Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мультимедійних систем

Платформа для мінту/купівлі/продажу NFT (по типу OpenSea), реалізована на фреймворку Telegram Mini Apps

Текстова частина до курсової роботи
за спеціальністю “Комп’ютерні науки” 122

Керівник курсової роботи
Старший викладач
Гороховський К.С.
_____ (Підпис)
“___” _____ 2024 року
Виконав студент
КН-3 Тарасенко А. Г.
“___” _____ 2024 року

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем

ЗАТВЕРДЖУЮ
Зав.кафедри мультимедійних систем,
Жежерун Олександр Петрович
“___” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

Студента Тарасенко Артема Григорович факультету інформатики 3 курсу
ТЕМА: Платформа для мінту/купівлі/продажу NFT (по типу OpenSea), реалізована на фреймворку Telegram Mini Apps

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
Розділ 1. Основні поняття технології Blockchain
Розділ 2. Особливості мережі TON
Розділ 3. Фреймворк Telegram Mini Apps
Розділ 4. Розробка додатку
Висновки
Використані джерела

Дата видачі „___” _____ 2024 р.

Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання роботи:

№ з/п	Назва Етапу	Термін виконання	Примітка
1	Вибір теми	05.10.2023	
2	Затвердження теми з керівником	23.10.2023	
3	Дослідження технології Blockchain	15.11.2023	
4	Дослідження стандартів NFT в технології Blockchain	25.12.2023	
5	Дослідження особливостей мережі TON	20.01.2024	
6	Написання текстової частини (розділи 1-3)	15.02.2024	
7	Розробка додатку	14.03.2024	
8	Закінчення написання текстової частини (розділ 4 і висновки)	28.03.2024	

Студент Тарасенко А. Г.

Керівник Гороховський К.С. “_____”_____ 2024р.

Зміст

Анотація

Ціллю курсової роботи є розробка прикладного блокчейн застосунку, що працює на основі мережі TON і платформи Telegram Bot Apps. Логіка роботи застосунку полягає у наданні користувачу можливості легко створювати, купляти і продавати NFT за допомогою Telegram.

У роботі розглянуті основні поняття технології Blockchain, особливості блокчейну TON, можливості платформи Telegram Bot Apps та її інтеграції з блокчейном, основні поняття NFT, стандарти NFT на блокчейні TON.

Вступ

Станом на 2024 рік, кількість способів використання Blockchain-технологій для вирішення прикладних проблем постійно збільшується. Більшість інновацій на цьому ринку полягають в адаптації вже існуючих способів до децентралізації.

Централізовані бази даних вразливі за своїм підходом до збереження інформації, так як вони не гарантують достовірності джерел даних, від яких вони її отримали. Через це забезпечити фактичну цілісність окремо взятих баз даних є нетривіальною задачею, що потребує витрат ресурсів для кожного окремо взятого випадку.

Blockchain в свою чергу вирішує проблему гарантованості, що відкриває його для ринків, де вона є необхідною складовою. Одним з таких ринків є ринок цінних предметів. Для нього основною проблемою є підтвердження факту оригінальності предмету, що зводиться до нуля при використанні Blockchain як гаранту оригінальності. Таким чином виник ринок NFT, що і є в контексті даного ринку “цінним предметом”.

Ринок NFT розвивається на рівні з ринком криптовалют, що є дуже активним у 2024 році. Для спрощення торгівлі на ньому виникають NFT-біржі, або NFT-маркетплейси, що слугують гарантом між покупцем і продавцем, що угода здійсниться, а також надають широкі можливості для створення і просування своїх NFT.

На момент 2024 року, мережа TON спритно розвивається і має великий потенціал. Можливість утвердитись на ній якомога раніше може стати запорукою певних перспектив у рості відповідного проєкту. Таким чином і

створення NFT-маркетплейсу для неї має неабиякий сенс в контексті 2024 року.

Розділ 1. Основні поняття технології Blockchain

1.1 Розподілена база даних

Назва технології “Blockchain” походить від двох слів англійської мови: “block” і “chain”, що перекладаються як “блок” і “ланцюг” відповідно. Її можна адаптувати українською як “ланцюг блоків”. Вона доволі буквально передає основу всієї технології.

Блок - основна структурна одиниця ланцюгу. Сам по собі він являється структурою, що містить деяку кількість транзакцій, інформацію про попередній блок та додаткові метадані.

Щодо кількості транзакцій всередині одного блоку, ця кількість є динамічною та залежить від обмежень конкретної мережі. Більшість великих мереж використовують розбиття на блоки за часовою основою, тобто блоки створюються один раз в певний проміжок часу. До мереж, що використовують розділення блоків на основі часу, можна віднести, наприклад, мережу Bitcoin, що створює новий блок кожні 10 хвилин, або мережу TRON, що створює блоки один раз на 3 секунди.

Розподіленість мережі створюється шляхом існування нод. Нода - сервер, що зберігає повну або часткову копію мережі, синхронізується з іншими нодами, а також певним чином взаємодіє з мережею. Основними цілями існування нод є підтримка децентралізації, відправка та перевірка транзакцій. Таким чином, мережа співзв’язних нод утворюють децентралізоване середовище, де кожна нода вкладає зусилля у підтримку правильності мережі.

1.2 Гарантованість

Запорукою гарантії правильності транзакцій в Blockchain є криптографія. Різні мережі використовують різні підходи для забезпечення безпеки їх транзакцій. За безпекою транзакцій в цьому контексті можна вважати гарантованість того, що транзакція виконувана і базується на погодженому між усіма нодами стані мережі на момент її виконання.

Як вже зазначалось раніше, будь-який блок в мережі має посилання на попередній блок. Це посилання зберігається у вигляді відповідного хешу, що обраховується з даних минулого блоку шляхом виконання на ньому односторонньої хеш-функції. Таким чином, зміна будь-яких даних блоку, старшого за поточний, призведе до необхідності зміни всієї послідовності хешів наступних блоків.

