Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Кваліфікаційна робота

освітній ступінь - бакалавр

на тему: «**РЕАЛІЗАЦІЯ NFT ПРОЕКТУ HTTPS://UACATSDIVISION.COM/ НА БЛОКЧЕЙН ПЛАТФОРМІ TON**»

Виконав: студент 4-го року навчання,

Освітньої програми «Прикладна математика», 113

Закутін Тимофій Павлович

Керівник: ас. Гороховський К.С. ,

_____ (підпис)

Рецензент: канд. фіз.-мат. наук

Жежерун О.П.

_____ (підпис)

Київ 2024

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

ЗАТВЕРДЖУЮ
Зав.кафедри математики, проф., д.ф.-м.н.
Р. К. Чорней
(підпис)
„____”_____2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу
студенту Закутіну Тимофію Павловичу факультету інформатики 4-го курсу
ТЕМА: **«Реалізація NFT проекту https://uacatsdivision.com/ на блокчейн платформі TON»**
Зміст ТЧ до дипломної роботи:

1. Індивідуальне завдання
2. Календарний план
3. Анотація
4. Вступ
8. Висновки
9. Список використаних джерел

Дата видачі «___» _____ 2024 р. Керівник _____
(підпис)

Календарний план виконання кваліфікаційної роботи

Тема: «Реалізація NFT проекту https://uacatsdivision.com/ на блокчейн платформі TON»

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання теми кваліфікаційної роботи	18.10.2023	
2.	Аналіз предметної області	30.11.2023.	
3.	Розробка плану і структури роботи	30.12.2023	
4.	Аналіз оригінальної колекції	30.01.2024	
5.	Розробка методів для взаємодії з блокчейном	28.02.2024	
6.	Імплементація алгоритму генерації псевдовипадкових чисел	28.03.2024	
7.	Аналіз та вибір вхідних параметрів	15.04.2024	
8.	Розробка телеграм-боту	13.05.2024	
9.	Попередній захист кваліфікаційної роботи	15.05.2024	
10.	Виправлення зауважень	25.05.2024	
11.	Захист кваліфікаційної роботи	02.06.2024	

Студент: Закутін Т.П.

Науковий керівник: Гороховський К.С.

ЗМІСТ

АНОТАЦІЯ **6**

ВСТУП **8**

РОЗДІЛ 1. ТЕХНОЛОГІЯ БЛОКЧЕЙН

1.1 Опис технології блокчейну. Поняття розподіленого реєстру, консенсусу, хеш-функції 8

1.2 Типи консенсусу 14

1.3 Класифікація блокчейнів 22

1.4 Блокчейн TON та історія його створення 28

РОЗДІЛ 2: Технологія NFT

2.1 Поняття NFT 30

2.2 NFT-колекції та їх приклади 31

РОЗДІЛ 3: РОЗРОБКА ВЛАСНОЇ NFT-КОЛЕКЦІЇ

3.1 Аналіз колекції UACatsDivision 36

3.2 Імплементація алгоритму генерування випадкових чисел та аналіз початкових параметрів 58

3.3 Розгортання колекції на блокчейн-платформі TON 73

ВИСНОВКИ 84

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 85

АНОТАЦІЯ

Дипломна робота присвячена створенню NFT-колекції на блокчейн платформі TON. Описано поняття блокчейну, консенсусу, а також розглянуто різні види цих понять. Означені переваги та недоліки кожного з типів.
Описано поняття NFT, NFT-колекцій та наведені приклади українських та світових колекцій. Окреслена історія створення блокчейну TON, а також його особливості.
Проведено статистичний аналіз колекції UACatsDivision, наведені висновки та припущення щодо алгоритму генерування зображень. Для створення власної колекції створено 49 піксельних зображень, що комбінуються між собою для генерування унікального зображення.
Для генерування використано лінійний конгруентний метод та досліджено різні набори початкових параметрів з метою максимального наближення до заданих ймовірностей. Параметри було оцінено використовуючи 5 різних критеріїв голосування.
Розроблено методи для створення гаманця, NFT та NFT-колекцій на блокчейн платформі TON. Створено телеграм-бота для покупки та мінту NFT. Розроблено алгоритм для хешування вхідних переказів, використовуючи алгоритм хешування SHA-256.
Ключові слова: блокчейн, криптовалюти, TON, колекція, алгоритми генерації, псевдовипадкові числа

The diploma work is dedicated to the creation of an NFT collection on the TON blockchain platform. The concepts of blockchain and consensus are described, and different types of these concepts are considered. The advantages and disadvantages of each type are outlined.

The concept of NFT, NFT collections is described and examples of Ukrainian and world collections are given. The history of the TON blockchain creation and its features are outlined.

The statistical analysis of the UACatsDivision collection is carried out, conclusions and assumptions about the image generation algorithm are presented. To create own collection, there were created 49 pixel images that are combined with each other to generate a unique image.

A linear congruent method was used for the generation and different sets of initial parameters were explored to maximize the approximation to the specified probabilities. The parameters were evaluated using 5 different voting criteria.

Methods for creating a wallet, NFT, and NFT collections on the TON blockchain platform were developed. A telegram bot for buying and minting NFTs was created. An algorithm for hashing incoming transfers using the SHA-256 hashing algorithm was developed.

Keywords: blockchain, cryptocurrencies, TON, collection, generation algorithms, pseudo-random numbers

ВСТУП

За останнє половину століття людство почало використовувати інформаційні технології майже у всіх сферах свого життя. Наприклад, у сфері охорони здоров'я використовується штучний інтелект для вчасного передбачення серйозних захворювань, для навчання використовують спеціальні додатки з інтегрованою ігровою методикою.

Не дивно, що цей прогрес дійшов і до банківської сфери. Каталізатором цього процесу стала світова криза 2008 року, під час якої центральний банк почав друкувати гроші і вливати їх у фінансову системи, що призвело до девальвації фіатної валюти. Саме під час цього періоду було представлено нову піринговy електрону грошову систему, що дістала назву "Bitcoin" і базується на технології блокчейн, яка дозволяє користувачам надсилати та отримувати цифрові активи без посередників. Блокчейн – децентралізована технологія, а отже не існує організації, яка б мала можливість керувати рахунками, блокувати перекази або контролювати ціну криптовалюти.

З плином часу, використання технології блокчейну розширилося до створення на них NFT – унікальних та неповторюваних токенів, що є видом криптовалюти і зберігаються на блокчейні. Ця технологія дозволяє підтвердити право власності над об'єктом, оскільки інформація зберігається на відкритих джерелах.

