A. Yanchynskyi

Lviv Polytechnic National University,
Computer Engineering Department
E-mail's: anatolii.o.ihnatovych@lpnu.ua, andrii.yanchynskyi.mkisp.2021@lpnu.ua

The decentralized technologies of the Ethereum blockchain were analyzed, based on the principles of which a decentralized software service solution was proposed using a smart contract implemented in the specialized Solidity programming language. This contract acts as one of the elements of the server-client application and is the server part for processing methods of interaction with the Ethereum blockchain. The methods include a set of various functions, including for interacting with a collection of non-fungible tokens. The metadata that describes the digital product (non-fungible tokens) is directly linked to the blockchain and resides in a decentralized repository for its most reliable and long-lasting existence. The Ethereum blockchain, in addition to development tools, provides stable support for developers and users thanks to the popular decentralization paradigm, and the relevance and constant development of technologies lead to user interest in products created on their basis. Non-fungible tokens can be used as a cryptocurrency unit, and the decentralized application itself as a platform for collecting funds for a specific purpose.

The methods of creating decentralized software services using smart contracts of non-fungible tokens were studied. The selection of the main nodes of the decentralized application is substantiated. A detailed functional diagram of the operation of the basic smart contract methods is proposed together with a diagram that describes all the functional nodes of the decentralized application in general. The results of the interaction of the client part of the software service with the smart contract of the Ethereum blockchain are also demonstrated.

Keywords: blockchain, Ethereum, DApps, decentralized application, decentralized storage, IPFS, smart contract, non-fungible token, NFT, digital product, Solidity, cryptocurrency.