Щоб запобігти попаданню в мережу блоків, що не є правильними, використовуються механізми консенсусу. Ціллю їх існування є узгодження попадання блоку в мережу між нодами. Наразі трьома найпопулярнішими механізмами консенсусу є:

1) “Proof of Work” - блок підтверджується за допомогою виконання складного алгоритму, що займає багато часу на обрахунок, але мало на перевірку правильності підрахунку. Для того, щоб отримати можливість додати у блокчейн неправильний блок, при цьому механізмі, злодіям необхідно мати 50% або більше від загальної кількості “майнерів” (саме так називаються сервери, що виконують складний алгоритм) всієї мережі.
2) “Proof of Stake” - блок підтверджується користувачами, що вклали певну кількість нативної валюти у вигляді застави за те,

що вони правильно підтверджуватимуть блоки. Таким чином, додавання неправильного блоку у блокчейн злодіями буде можливо тільки у випадку заволодіння ними більшості в сумі застав.

3) BFT (або “byzanite fault-tolerant”) - цитуючи наукову роботу Університету Торонто “Reaching Consensus in the Byzantine Empire: A Comprehensive Review of BFT Consensus Algorithms” [3], “алгоритми консенсусу BFT є в основі впровадження безпеки і гарантій життєздатності для розподілених систем, що оперують в середі існування довільних помилок”. Хоч і існує багато варіацій алгоритму BFT, децентралізовані його варіації базуються на припущенні що більшість нод знатимуть правдиву інформацію. Таким чином, щоб додати неправильну транзакцію у BFT блокчейн, зловмисникам необхідно було б оперувати більшістю нод.

Таким чином, гарантованість Blockchain-мереж забезпечується як на рівні додавання даних, так і на рівні збереження вже існуючих

1.3 Нативна валюта

Так як будь-яка транзакція має бути підтверджена з використанням певної кількості ресурсів, ні одна мережа не може існувати без основного способу розрахунку за комісію транзакцій. Таким способом для абсолютної більшості мереж є нативна валюта – тобто валюта, що є вбудованою в конструкцію мережі.

Нативні валюти зазвичай створюються у ході підтвердження блоків як винагорода за витрати (механізм“Proof of Work”) або блокування (механізм “Proof of Stake”) ресурсів.

1.4 Blockchain як платформа

1.4.1 Програми для Blockchain

Крім основного функціоналу - переказу нативної валюти, сучасні Blockchain-мережі не можуть існувати без функціоналу для розробки і виконання програм, написаних сторонніми розробниками. Такі програми дозволяють розробити додаткову логіку на основі вже існуючої логіки Blockchain, при цьому користуючись всіма перевагами мережі.

Різні мережі вводять різні назви для таких програм. Таким чином, мережа Solana використовує термін “Program”, тобто “Програма”, а Ethereum визначає програму, що виконується на Blockchain, як “Smart-Contract” (надалі “смарт-контракт”). Цей термін є найбільш використаним і серед інших мереж.

Хоч різниця між різними мережами і може утворювати доволі велику різницю в написанні смарт-контрактів, для більшості мереж логіка є такою:

1) Виконується код контракту з станом мережі на момент виконання
2) Якщо при виконанні код повертає помилку, транзакція автоматичним чином відхиляється і є не валідною
3) Якщо при виконанні код завершує своє виконання без помилок, дані про зміни в стан мережі, що були спричинені в ході виконання програми, відправляються в мережу для подальшого підтвердження нодами.

4) Ноди виконують цей контракт у себе з відповідними вхідними даними і перевіряють співпадіння вхідних і вихідних даних. У випадку, якщо вони співпадають, нода підтверджує факт правильності транзакції

Зміни стану мережі, що дозволені для смарт-контракту, є обмеженням, що вводиться різними Blockchain-мережами по-різному. В свою чергу, найбільш розповсюджена модель, що використовується як у Ethereum, так і у TON, дозволяє програмі мати власний стан на мережі, до якого можна отримати доступ, виключно виконавши код цієї ж програми. Цей підхід можна порівняти з об’єктно-орієнтованим програмуванням, де кожен клас має власний внутрішній стан, як і смарт-контракт на мережі.

1.4.2 Off-chain виконання програм

Раніше зазначений метод, коли після локального виконання коду контракту, результат відправляється для підтвердження нодами, має сенс тільки тоді, коли виконання контракту змінює стан мережі.

Для інших випадків використовується виконання коду “Off-chain”, що означає локальне виконання коду контракту, коли розробник може отримати і зрозуміти результат виконання, але він не відправляється в мережу для підтвердження

Цей метод використовується в двох випадках:

1) Коли необхідно “прочитати” стан мережі, тобто виконати чисту функцію з коду, що повертає дані, не змінюючи їх
2) Коли необхідно зробити симуляцію виконання контракту з деякими вхідними даними і зрозуміти результат виконання

1.5 Об’єкти економіки в Blockchain

1.5.1 Нативна валюта

Окрім основного застосування нативної валюти - розрахунок за комісію транзакції і отримання нагород за підтримку мережі, нативні валюти мають і економічне значення для мережі.

1.5.2 Взаємозамінні токени

Взаємозамінні токени (або “Fungible Tokens”) - це валюти, що побудовані на основі мережі, але не є її вбудованою частиною. Здебільшого, взаємозамінні токени імплементуються за допомогою смарт-контракту

Смарт-контракт взаємозамінного токену імплементує логіку переказу валюти і отримання балансу користувача. Також, серед популярних функціональностей смарт-контрактів взаємозамінних токенів є можливість надання доступу іншій адресі до певної кількості своїх коштів (“allowance”). Таким чином, можна дозволити смарт-контракту списання якоїсь кількості валюти з свого рахунку, не роблячи додатковий переказ на адресу самого смарт-контракту.

Так як всі токени підпорядковуються схожій логіці, різні мережі мають різні, хоч і схожі, домовленості з приводу форми таких контрактів. Зазвичай, загальноприйнятим є один стандарт для однієї мережі з деякими можливими його розширеннями. Прикладом такого стандарту є ERC-20 [1], стандарт взаємозамінного токену на мережі Ethereum. Цей стандарт визначає методи, що повинні бути визначені на контракті:

1) “name” - метод повинен повертати назву токену

2) “symbol” - метод повинен повертати коротку назву (символ) токену
3) “decimals” - метод повинен повертати кількість цифр після коми у чисел, що визначають кількість токену. Цей параметр є необхідним, так як більшість мереж не мають можливості обчислень над числами з плаваючою точкою, а натомість використовують числа з сталою точку
4) “totalSupply” - загальна кількість токенів в циркуляції
5) “balanceOf” - функція приймає адресу і повертає її баланс
6) “transfer” - функція приймає адресу користувача, що повинен отримати певну кількість (що передається як параметр) токенів.
7) “transferFrom” - функція приймає адресу користувача, що повинен отримати певну кількість (що передається як параметр) токенів, а також адресу користувача, чиї кошти ми хочемо використати для відправки (ця функція використовує функціонал “allowance” для відправки токену з іншого акаунту)
8) “approve” - функція, що приймає адресу користувача і кількість токенів, які можна дозволити для використання цьому користувачеві. Таким чином, можна надати доступ до певної частини балансу свого акаунту іншому акаунту
9) “allowance” - функція, що надає дані про баланс, що дозволений до використання одній адресі іншої адреси.