Мета кваліфікаційної роботи: дослідити особливості блокчейн-мережі TON, та створити на ній NFT-колекцію.

1 ТЕХНОЛОГІЯ БЛОКЧЕЙН

1.1 Опис технології блокчейн. Поняття розподіленого реєстру, консенсусу, хеш-функції

Для кращого розуміння технології блокчейн спочатку розглянемо таке поняття як реєстр. Це є формою систематизації та обліку певної інформації. З давніх часів він використовувався для фіксації та зберігання різної інформації, зокрема про гроші та майно певної особи. Переломним моментом стало впровадження комп'ютерних технологій, що спочатку перетворювало інформацію з паперу у цифровий код. З часом, було створено цифрові розподілені реєстри, які мають можливості та властивості, що виходять за межі традиційних паперових реєстрів.

Розподілений реєстр – аналог бази даних, яка розподілена на декількох мережевих вузлах. Кожен вузол оновлюється незалежно від інших, отримуючи дані з інших вузлів і зберігаючи повну копію реєстру. Основною характеристикою розподіленого реєстру є децентралізація, тобто всі дані не зберігаються в одному місці. Однак інформація всіх вузлів повинна бути точною та актуальною, а цього можна досягти лише за умови консенсусу між усіма вузлами реєстру. Після того, як кожен вузол збирає і додає оновлення до реєстру, проводиться голосування, щоб переконатися, що більшість вузлів схвалюють певну остаточну версію реєстру. Таке досягнення згоди називають консенсусом, цей процес виконується автоматично, за допомогою попередньо заданого алгоритму консенсусу. Щойно консенсус був

утверджений, розподілений реєстр оновлюється та остання версія зберігається в кожному з вузлів.

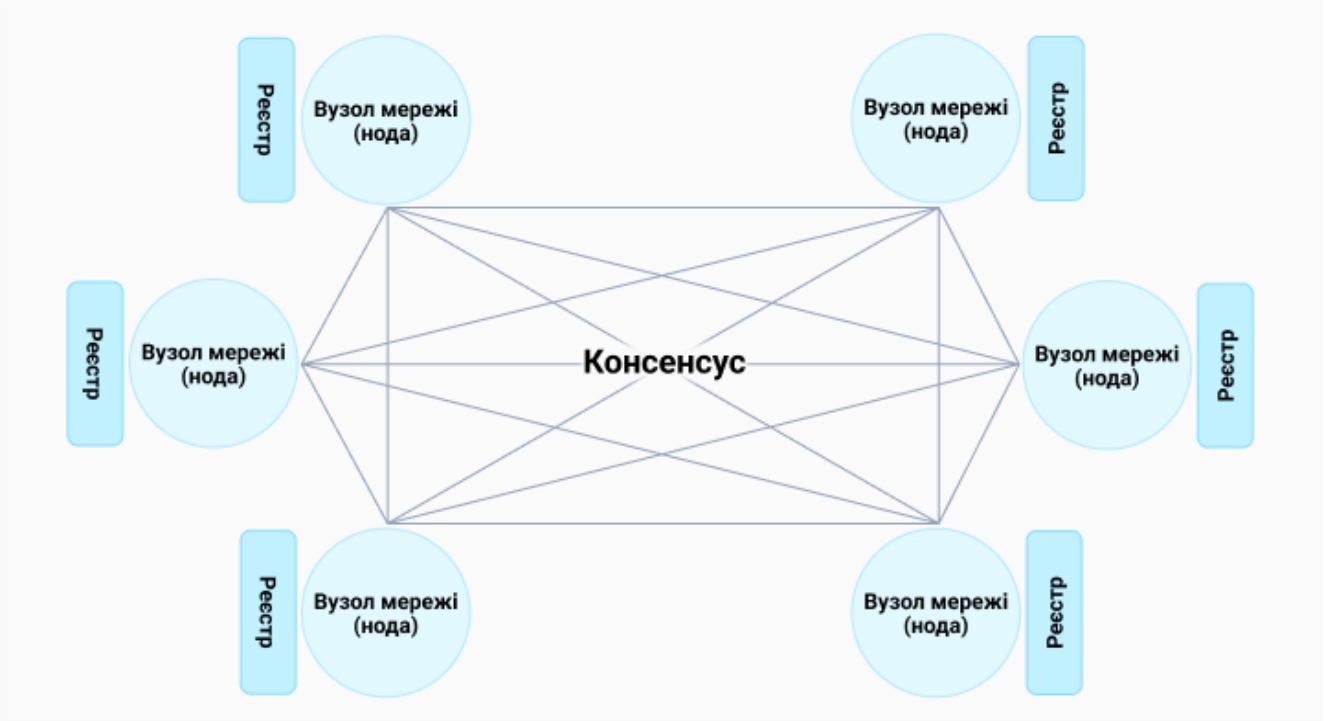

Рисунок 1.1.1 Приклад структури розподіленого реєстру

Власне блокчейн є одним з типів розподіленого реєстру, де для досягнення згоди використовується послідовність блоків, звідки і походить назва, від англ. “block chain” – ланцюг з блоків. Блоки зберігаються у хронологічній послідовності, об’єднані одні з одним та криптографічно захищені. Розглянемо на прикладі блок, який використовується для роботи мережі Bitcoin.

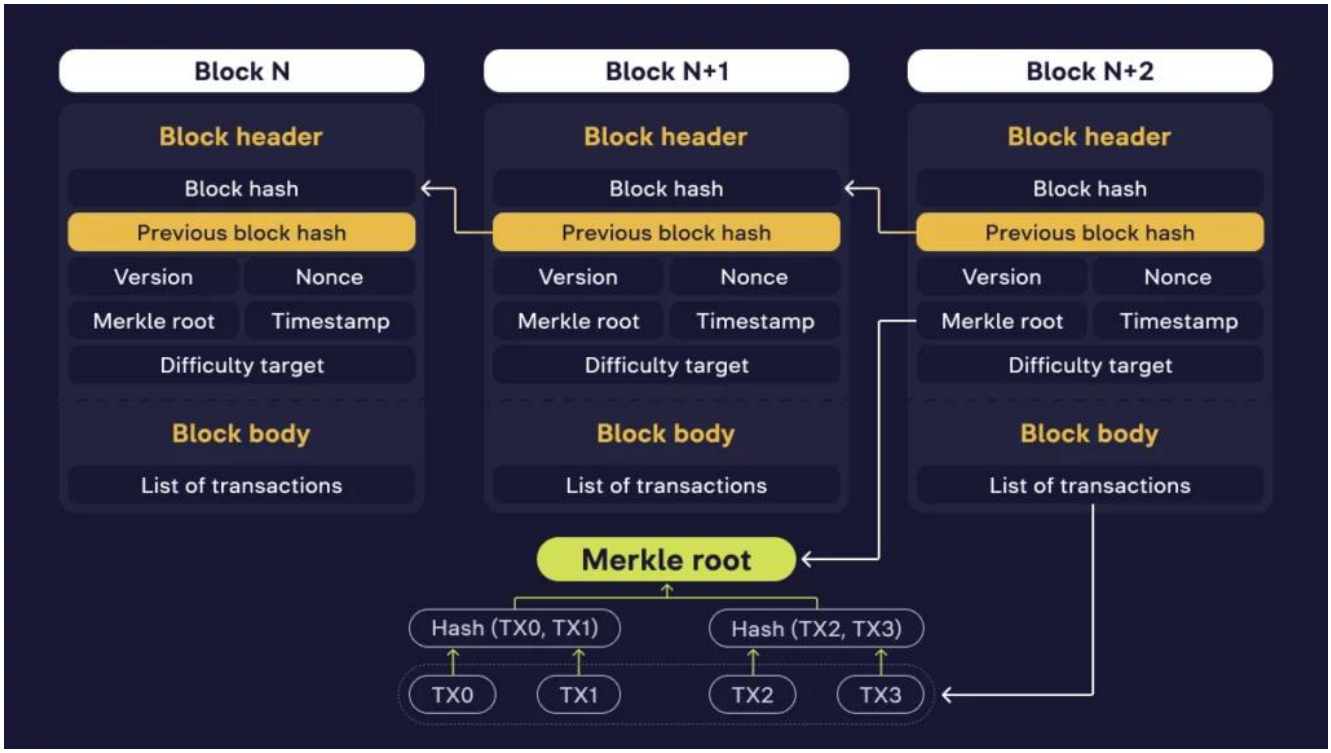

Рисунок 1.1.2 Приклад структури блокчейну

Кожен блок складається з заголовку(Block header), тіло (Block body), що містить список транзакцій, та додаткові службові метадані. Розглянемо елементи, що входять до складу заголовку більш детально.

1. Хеш-код попереднього блоку, який є 32-байтовим хешем попереднього блоку ланцюжка. Хеш-функція - це математичний процес, який перетворює вхідні дані в рядок фіксованої довжини, а результатом є хеш-код. Детерміноване за своєю природою, це перетворення гарантує, що заданий вхідний сигнал завжди призводить до одного і того ж хеш-коду. Крім того, воно є одностороннім, тобто значення хеш-коду не може бути використане для відновлення вихідного вхідного сигналу.

2. Версія блоку (Version) – 4-байтове ціле число, яке представляє останню версію блоку.
3. Випадкове число (Nonce) - це випадкове 4-байтове число, яке блок отримує під час майнінгу.
4. Корінь Меркла (Merkle Root) – 32-байтовий хеш, що є корнем дерева Меркла. У цьому дереві кожен листовий вузол ідентифікується криптографічним хешем блоку даних, а кожен не листовий вузол ідентифікується криптографічним хешем міток його нащадків. За допомогою дерева Меркла можна швидко і безпечно дослідити вміст великої структури даних.
5. Мітка часу (Timestamp) – 4-байтове число, яке представляє час створення блоку.
6. Цільова складність (Target difficulty) - 4-байтове число, що визначає умову складності. Цей параметр регулюється мережею і потрібен для того, щоб новий блок створювався з певним інтервалом.

Як було зазначено раніше, в тілі блоку зберігається список транзакцій, які в свою чергу теж піддаються дії хеш-функції. Спочатку кожна транзакція хешується окремо. Після цього створюються пари хешів транзакцій і знову хешуються. Цей процес триває до тих пір, поки корінь Меркла не буде містити лише один хеш-код. Якщо кількість транзакцій непарна, створюється ідентична копія попереднього хешу і хешується сама з собою. Цей метод дає унікальне хеш-значення, яке можна використовувати для ефективної перевірки цілісності кожної

транзакції та її ідентифікатора, оскільки зміна транзакції призведе до створення нового Кореню Меркла.

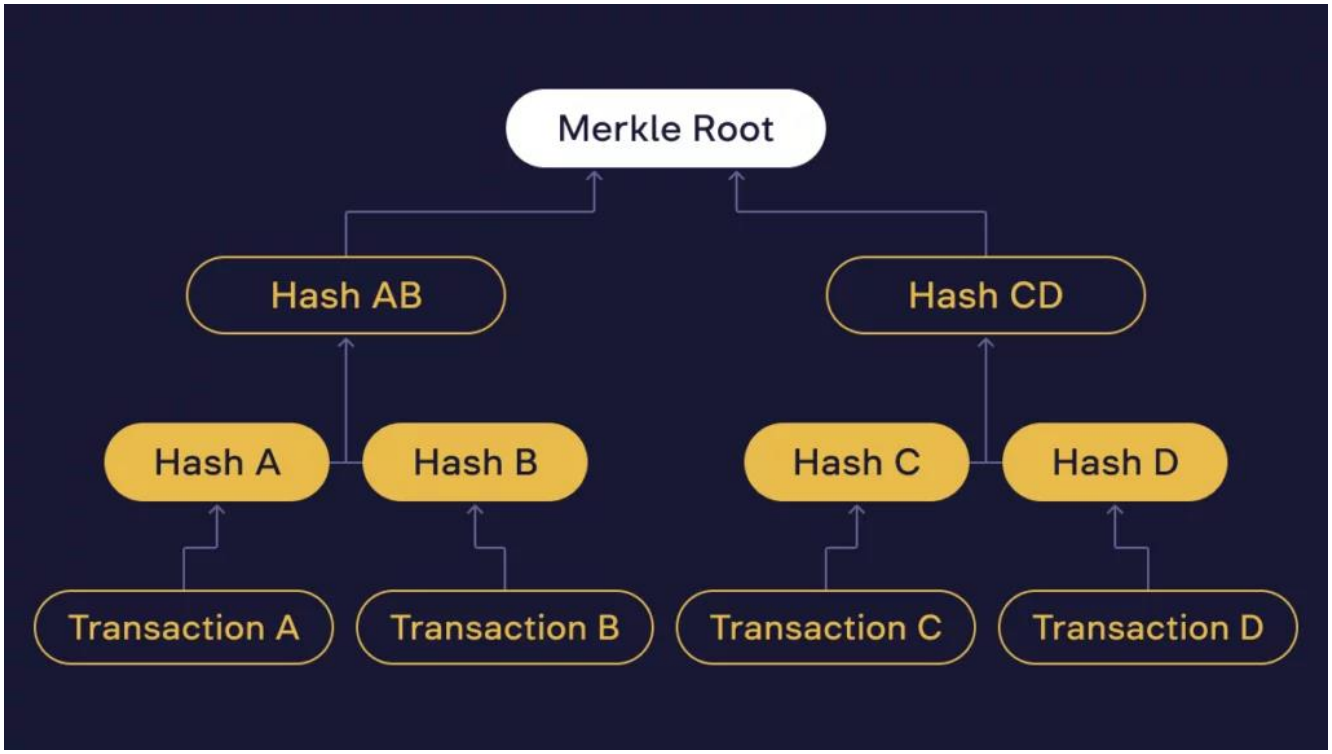

Рисунок 1.1.3 Схема хешування транзакцій

Таким чином, хешування робить технологію блокчейн надійним і непроникним реєстром і гарантує цілісність і безпеку криптовалютних транзакцій.