Хоч цей приклад і показує стандарт взаємозамінних токенів виключно для мережі Ethereum, за допомогою цього стандарту можна зрозуміти основну функціональність, що потребується від токену, і на інших мережах.

1.5.3 Невзаємозамінні токени (NFT)

Невзаємозамінні токени (NFT) є способом представлення унікальних об’єктів в мережі Blockchain. На протиставлення взаємозамінним токенам, будь-який NFT має тільки 1 копію і може бути у власності виключно одного акаунту в момент часу.

Основними способами застосування NFT є:

1) Можливість акаунтом володіння і передачі NFT
2) Можливість віддати контроль власним NFT іншому акаунту

Щоб детальніше зрозуміти NFT, варто звернути увагу на стандарт невзаємозамінних токенів для Ethereum, ERC-721 [2] (стандарт визначає контракт, що є набором NFT, хоча і не забороняє існування тільки одного NFT в наборі):

1) “balanceOf” - функція, що приймає адресу акаунту і повертає кількість NFT з набору в його власності
2) “ownerOf” - функція, що приймає ідентифікатор NFT в наборі і адресу користувача і повертає логічне значення, що підтверджує або заперечує власність акаунту над певним NFT
3) “transferFrom” - функція, що дозволяє зробити переказ NFT з акаунту іншого користувача (якщо він надав доступ до цього за допомогою “approve”)
4) “safeTransferFrom” - має логіку ідентичну до “transferFrom”, але додатково перевіряє можливість акаунтом отримувати NFT
5) “approve” - дозволяє надати доступ іншому акаунту до управління своїм NFT

6) “setApprovalForAll” - дозволяє надати доступ іншому акаунту до управління всіма своїми NFT в наборі
7) “getApproved” - дозволяє отримати адресу, що має управління над NFT
8) “isApprovedForAll” - дозволяє перевірити, чи має акаунт повний доступ до NFT іншого в межах набору

Схожі стандарти використовуються і для інших мереж, хоч і мають певні різниці в назвах або параметрах функцій

Розділ 2 Особливості мережі TON

2.1 Actor Model

Першочерговою особливістю мережі TON є використання “Моделі Виконавця” (“Actor Model”). Це є імплементацією поведінкового патерну “Виконавець” (“Actor”).

В основі патерну “Виконавець” лежить сутність, що має можливість приймати і обробляти повідомлення. Логіка обробки повідомлень схована за принципом чорної коробки, де чорною коробкою виступає виконавець. Варто зазначити, що патерн не обмежує надсилання повідомлень іншим виконавцям під час обробки повідомлення, але зобов’язує середу виконання до створення функціональності комунікації між ними.

TON імплементує патерн за допомогою смарт-контрактів, тобто будь-яка адреса мережі представляє собою деякий смарт-контракт або пусту адресу, де смарт контракти можуть комунікувати одне з одним або розміщувати нові смарт контракти. Персональні рахунки в цьому випадку також є смарт-контрактами. Виходячи з цього, транзакція в TON визначається як порядок дій:

1) Смарт контракт отримує повідомлення
2) Код контракту виконується в віртуальній машині TON
3) Контракт модифікує власні дані
4) Контракт може надіслати вихідні повідомлення
5) Контракт переходить в режим очікування наступного повідомлення

Важливо зазначити, що при цьому один смарт контракт може обробляти одночасно тільки одне повідомлення

2.2 Організація транзакцій

Базовою складовою організації транзакцій в TON є “Ланцюжок Рахунку” (“AccountChain”). Він представляє з себе певну послідовність транзакцій для одного рахунку (адреси), де кожна транзакція посилається на тільки одну попередню. Через необхідність синхронізації транзакцій між нодами, ланцюжок рахунку представлений не у вигляді однозв’язного списку транзакцій, а у вигляді однозв’язного списку блоків, кожен з яких представляє певну кількість зв’язаних між собою транзакцій. На рівні структур даних, ланцюжок рахунку можна порівняти з однозв’язним списком однозв’язних списків.

Набір ланцюжків рахунків утворює “Ланцюжок Уламків” (“ShardChain”). Головною особливістю ланцюжку уламків є те, що вони можуть бути розділені і об’єднані. Це дозволяє мережі балансувати навантаження між нодами

2.3 Збереження даних

Всі дані в мережі TON зберігаються в “клітинках” (“cell”). Клітинка є структурою даних, що містить в собі:

1) Не більше 1023 біт даних
2) Посилання на не більше ніж 4 інші клітинки

При цьому, для клітинок застосовується правило, що клітинки повинні утворювати ациклічний граф (іншими словами, циркулярні залежності є забороненими).

Використання клітинок дозволяє мережі більш ефективно розміщувати дані і використовувати дедублікацію для оптимізації розміру мережі

2.4 Розробка смарт-контрактів

Будь-який смарт-контракт в TON виконується в TVM (віртуальна машина TON). Вона працює за принципом стеку, що дозволяє їй бути ефективною і простою для реалізації.

Екосистема TON пропонує декілька мов програмування для написання смарт-контрактів:

1) FunC - мова програмування, що є основною в екосистемі. Вона використовується для написання більшості смарт-контрактів, в тому числі і системних
2) Tact - мова, розроблена спільнотою TON. Вона дозволяє приблизити досвід розробки на мережі до розробки контрактів на мові Solidity. Є більш високорівневою альтернативу FunC
3) Fift - низькорівнева мова програмування, що працює на принципу стеку, як і TVM. Це дозволяє використовувати мову для написання більш ефективних смарт-контрактів, а також більш низькорівневої взаємодії з віртуальною машиною для вирішення деяких технічних викликів

Окрім мов програмування, TON також надає середовище для розробників під назвою “Blueprint”. Воно включає в себе утилітарні застосунки для тестування, побудови і розгортання смарт контрактів як у тестову, так і у звичайну мережу.