1.2 Типи консенсусу

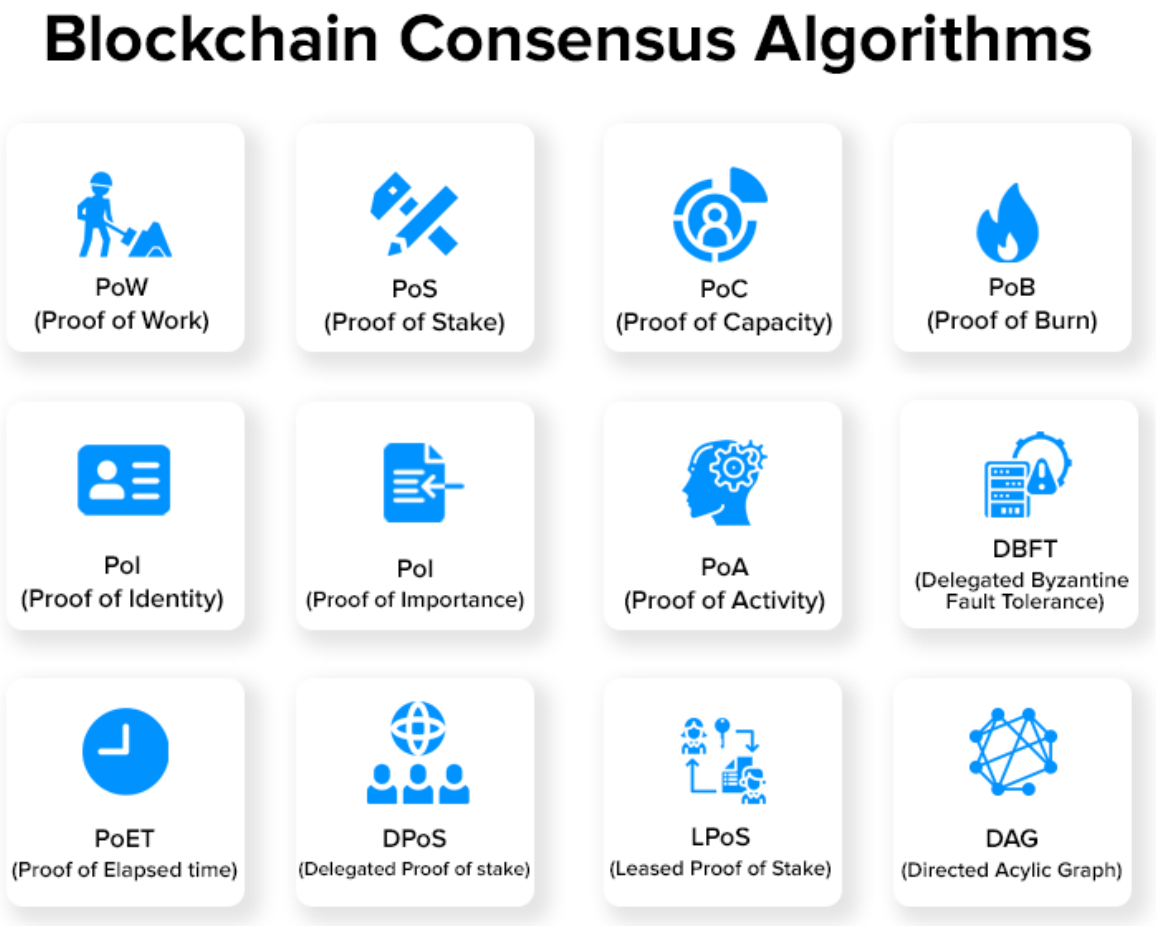

Рисунок 1.2.1 Типи консенсусу у блокчейні

Як згадувалося у роботі раніше, консенсус - це набір правил, що використовується вузлами для досягнення згоди. Розглянемо найбільш популярні їх типи:

1. Підтвердження роботи

   Вперше цей алгоритм був представлений творцем біткоїна Сатоші Накамото як спосіб захистити мережу і запобігти подвійним витратам. Алгоритм PoW вимагає від майнерів вирішувати складні

математичні задачі, відомі як хешування, щоб підтверджувати транзакції і додавати нові блоки до ланцюжка. Хеш-функція, яка використовується в алгоритмах PoW, розроблена таким чином, щоб бути обчислювально складною для вирішення, а це означає, що для вирішення проблеми і додавання блоку в ланцюжок потрібна значна кількість обчислювальної потужності. Майнери змагаються один з одним за вирішення проблеми, і той, хто вирішить її першим, отримує винагороду у вигляді новоствореної криптовалюти.

Безпека цього алгоритму полягає в тому, що проблему хешування складно вирішити, а це означає, що зловмиснику дорого коштуватиме спроба захопити мережу. Для здійснення атаки зловмиснику потрібно мати контроль над значною частиною обчислювальних потужностей мережі, відомих як хеш-рейт, щоб розпочати атаку. Така атака відома як атака 51%, і її важко здійснити, оскільки вона потребує величезної кількості ресурсів.

Однією з причин критики алгоритму PoW є його енергоємність, оскільки майнери повинні використовувати значну кількість обчислювальних потужностей для вирішення хеш-задачі. Це викликало занепокоєння щодо впливу блокчейн-мереж, які використовують PoW, на навколишнє середовище. Однак дехто стверджує, що споживання енергії необхідне для захисту мережі та запобігання атакам.

2. Підтвердження частки (Proof of Stake)

На відміну від Proof of Work (PoW), який вимагає від майнерів вирішення складних математичних задач, PoS покладається на

валідаторів, які володіють певною кількістю криптовалюти для підтвердження транзакцій і додавання нових блоків до ланцюжка.

У мережі PoS валідатори обираються для додавання нових блоків до ланцюжка на основі кількості криптовалюти, якою вони володіють, що називається стейком. Чим більший стейк, тим більший шанс бути обраним для додавання блоку в ланцюжок. Валідатори зацікавлені діяти чесно, оскільки вони ризикують втратити свій стейк, якщо підтвердять шахрайські транзакції або спробують атакувати мережу.

Однією з переваг PoS над PoW є менша енергоємність. PoW вимагає від майнерів значних обчислювальних потужностей для вирішення складних математичних задач, в той час як PoS потребує лише валідаторів для зберігання криптовалюти. Це робить PoS більш екологічно чистим і менш витратним в експлуатації.

Ще однією перевагою PoS є те, що вона сприяє децентралізації. У мережі PoW майнери з найбільшим хешрейтом мають більший контроль над мережею, що може призвести до централізації.

У PoS-мережі валідатори з найбільшою часткою мають більше контролю, але одному валідатору або групі валідаторів складно отримати контроль над мережею, оскільки їм потрібно контролювати значну кількість криптовалюти.

Одним з потенційних недоліків PoS є те, що він може призвести до ситуації «багатий отримує багатство», коли валідатори з найбільшою часткою продовжують заробляти більше криптовалюти, що ускладнює участь у мережі для менших валідаторів.

Однак деякі мережі PoS впровадили механізми для вирішення цієї проблеми, такі як випадковий вибір валідаторів або обмеження кількості криптовалюти, яку може тримати один валідатор.

3. Підтвердження спроможності (Proof of capacity)

   PoC схожий на Proof of Work, оскільки вимагає від учасників розв'язання обчислювальної головоломки для додавання нових блоків до блокчейну, але відрізняється тим, що використовує не обчислювальну потужність, а комп'ютерне сховище.

   У системі PoC учасники виділяють частину простору на жорсткому диску свого комп'ютера, яка слугує графіком, тобто попередньо обчисленим сегментом даних, який може бути використаний для генерування рішення обчислювальної головоломки. Коли до блокчейну потрібно додати новий блок, відбувається пошук розв'язку головоломки у ділянці учасника. Перший учасник, який знайде правильне рішення, може додати новий блок до блокчейну і отримати винагороду у вигляді криптовалюти.

   Використання простору для зберігання даних, а не обчислювальної потужності, робить PoC більш енергоефективним механізмом консенсусу порівняно з PoW. Оскільки учасникам не потрібно постійно виконувати складні обчислення, PoC потребує менше електроенергії і має менший вплив на навколишнє середовище.

   Однак одним з недоліків PoC є те, що він може бути вразливим до певних типів атак, таких як попередні обчислення та атаки Sybil. В атаці попередніх обчислень зловмисник може попередньо обчислити велику кількість ділянок, а потім використати їх для швидкого вирішення обчислювальної головоломки і додавання нових блоків до блокчейну, що дає йому несправедливу перевагу над іншими

учасниками. В атаці Sybil зловмисник може створити кілька облікових записів, щоб збільшити свої шанси знайти рішення головоломки.

Щоб зменшити ці ризики, деякі системи PoC включають додаткові заходи безпеки, наприклад, вимагають від учасників довести, що вони не є зловмисником Sybil, шляхом виконання протоколу «виклик-відповідь».

4. Підтвердження знищення/спалювання (Proof of Burn)

   Замість того, щоб використовувати обчислювальну потужність або простір для зберігання даних, як інші механізми консенсусу, PoB вимагає від учасників спалювати або знищувати токени криптовалюти, щоб довести свою прихильність до мережі.

   Для участі в системі PoB користувач повинен надіслати певну кількість криптовалюти на адресу, де вона буде знищена назавжди. Цей акт знищення криптовалюти називається «спалюванням». Після спалення криптовалюти користувач отримує право додавати нові блоки до блокчейну та отримувати за це винагороду.

   Ідея PoB полягає в тому, що, спалюючи криптовалюту, користувач приносить фінансову жертву і демонструє свою відданість мережі. Це зменшує ймовірність того, що зловмисники спробують атакувати мережу, оскільки для цього їм доведеться спалити значну кількість криптовалюти.

   Однією з потенційних переваг PoB є те, що він може допомогти зменшити інфляцію в криптовалютній екосистемі. Оскільки токени знищуються, а не створюються, загальна пропозиція токенів зменшується, що може допомогти стабілізувати вартість криптовалюти.

Однак, PoB також має деякі недоліки. Один з них полягає в тому, що може бути важко визначити вартість згорілих токенів, оскільки вони назавжди знищуються і не можуть бути відновлені. Це може ускладнити точне вимірювання рівня прихильності та інвестицій в мережу. Крім того, спалювання токенів може розглядатися як марнотратство і недружній до навколишнього середовища.

5. Підтвердження особи (Proof of Identity)

Це тип механізму консенсусу, який спрямований на підвищення довіри, безпеки та автентичності в транзакціях на блокчейні.
PoI працює, вимагаючи від учасників надати цифрову ідентичність, яка пов'язана з процесом перевірки особи в реальному світі. Це може бути досягнуто за допомогою різних методів, таких як видані державою посвідчення особи, біометричні дані або інші форми посвідчення особи, які можна перевірити. Процес перевірки особи гарантує, що кожен учасник є реальною особою, яку можна ідентифікувати, що може допомогти запобігти шахрайській або зловмисній діяльності в мережі.
Після перевірки особи учасника він отримує право брати участь у роботі мережі та сприяти досягненню консенсусу. Цього можна досягти за допомогою різних механізмів, залежно від конкретної мережі блокчейн.

Наприклад, деякі мережі PoI можуть вимагати, щоб учасники голосували за запропоновані транзакції, в той час як інші можуть дозволити учасникам самим пропонувати транзакції.
Однією з головних переваг PoI є те, що вона може допомогти запобігти атакам Sybil, коли один учасник створює кілька облікових

записів у мережі, щоб отримати контроль або маніпулювати системою. Вимагаючи від учасників надати ідентифікаційні дані, які можна перевірити, PoI гарантує, що кожен учасник є унікальним, ідентифікованим об'єктом у мережі.

Однак PoI також має деякі недоліки. Один з них полягає в тому, що може бути важко збалансувати анонімність і конфіденційність з перевіркою особи. Деякі учасники можуть не бажати розкривати свою особистість, щоб зберегти конфіденційність, в той час як інші можуть не мати доступу до необхідних інструментів перевірки особи. Крім того, процес перевірки особи може зайняти багато часу і коштувати дорого, що може відштовхнути деяких учасників від приєднання до мережі.