Розділ 3 Фреймворк Telegram Mini Apps

3.1 Telegram Mini Apps як спосіб розміщення веб-додатків

Telegram Mini Apps - одна з функціональностей ботів в месенджері Telegram, що дозволяє використовувати веб-додатки в поєднанні з Telegram. Реалізується вона за допомогою браузера, що відкриває певну веб-сторінку зсередини додатку. Фактором, що відрізняє фреймворк Telegram Mini Apps від звичайного вбудованого браузеру є API, які він дозволяє використовувати зсередини веб-додатку.

API включає, але не обмежується таким списком функціональностей:

1) Отримання інформації про користувача
2) Отримання інформації про тему месенджеру
3) Відправка повідомлень в чат користувача з ботом
4) Взаємодія з онлайн-оплатою через Telegram
5) Відображення спливаючих вікон для нотифікації користувача або підтвердження певних дій
6) Використання вбудованого в Telegram читача QR кодів
7) Взаємодія з гаманцями TON

Це дозволяє розробникам розробляти додатки, щільно інтегровані з екосистемою Telegram, не обмежуючись тим, що дозволяє звичайний API взаємодії з телеграм-ботами, а також розробляти інтерфейси, що краще підходять для виконання певної задачі користувачем.

3.2 Telegram Mini Apps і TON

Однією з ключових функціональностей Telegram Mini Apps є API взаємодії з мережею TON. Ключовою складовою, що робить це можливим, є “TON Connect” - набір інструментів з відкритим вихідним кодом, що є універсальним стандартом авторизації для TON екосистеми. Він використовує особистий протокол для передачі даних між додатком та гаманцем користувача, таким чином дозволяючи комфортну взаємодію з смарт-контрактами мережі через користувацький інтерфейс

Розглядаючи Telegram Mini Apps, розробник може з легкістю зареєструвати користувача за допомогою вбудованого в Telegram гаманця, а також використовувати Wallet Pay для сплати за допомогою TON. Також підключення гаманця дозволяє швидку взаємодію задля підписання транзакцій і спрощує користувацьку взаємодію

Розділ 4 Розробка додатку

4.1 Визначення вимог

Функціональні вимоги:

1. Користувач повинен мати можливість авторизуватись за допомогою криптовалютного гаманця
2. Користувач повинен мати можливість подивитись список своїх NFT
3. Користувач повинен мати можливість подивитись список активних продажів NFT
4. Користувач повинен мати можливість розмістити свій NFT на продаж за певну ціну, що визначається в TON
5. Користувач повинен мати можливість подивитись список своїх активних продажів NFT
6. Користувач повинен мати можливість відмінити розміщення NFT на продаж
7. Користувач повинен мати можливість придбати NFT за допомогою одного з активних продажів

Нефункціональні вимоги:

1. Мережа TON повинна виступати основною базою даних, єдиним “source of truth”
2. Функціональність роботи з NFT на Blockchain повинна бути реалізована за допомогою смарт-контрактів
3. Додаток повинен мати приємний користувацький інтерфейс

4. Інтерфейс додатку повинен підлаштовуватись під користувацьку тему месенджеру Telegram
5. Додаток повинен працювати стабільно
6. Додаток не повинен зберігати дані для прямого доступу до гаманця користувача (приватний ключ, мнемоніка і т.д.)

4.2 Архітектура

Враховуючи вимоги до застосунку, очевидною є необхідність розробки веб-застосунку, що інтегрувався б в Telegram за допомогою фреймворку Telegram Mini Apps. Однак, при цьому постає питання надання користувачу кнопки для входу в додаток. Її повинен надіслати бот, відповідно одним з необхідних елементів архітектури є сервер, що обробляв би взаємодію користувачів з ботом і відправляв їм кнопку для відкриття додатку.

Враховуючи, що основним джерелом правдивої інформації за вимогами є мережа TON, в додатку використовуватиметься пряма взаємодія з нею у місцях, де необхідна повна точність зображеної інформації. При цьому обмеженням, яке важливо вбачати, є те, що додаток не може запитувати у мережі напряму інформацію щодо розміщених контрактів. Так як за філософією TON кожна можлива сутність повинна визначатись окремим контрактом, в даному випадку проблема буде полягати в тому, щоб знайти всі контракти продажів маркетплейсу, а також NFT, чиїм власником є акаунт.

Проблема з NFT вирішується за допомогою індексатора - додатку, що збирає інформацію транзакцій, що відбуваються в мережі і розміщує їх в базу даних, з якої їх можна запитувати вже напряму. Більшість індексаторів, що вже існують на мережі TON, підтримують пошук NFT за власником. Хоча

вибір індексаторів і невеликий, для правильного вибору необхідно сформулювати критерій оцінювання. Враховуючи, специфіку вимог додатка в даному випадку його можна сформулювати як комбінацію двох факторів:

1. Актуальність даних
2. Відсутність істотних обмежень щодо запиту даних
3. Доступність даних у Testnet
4. Безкоштовність

Таким чином, порівнюючи найпопулярніші індексатори мережі TON:

1. Anton - має непогану актуальність даних для Mainnet, є безкоштовним, але при прямому порівнянні з іншими індексаторами показує набагато гірші результати актуальності порівняно з іншими в Testnet, що ускладнює процес розробки додатку
2. Індексатор від TonApi - має гарну актуальність як для Mainnet, так і для Testnet, але не є безкоштовним і має обмеження щодо запиту відносно купленого пакету
3. Індексатор від TonCenter - має середню актуальність даних, яка розповсюджується на обидві мережі. Має обмеження в 1 запит в секунду без API ключа, але отримання API ключа є безкоштовним, тож кількість запитів можна вважати умовно необмеженою

Таким чином, для розробки проєкту був обраний індексатор від TonCenter, так як він є оптимальним відносно заданих критеріїв.

Однак, індексатор є вирішенням тільки проблеми з NFT. Вирішення проблеми з отриманням списку продажів вирішується за допомогою написання власного сервера, що слідкуватиме за мережею і зберігатиме нові

продажі у своїй базі даних. Такий підхід хоч і потребуватиме дещо особливого підходу до вирішення питання взаємодії смарт-контрактів, але є оптимальним за співвідношенням між часовою складністю реалізації і кінцевою ефективністю. Особливість підходу до роботи з контрактами полягатиме в необхідності існування окремого контракту, задачею якого стало б “розміщувати” нові продажі, а тобто створювати нові смарт контракти продажів. Таким чином, щоб дізнатися про створення нових продажів на мережі, необхідно просто перевірити, чи були у цього контракту нові транзакції і відфільтрувати їх по коду операції. Щодо оновлення статусу продажів, це можливо зробити на основі даних про створені продажі, запитуючи їх статус за допомогою get-методу.

У підсумку, архітектура додатку складатиметься з таких складових:

1. Веб-застосунок, що відображатиметься в Telegram за допомогою Telegram Mini Apps
2. Сервер, що відстежуватиме нові події, пов’язані з ботом в Telegram і відправлятиме новим користувачам кнопку для входу в додаток
3. Веб-сервер, що відстежуватиме події, що відбуваються з контрактами продажів
4. Індексатор TonCenter
5. Мережа TON

4.3 Вибір технологій

4.3.1 Веб-застосунок

Так як Telegram Mini Apps дозволяє нам використовувати ті ж API, що доступні і в браузері (бо реалізується за допомогою браузеру), вибір технології для цієї частини застосунку спрощується до вибору фреймворку для веб-застосунку.

Так як додаток не має необхідності індексуватись в пошукових системах, а також використовує не тільки власну базу даних для отримання даних, більш оптимальним у порівнянні з SSR (Server-Side-Rendering) тут є підхід CSR (Client-Side-Rendering). Серед CSR фреймворків, найпопулярнішими є:

1. React
2. Angular
3. Vue.js

Порівнюючи їх, для вирішення даної проблеми на думку автора найбільше підходить Vue.js. Angular є занадто комплексним і ускладненим рішенням, яке буде тільки уповільнювати розвиток проєкту. React, хоч і є популярним при розробці у сфері Blockchain, але є доволі низькорівневим і додає складності через його модель рендерингу.

Також важливою частиною веб-застосунку є можливість підключення гаманця. Ця функціональність досягається за допомогою протоколу TON Connect і TON Connect SDK.

4.3.2 Веб-сервер відстеження подій контрактів продажів

Так як необхідність цього сервісу здебільшого полягає у кешуванні і локальному індексуванні даних з блокчейну, основною вимогами перед ним є легкість, різноманітність можливостей розгортання та можливість виконання відкладених задач. Легкість можна гарантувати, використовуючи примітивні Javascript-фреймворки (Typescript-фреймворки), наприклад Express, Fastify, H3 або Nitro. В той самий час різноманітність можливостей розгортання є більшою проблемою. В сучасному світі, з постійним розширенням ринку хостингових технологій, з’являється все більше пропозиції щодо способів розгортання додатків. Одним з свіжих, станом на 2024 рік, є концепція “Edge”, коли сервер запускається не як постійно працюючий процес в конкретному датацентрі, а за необхідності розгортається в локації, що є найближчою до кінцевого користувача. Використання технології надає перевагу не тільки в швидкості роботи додатку, а й в вартості, так як сервер запускає код програми тільки в проміжок часу, коли він потрібен для відповіді на запит. З вже перелічених фреймворків, тільки 2 мають можливість розгортання на “Edge” так само як і на звичайному сервері: H3 і Nitro (використовує H3 як основу). Серед них для розробки даної частини проєкту був обраний Nitro, так як він має всі переваги H3, але крім цього реалізує додаткову функціональність для спрощення процесу розробки

Необхідною складовою даного веб-серверу є взаємодія з базою даних. Враховуючи особливість платформи “Edge”, важливо будувати взаємодію з базою даних таким чином, щоб якомога менше залежати від середовища виконання. Цього можна досягти, використовуючи ORM, що є відокремленою від конкретної бази даних. В контексті цього проєкту була використана ORM

Prisma. Ця система взаємодії з базою даних дозволяє підключатись як до класичних баз даних (PostgreSQL, MySql, і т.д.), так і до “Edge” баз даних, таких як, наприклад, Nitro (розподілений PostgreSQL), PlanetScale (розподілений MySQL), Cloudflare D1 (розподілений SQLite). При цьому можна використовувати ідентичну схему бази даних, що дозволяє легко змінити середовище виконання у разі знаходження варіанту, що виконуватиме задані задачі ефективніше.

4.3.3 Сервер для роботи з ботом Telegram

Беручи до увагу те, що відправка заявок на отримання посилання користувачем відбувається тільки 1 раз за період його користування додатком, тобто навантаження на сервер майже не здійснюється, оптимальним варіантом в контексті роботи автор вважає використання JavaScript (TypeScript) для вирішення задачі.

Для мови JavaScript (TypeScript) існує декілька найпопулярніших бібліотек для роботи з Telegram:

1. node-telegram-bot-api - найстаріша бібліотека, має погану підтримку TypeScript, погано підходить для виконання на “Edge”
2. telegraf - більш модерна технологія, яка має значно покращену підтримку TypeScript, але погано підходить для виконання на “Edge”
3. grammy - офіційна рекомендація від Telegram для TypeScript. Має ідеальну підтримку TypeScript і побудована, щоб працювати на інших платформах, враховуючи різні середи виконання “Edge”

4.3.4 Смарт-контракти

Серед можливих мов написання смарт-контрактів для TON, для розробки даного проєкту автором була вибрана мова FunC. Вона є більш високорівневою ніж Fift, що дозволяє розробці розвиватись набагато більш стрімко, але в свою чергу має більшу документованість, ніж мова Tact, що є розробленою спільнотою високорівневою мовою програмування.

4.3 Виконання розробки

4.3.1 Смарт-контракти

Враховуючи, що всі частини додатку, так чи інакше, базуються навколо взаємодії з мережею TON, основна частина логіки маркетплейсу розміщуватиметься саме в смарт-контрактах.