6. Підтвердження активності (Proof of Activity)

Це гібридний механізм консенсусу, який поєднує в собі Proof of Work і Proof of Stake для досягнення більш безпечної та ефективної мережі блокчейн. PoA був розроблений як альтернатива традиційним механізмам PoW і PoS, які критикували за високе енергоспоживання і ризики централізації.

У PoA мережа блокчейн спочатку використовує PoW для генерації нових блоків, подібно до Bitcoin. Однак, на відміну від Bitcoin, PoA не покладається виключно на PoW для досягнення консенсусу. Замість цього, як тільки новий блок згенеровано, система вибирає випадкову групу валідаторів з мережі на основі їхніх PoS-власників. Ці валідатори повинні підписати новий блок, забезпечуючи другий рівень перевірки консенсусу.

Поєднуючи PoW і PoS, PoA має на меті досягти більш безпечного і ефективного механізму консенсусу. PoW забезпечує складність створення блоків і допомагає запобігти зловмисним атакам, в той час як PoS заохочує довгострокову участь в мережі і запобігає централізації, розподіляючи винагороду на основі розміру частки, якою володіє учасник.

Однією з головних переваг PoA є те, що він менш енергоємний, ніж PoW, оскільки не вимагає від майнерів постійного розв'язання складних математичних задач. Крім того, елемент PoS в PoA допомагає запобігти централізації, оскільки заохочує більшу кількість учасників мати частку в мережі і зменшує стимули для великих майнінг-пулів домінувати в мережі.

Однак PoA також має деякі недоліки. Один з них полягає в тому, що він все ще може бути вразливим до атак 51%, коли група валідаторів вступає в змову з метою маніпулювання мережею. Крім того, PoA є складнішим у реалізації, ніж традиційні механізми PoW або PoS, оскільки він вимагає поєднання обох механізмів.

Отже, вибір правильного алгоритму консенсусу є критично важливим рішенням при розробці мережі блокчейн. Кожен алгоритм консенсусу має свої сильні та слабкі сторони, і вибір неправильного алгоритму може мати серйозні наслідки для безпеки, децентралізації та продуктивності мережі.

Тому при виборі алгоритму консенсусу важливо оцінити вимоги блокчейн-додатку і врахувати такі фактори, як масштабованість, ефективність, безпека і децентралізація. Добре розроблений алгоритм консенсусу може забезпечити ряд переваг, таких як покращена безпека, підвищена ефективність, скорочення часу транзакцій і посилена децентралізація. З іншого боку, погано підібраний алгоритм

консенсусу може призвести до уповільнення транзакцій, збільшення витрат і зниження безпеки.

1.3 Класифікація блокчейнів

Як було описано раніше, блокчейн є розподіленим реєстром даних і єдиним фактором, що може суттєво відрізняти один блокчейн від іншого це доступ до даних. Прийнято вважати, що існують наступні типи:

1. Публічний

Ці блокчейни повністю наслідують ідею децентралізації. Вони не мають жодних обмежень, будь-хто, хто має комп'ютер та інтернет, може брати участь у мережі.

- Оскільки ця мережа є публічною, вона відкрита для громадськості, а це означає, що вона нікому не належить.
- Будь-хто, хто має інтернет і комп'ютер з хорошим апаратним забезпеченням, може брати участь в цьому публічному блокчейні.
- Всі комп'ютери в мережі зберігають копії інших вузлів або блоків, присутніх в мережі.
- У цьому публічному блокчейні ми також можемо виконувати перевірку транзакцій або записів.

Серед переваг цього типу можна виділити:

- Надійність. Вузлам публічного блокчейну не потрібно знати або довіряти один одному, оскільки процедура підтвердження роботи гарантує відсутність шахрайських транзакцій.

- Захищеність. Оскільки блокчейн є публічним, отже має багато користувачів і відповідно багато вузлів, що сильно ускладнює процес взлому.
- Анонімність. Тобто не потребує жодної авторизації для приєднання до мережі вузлів.

Водночас існують і недоліки:

- Довгий час обробки транзакцій. Кількість транзакцій в секунду в публічному блокчейні надзвичайно низька. Це пов'язано з тим, що це велика мережа з великою кількістю вузлів, яким потрібен час для перевірки транзакції та підтвердження роботи.
- Ресурсозатратність. Підтвердження роботи (PoW) є доволі енергозатратним.
- Складність масштабування. Транзакції обробляються і завершуються повільно. Це шкодить масштабуванню, адже чим більше ми намагаємося збільшити розмір мережі, тим повільніше вона буде працювати.

Серед прикладів публічних блокчейнів можна навести загальновідомі: Bitcoin, Ethereum, TON.

2. Приватний

Ці блокчейни мають обмежений доступ або можуть контролюватися навіть однією особою. Хоча цей тип має таку ж структуру як і публічні блокчейни, ця мережа набагато менша. Цей клас часто використовується всередині компаній або організацій.

Переваги:

- Швидкість. Транзакції в приватному блокчейні відбуваються швидше. Це пов'язано з тим, що приватна мережа має меншу кількість вузлів, що скорочує час, необхідний для перевірки транзакції.

- Масштабованість. Розмір блокчейну може бути налаштований згідно до вимог користувача.

Недоліки:

- Менша захищеність. Оскільки така мережа має меншу кількість вузлів, то вона є більш вразливою до хакерських атак.
- Мала кількість вузлів. Якщо деякі вузли відключаються, виникає загроза для роботи всієї системи.

Ethereum, базовий блокчейн ETH, також має приватний блокчейн, який обслуговує підприємства. Він називається Enterprise Ethereum і дозволяє підприємствам використовувати приватні ланцюжки на основі Ethereum та публічну мережу. Інші компанії, такі як IBM, Hyperledger, R3 Corda і Tezos, також надають послуги приватних блокчейнів.

Hyperledger Fabric - це блокчейн-фреймворк для розробки додатків з унікальними функціями управління ідентифікацією та контролем доступу. Ці функції роблять його придатним для компаній, що займаються відстеженням ланцюжка поставок, лояльності та винагород, клірингом та розрахунками за фінансові активи тощо.

Наприклад, Walmart – найбільша мережа гіпермаркетів у США, використовує приватну блокчейн-систему на основі Hyperledger Fabric. Він дозволяє Walmart відстежувати походження своїх товарів. Його постачальники завантажують сертифікати автентичності до реєстру. Це дозволяє компанії відстежувати джерело походження своєї продукції за лічені секунди.

3. Гібридний.