4.3.1.1 NFT та колекції NFT на TON

Для початку, необхідно розглянути, як NFT зберігаються в мережі TON. Стандартом інтерфейсу є TEP-0062 [4], що визначає вимоги для контрактів предмету NFT і колекції NFT.

Колекція NFT є набором однотипних (зазвичай, з однаковим початковим кодом) предметів NFT. Колекція NFT визначається інтерфейсом nft_collection. Кожен NFT в колекції повинен мати унікальний ідентифікатор, що зазвичай позначається словом index. Це дозволяє однотипним контрактам мати різну початкову адресу. За стандартом, колекція не має визначати ніяких

кодів операцій, тобто логіка розгортання нових NFT повністю залежить від конкретної реалізації, але повинна визначати декілька get-методів:

1. get_collection_data - метод повинен повертати дані про індекс наступного елементу колекції, адресу власника колекції, а також “контент” (що зазвичай містить метадані) колеції**.**
2. get_nft_address_by_index(int index) - метод повинен повертати адресу NFT за індексом
3. get_nft_content(int index, cell individual_content) - метод повинен повертати “контент” предмету NFT на основі індексу NFT і “контенту”, отриманого з самого NFT. Цей метод використовується для того, щоб дати колекції можливість розширювати контент предмету, наприклад, визначаючи спільні метадані, або генеруючи їх на основі індексу. Популярною імплементацією цього метода є пряме повернення individual_content

Також стандарт TEP-0062 [4] визначає інтерфейс і для NFT, що має назву nft_item і, на відміну від колекції, містить як операції для роботи з NFT, так і get-методи.

Операції інтерфейсу nft_item:

1. transfer - операція, що дозволяє передати володіння NFT іншій адресі. Окрім функціональності зміни власника, містить додаткові опції. Першою є можливість після зміни власника відправити певну кількість TON на гаманець нового власника, сповістивши його таким чином про переказ NFT. Це є доречним для використання тоді, коли є необхідність робити певну дію після переказу NFT. Також до TON можна додати наповнення, що

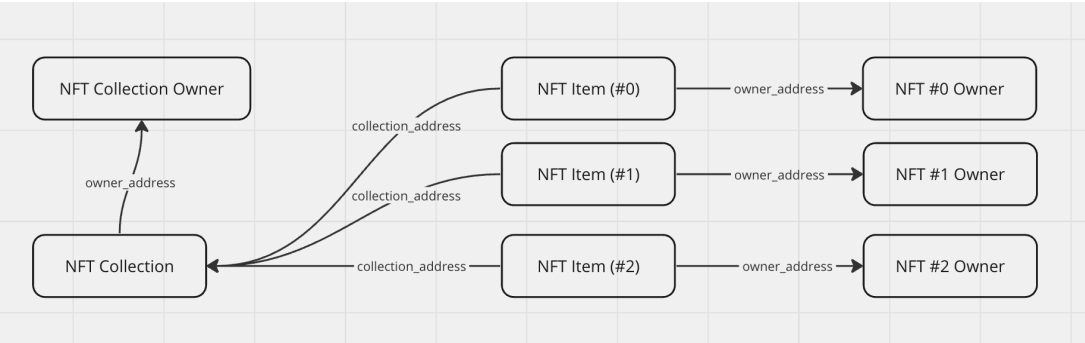
передається за допомогою параметру forward_payload. Другою додатковою опцією є можливість встановлення рахунку, що отримає кошти, що залишились після виконання транзакції. Таким чином, при більш комплексних обрахунках, можна відправляти залишок TON, що не був використаний на виконання транзакцій, наприклад, відправнику транзакції

2. get_static_data - метод, що дозволяє асинхронно отримати дані, про індекс та колецію, якій належить предмет NFT. Необхідність існування цієї операції полягає у неможливості повернення даних після виконання контракту в TON, таким чином єдиним можливим рішенням є передання статичних даних за допомогою додаткового виклику того ж контракту, що ініціював виконання операції

Get-методи інтерфейсу nft_item:

1. get_nft_data - повертає інформацію про індекс, адресу колекції, адресу власника, а також контент NFT.

Таким чином, зображаючи графічно, зв’язок між акаунтами NFT, колекції NFT і їх власниками виглядає як:



Варто зазначити, що власники NFT і колекції можуть бути одним акаунтом.

Для тестування і виконання роботи були використані референсні контракти NFT та колекції NFT, так як вони є лише допоміжними елементами для реалізації системи.

4.3.1.2 Контракт продажу NFT

Для реалізації продажів була вибрана модель, схожа на модель збереження NFT: існує основний контракт - маркетплейс, що управляє продажами, кожен з якого має однаковий код і певний індекс. Це дозволяє нам отримати історію створення продажів, переглянувши історію транзакцій акаунту маркетплейсу і відфільтрувавши їх за кодом операції.

В той же час, на відміну від NFT, контракт маркетплейсу хоч і має власника, але не обмежує використання операції створення контрактів продажів, тобто ця операція може бути виконана будь-яким іншим контрактом. Таким чином, маркетплейс повинен мати тільки 1 можливу операцію і додаткові get-методи для отримання адреси продажу за індексом і отримання даних про маркетплейс.

Щодо контракту продажу, get-метод у ньому є тільки один, для отримання даних про продаж. В той самий час він має декілька можливих операцій:

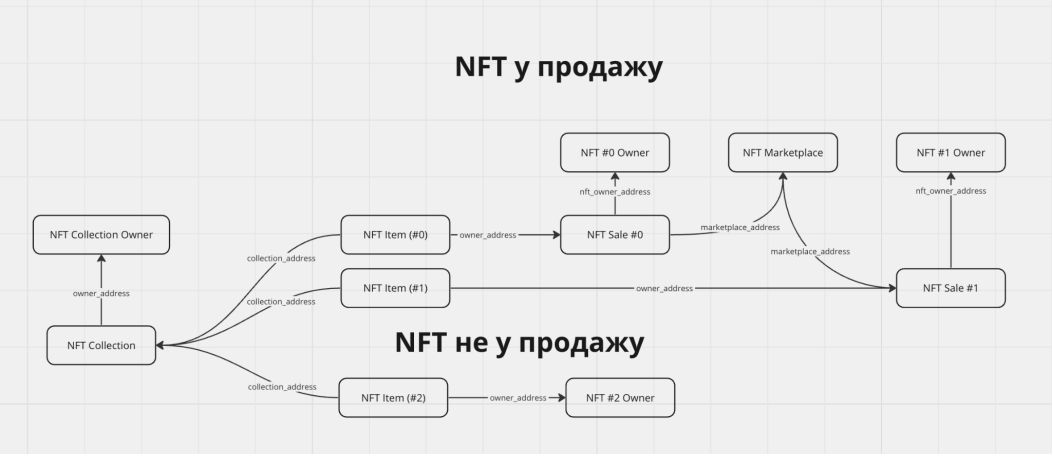
1. op::ownership_assigned - операція, що викликається NFT після його переведення на адресу контракту
2. Отримання TON - номер транзакції, який не робить ніяких дій з вхідними даними і потрібен тільки для можливого продовження оренди адреси контракту

3. Купівля NFT - операція перевіряє кількість надісланих у контракт TON і у випадку коли кількість є достатньою для покриття комісії переказів і купівлі NFT, відбувається купівля
4. Відміна продажу NFT - операція, яку може виконати тільки власник NFT. Вона повертає NFT власнику

Життєвий цикл контракту включає такі етапи:

1. Контракт є створеним але не є активованим - статус, якого набуває контракт одразу після створення за допомогою контракту маркетплейсу
2. Контракт є активованим - статус, якого набуває контракт після переказу на нього NFT і виконання операції op::ownership_assigned, що викликається з NFT на адресу отримувача у випадку, коли forward_amount є більшим за 0
3. Контракт знищено - цього статусу контракт може набути в одному з двох випадків: продаж було відмінено власником NFT або NFT було куплено. В обох випадках контракт закінчує своє з повним переведенням залишкового балансу отримувачу NFT і самознищенням.

Таким чином, зображуючи графічно зв’язки між акаунтами:



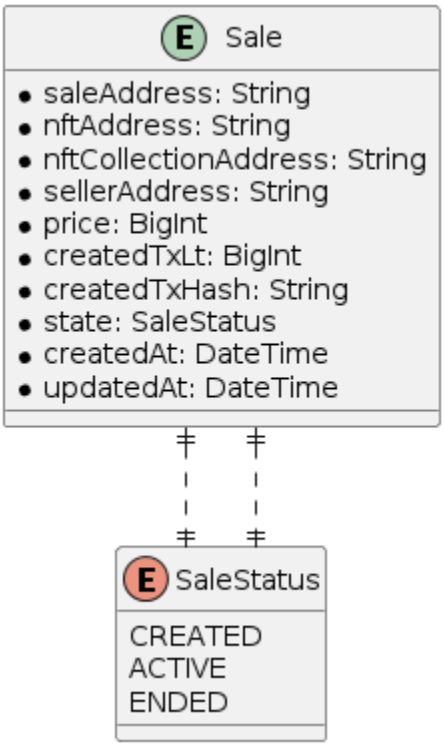
4.3.2 Веб-сервер відстеження подій контрактів продажів

Реалізацію цієї частини застосунку можна розбити на дві основні частини:

1. Задачі синхронізації інформації з мережею TON
2. API для веб-додатку

Обидві частини у такому випадку взаємодіють з базою даних, де 1 - записує дані, 2 - запитує.

Схема бази даних для цієї частини роботи:



Для можливості виконання синхронізації в автоматичному режимі, наприклад за допомогою cron-задач, використано Nitro Tasks. Ця можливість фреймворку Nitro дозволяє виконувати і планувати для автоматичного виконання задачі. Визначним плюсом при цьому є незалежність від середи виконання і можливість запуску задач як за допомогою планування cron-задач

на стороні серверу, так і виконавши HTTP-запит для виконання задачі, що дозволяє виокремити логіку планування синхронізацій у зовнішній сервіс за необхідності. Такий сервіс може, наприклад, викликати синхронізацію, отримавши інформацію про новий блок в мережі. В рамках виконання роботи використовується тільки інтервальний підхід до синхронізації, де задача відбувається 1 раз на хвилину.

Сам процес синхронізації складається з двох етапів, що відбуваються по порядку:

1. Синхронізація створення нових контрактів - перевіряється інформація про нові транзакції акаунту контракту маркетплейсу і за кодами операцій визначаються адреси створення нових контрактів продажів (або відсутність нових створених контрактів)
2. Синхронізація існуючих контрактів продажів - оновлення стану кожного окремого контракту продажу. Для цього використовується вбудований в контракт get-метод get_sale_data, що повертає його статус, або, у випадку знищення контракту, помилку, обробляючи яку ми можемо дізнатись про те, що продаж більше не є активним

Таким чином, за допомогою відкладених задач додаток автоматично синхронізує стан бази даних з станом на мережі TON і дає можливість веб-додатку використовувати ці дані для отримання актуального списку продажів

4.3.3 Сервер взаємодії з ботом Telegram

Враховуючи вже зазначений вище вибір технологій, для виконання обробки користувацького повідомлення про запуск роботи боту може бути використаний один з двох методів:

1. Long Polling - метод, що отримує оновлення, що надходять в бот, надсилаючи запит на сервер Telegram і отримуючи у відповідь оновлення, які отримав бот за останній час
2. WebHook - метод, що полягає у налаштуванні адреси, на яку Telegram відправлятиме HTTP запит при отриманні ботом нового повідомлення.

Основним плюсом підходу “Long Polling” є простота розробки і відсутність необхідності налаштування публічної адреси і SSL-сертифікату для URL отримання оновлення, що є необхідним для роботи методу “WebHook”. В той самий час, мінусом у порівнянні з іншим методом є складність у горизонтальному масштабуванні у разі зміни навантаження і відсутність можливості роботи з цим методом за допомогою технології “Edge”.

Для виконання даної роботи був використаний метод “Long Polling” через простоту реалізації та використання в середі розробки, а також через низьке навантаження, що створюватиметься на бот, так як у стандартній ситуації (відсутність DDoS атаки) кількість оновлень не буде значно перевищувати приріст нових користувачів.

4.3.4 Веб-застосунок

В основі веб-застосунку, як вже зазначалось раніше, лежить фреймворк Vue.js. Також при розробці використовувався сучасний бандлер Vite, що дозволяє зібрати код в декілька .js файлів, які виконуватимуться в браузері (в даному випадку вбудованому в телеграм).