   Цей тип поєднує у собі властивості публічних та приватних блокчейнів. Це дозволяє організаціям створити приватну мережу, що базується на дозволах одночасно з публічною системою, що не вимагатиме дозволів. Таким чином, можна обирати хто матиме доступ до певних даних блокчейну і які дані будуть публічними. У гібридному блокчейні транзакції та записи, як правило, не оприлюднюються, але за необхідності їх можна підтвердити, надавши доступ за допомогою смарт-контракту.

   Переваги:

   - Безпека: Гібридний блокчейн працює в закритому середовищі, що унеможливлює багато типів атак.
   - Економічно ефективність. Транзакції недорогі, швидкі та масштабуються краще, ніж у публічній мережі

Недоліки:

- Відсутність прозорості: Оскільки інформація може бути прихована, цей тип блокчейну не є повністю прозорим.

- Повільний розвиток: Модернізація може бути складною, і користувачі не мають стимулу брати участь у мережі або робити внесок у її розвиток.

Прикладом гібридної блокчейн платформи є Komodo - це блокчейн-платформа, яка використовує унікальний гібридний механізм консенсусу під назвою Delayed Proof of Work (dPoW). Він поєднує в собі безпеку PoW біткоїна з PoS блокчейну Komodo, створюючи вторинний рівень безпеки для мережі.

4. Консорціум

Подібно до того, як гібридний блокчейн має як приватні, так і публічні функції, блокчейн консорціуму, також відомий як федеративний блокчейн, також має ці функції. Однак він відрізняється тим, що в ньому беруть участь різні члени організації, які працюють разом у децентралізованій мережі. Заздалегідь визначені вузли контролюють методи консенсусу в консорціумному блокчейні. У ньому є вузол-валідатор, який відповідає за ініціювання, отримання та перевірку транзакцій. Транзакції можуть бути ініційовані або отримані вузлами-учасниками.

Переваги:

- Безпека. Консорціумний блокчейн є більш безпечним, масштабованим та ефективним, ніж публічний блокчейн. Він, як і приватні та змішані блокчейни, має алгоритми для контролю доступу.

Недоліки:

- Відсутність прозорості: Блокчейн консорціуму має нижчий ступінь прозорості. Якщо до вузла-учасника проникне зловмисник, його все одно можна зламати, а правила блокчейну можуть зробити мережу непрацездатною.

IBM Food Trust є яскравим прикладом консорціумного блокчейну. Він об'єднує кілька суб'єктів у ланцюжку постачання продуктів харчування для підвищення прозорості та відстежуваності. Учасники складаються з виробників, переробників, дистриб'юторів та роздрібних торговців.

Підводячи підсумки, можна зробити висновок, що немає найкращого чи найефективнішого типу блокчейну. Все залежить від потреби конкретного користувача чи організації.

1.4 Блокчейн TON та історія його створення

TON (з англ. The Open Network – Відкрита мережа) - це децентралізована, високопродуктивна блокчейн-платформа, спочатку розроблена командою месенджеру Telegram. Шлях TON розпочався, коли команда розробників Telegram намагалася створити блокчейн-платформу, яка могла б інтегруватися з його сервісом обміну повідомленнями, щоб забезпечити безпечні, швидкі та масштабовані додатки. У 2018 році Telegram випустив технічний документ «Telegram Open Network», привернувши значну увагу та інвестиції завдяки дуже успішному первинному розміщенню монет, яке зібрало 1,7 мільярда доларів. Цей амбітний проект мав на меті включити різноманітні послуги, такі як децентралізована служба доменних імен (DNS), віртуальна приватна мережа (VPN) та платформа для децентралізованих додатків (dApps). Однак проект зіткнувся зі значними юридичними перешкодами, коли в справу втрутилася Комісія з цінних паперів і бірж США , стверджуючи, що продаж власних токенів - «Grams», є незареєстрованою пропозицією цінних паперів. Це призвело до судового рішення, яке фактично зупинило проект у 2020 році. Після цього команда Telegram заявила, що вони припиняють свою участь у розробці проєкту. Незважаючи на відхід від справ Telegram, проєкт продовжував жити завдяки зусиллям спільноти. Розробники та ентузіасти блокчейну з усього світу об'єдналися, щоб продовжити розвиток технології TON під новими іменами, такими як «Free TON», а пізніше просто «The Open Network», підтримуючи

бачення проекту та розширюючи його можливості без прямої участі Telegram. Toncoin – токен вищезгаданого блокчейну відіграє багатогранну роль в його екосистемі. Його основна функція - сприяти роботі dApps, виступаючи в якості палива для мережі. Крім розробки додатків, Toncoin є невід'ємною частиною обробки транзакцій в мережі TON, включаючи міжмережеві обміни, де він використовується для покриття комісій за транзакції. У сфері мережевої безпеки Toncoin використовується в механізмах стейкінгу, які підвищують безпеку блокчейну. Стейкери використовують Toncoin для підтвердження транзакцій і, в свою чергу, забезпечують безпеку мережі та її сервісів. Toncoin також дозволяє брати участь у децентралізованих оплатах за послуги таких сервісів, як TON DNS і TON Proxy, які мають вирішальне значення для підтримки широкого спектру децентралізованих функціональних можливостей мережі. Крім того, Toncoin використовується в управлінні через TON VOTE - платформу, яка дає можливість спільноті брати участь у процесах прийняття рішень. Ця система голосування дозволяє власникам токенів впливати на траєкторію розвитку та зміни політики в екосистемі TON.

Toncoin також інтегрований в Telegram, що дозволяє користувачам здійснювати криптоперекази без комісії будь-якому користувачеві Telegram безпосередньо через додаток. Ця інтеграція також підтримує використання Toncoin в різних децентралізованих додатках Telegram, підвищуючи корисність і доступність криптовалюти для звичайних користувачів.

2. Технологія NFT

2.1 Поняття NFT

Невзаємозамінні токени (NFT) - це унікальні криптографічні токени, які існують на блокчейні і не можуть бути відтворені. Токени - це унікальні ідентифікаційні коди, створені з метаданих за допомогою функції шифрування. Ці токени зберігаються в блокчейні, тоді як самі активи зберігаються в інших місцях. Зв'язок між токеном і активом робить їх унікальними.

NFT можна торгувати і обмінювати на гроші, криптовалюти або інші NFT - все залежить від цінності, яку надають їм ринок і власники. Варто зазначити, що криптовалюта так само є токеном, але на відміну від NFT, вона не є взаємозамінною, тобто якщо взяти дві схожі по зовнішньому вигляду NFT, вони все одно матимуть різні метадані.