4.3.4.1 Компонентний підхід

Для підтримання чистоти кодової бази, а також кращої організації функціоналу у додатку використовується компонентний підхід. Компонент у контексті Vue.js - сутність, що інкапсулює певну логіку, в тому числі і рендерингу, для перевикористання або для кращої організації коду.

Прикладом використання компонентного підходу є компонент “NftCard”, що використовується на багатьох сторінках додатку і у різних контекстах, але виконує однакову функцію. Більш цікавим компонентом є “IndexerNftCard”, що реалізує логіку отримання повних метаданих NFT на основі об’єкту, отриманого з індексеру.

4.3.4.2 Візуальне зображення NFT

Однією з нетривіальних задач, з якими доводиться стикатись при роботі з NFT - це отримання візуальної інформації. Назва, опис, зображення - не є частиною стандарту NFT TEP-0062 [4], в свою чергу стандарт залишає можливість використання будь-яких даних в NFT. Це дає можливість, в свою

чергу, розміщувати метадані з назвою, описом, зображенням, або інший формат даних, який визначається в залежності від задачі.

Для більшості стандартних NFT контент визначається як метадані, стандартом для збереження яких в TON є TEP-0064 [5]. Він визначає формат метаданих, а також способи їх розміщення.

Способами розміщення метаданих токену в TEP-0064 [5] є

1. On-chain - метадані зберігаються прямо на блокчейні. Таким чином, їх отримання не потребує додаткових запитів до інших ресурсів, а також, в залежності від конкретного контракту, на зміну метаданих можуть накладатись певні обмеження або логіка зміни метаданих може бути повністю відсутня, таким чином роблячи токен іммутабельним.
2. Semi-chain - метадані зберігаються на блокчейні, але серед метаданих міститься поле “uri”, яке визначає URI для запиту додаткових метаданих в форматі JSON. У випадку, якщо певні дані (наприклад, назва) існують і в метаданих на блокчейні, і в JSON-файлі отриманому після запиту, пріоритетність мають дані, визначені на блокчейні. Це дозволяє робити часткову динамічну зміну метаданих, не створюючи транзакцій у блокчейн.
3. Off-chain - метадані зберігаються в мережі інтернет. У цьому випадку URI - це єдине, що міститься в контенті токену. Популярним є використання цього способу розміщення разом з IPFS, що гарантуватиме сталість файлу за визначеним URI.

Також важливо зазначити, що отримання метаданих NFT не зводиться тільки до запиту даних у контракту токену. Через особливість стандарту TEP-0062 [4], важливим є використовувати get-метод колекції NFT

get_nft_content, що може додатково модифікувати метадані, отримані від контракту токену.

Для спрощення взаємодії з метаданими NFT, при виконанні роботи було використано індексатор для отримання повної інформації про NFT. Цей спосіб, хоч і потребує додаткових запитів до індексатора, дає можливість значно пришвидшити процес зображення NFT користувачу, так як скорочує час на виконання операцій отримання даних.

4.3.4.3 Взаємодія з мережею TON

Як вже зазначалось раніше, для підключення гаманця користувача до додатку використовується TON Connect, що надає можливість взаємодіяти з мережею за допомогою гаманця користувача, не маючи доступу до приватного ключа/мнемоніки.

У роботи використовувалось два способи взаємодії з мережею, в залежності від необхідних дій: off-chain і on-chain.

Для роботи з off-chain взаємодією використовується TON SDK, що використовує JSON RPC API для взаємодії з нодами. Це дозволяє отримувати дані з Blockchain, не виконуючи підписання транзакцій.

При роботі з on-chain транзакціями, важливим етапом є їх підписання гаманцем користувача. До дій, що відносяться до цієї категорії, можна віднести: створення продажу, активація продажу, відміна продажу, купівля NFT. Процес відправки і підписання для такої взаємодії відбувається за допомогою TON Connect SDK, що надає можливість використовувати стандарт TON Connect з веб-сторінки, запитуючи у користувача дозвіл на проведення і підписання транзакцій.

4.3.4.4 Робота з даними отриманими від додатку Telegram

Для виконання вимоги щодо інтеграції додатку в користувацький інтерфейс Telegram використовується Telegram Web Apps SDK. Це шар взаємодії, що надає доступ до даних, які месенджер передає у додаток, а також доступ до функцій, що можуть використовуватись для взаємодії з Telegram.

В цій роботі інструмент використаний для того, щоб підлаштувати кольори додатку до теми, що використовується її користувачем. Також в додатку використовуються API для підтвердження і нотифікації користувача за допомогою вбудованих в Telegram модальних вікон

Висновки

В роботі досліджено способи, підходи та технології для реалізації додатку децентралізованого NFT-маркетплейсу на основі мережі TON і фреймворку Telegram Mini Apps. Розглянуто способи реалізації архітектури додатку, а також можливі варіації у підходах до реалізації окремих компонентів системи

Актуальність цієї роботи полягає у стрімкому рості мережі TON станом на 2023-2024 роки і відповідному зростанні активності, як серед авторів цифрового контенту, так і серед людей, зацікавлених в їх придбанні.

В роботі було розглянуто стандарти, що використовує мережа TON для роботи з NFT, способи і шляхи їх використання, деталі, що можуть стати проблемою в процесі роботи з ними.

Система, побудована в ході виконання курсової, може послугувати як самостійним продуктом, так і стартом для розвитку продукту у більший по кількості функціональностей маркетплейс.

Використані джерела

1. ERC-20. EIP-20: Token Standard. 2019. URL: https://eips.ethereum.org/EIPS/eip-20.
2. ERC-721. EIP-721: Non-Fungible Token Standard. 2018. URL: https://eips.ethereum.org/EIPS/eip-721.
3. Reaching Consensus in the Byzantine Empire: A Comprehensive Review of BFT Consensus Algorithms / G. Zhang et al. *ACM Computing Surveys*. 2023. URL: https://doi.org/10.1145/3636553 (date of access: 14.05.2024).
4. TEP-0062 NFT Standard. Official edition. 2022. URL: https://github.com/ton-blockchain/TEPs/blob/master/text/0062-nft-standard.md.
5. TEP-0063 Token Data Standard. 2022. URL: https://github.com/ton-blockchain/TEPs/blob/master/text/0064-token-data-standard.md.