NFT створюються за допомогою процесу, який називається “minting” (з англ. карбування, чеканка) , в якому інформація про актив шифрується і записується в блокчейн. На високому рівні цей процес передбачає створення нового блоку, перевірку інформації NFT валідатором і закриття блоку. Процес карбування часто передбачає включення смарт-контрактів, які визначають право власності та керують переказами NFT.

Під час карбування токенів їм присвоюється унікальний ідентифікатор, безпосередньо пов'язаний з однією адресою в блокчейні. Кожен токен має власника, і інформацію про власника є загальнодоступною. Навіть якщо буде викарбувано тисячу NFT одного і того ж найменування, кожен з цих токенів

матиме унікальний ідентифікатор і його можна буде відрізнити від інших.

Ще одною особливістю NFT можна виділити royalty percentage (відсоток винагороди). Це заздалегідь визначений відсоток від вартості продажу NFT, який виплачується автору або первинному власнику активу щоразу, коли він перепродається. Ця функція, по суті, дозволяє авторам продовжувати отримувати вигоду від своїх NFT після їх першого продажу. Роялті за NFT покликані забезпечити творців цифрового мистецтва, аудіо та інших цифрових активів надійним джерелом доходу.

2.2 NFT-колекції та їх приклади

Хоча кожен NFT є унікальним, багато найпопулярніших айтемів є частиною більшого масиву робіт, відомого як колекція NFT. Одна колекція може складатися з багатьох окремих NFT. Часто колекції невзаємозамінних токенів відповідають певній естетичній темі і включають широкий спектр унікальних ознак. Різні комбінації ознак дозволяють творцям створювати колекції NFT з різним ступенем рідкісності.

Однією з перших NFT-колекцій прийнято вважати CryptoPunks. CryptoPunks – це колекція, що складається з 10 тисяч неповторних портретів, що були випущені на блокчейні Ethereum. Кожна з робот – це портрет голови різних створінь розміром 24 × 24 пікселя з певним набором характеристик. Коли проект CryptoPunks був вперше представлений у червні 2017 року, кожен, хто мав гаманець Ethereum, міг безкоштовно

змінтити собі NFT, якщо був готовий заплатити комісію за створення.

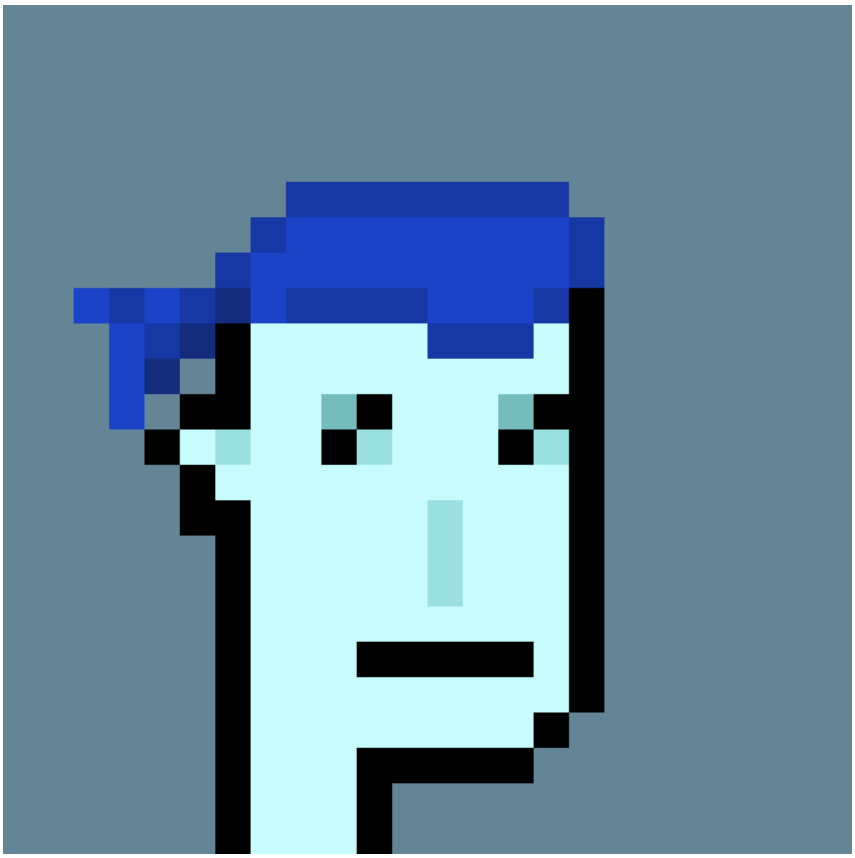

Рисунок 2.2.1 Приклад одного з найдорожчих NFT колекції – Punk #5822 з колекції Cryptopunks, що був проданий за $23,702,160 у лютому 2022 року.

Всього існує 87 атрибутів, які може мати Punk, деякі з них це: колір волосся, окуляри, сигара або сережка. Жоден з Punk'ів не може мати більше ніж 7 унікальних ознак. Це призвело до того, що деякі з них стали доволі унікальними і бажаними для користувачів. Проте існує лише один Punk (#8348) з сімома рисами. У нього є сигарета, сережка, родимка, вставні зуби, класичні окуляри, капелюх і велика борода. Punk також може не мати жодної риси, але більшість з них мають дві або три риси. Вся колекція в основному складається зі зображень чоловік та

жінок, що складають відповідно 6039 та 3840 прикладів. Однак є ще кілька унікальних типів, які вважаються більш цінними через свою рідкісність. Серед них зомбі (88), мавпи (24) та інопланетяни (9).

Attribute Counts

Attribute	#	Avail	Avg Sale	Cheapest	More Examples
0 Attributes	8	1	0	499Ξ	
1 Attributes	333	27	69.67Ξ	68Ξ	
2 Attributes	3560	352	57.67Ξ	51.50Ξ	
3 Attributes	4501	475	61.47Ξ	52.98Ξ	
4 Attributes	1420	164	60.27Ξ	52Ξ	
5 Attributes	166	31	57.24Ξ	89.95Ξ	
6 Attributes	11	3	185Ξ	199Ξ	
7 Attributes	1	0	0		

Рисунок 2.2.2 Розбивка NFT по кількості атрибутів

Варто зазначити, що NFT-колекції також часто використовують як спосіб, щоб зібрати кошти на певні потреби. З початком повномасштабного вторгнення багато українських компаній

почали створювати NFT-колекції та виставляти їх на аукціони, а всі зібрані кошти передавати на потреби армії або волонтерам. Як приклад можна навести компанію “m81.studio”, що займається створенням візуального контенту для відеоігор та мультфільмів.

У квітні 2022 року студія виставила на торговому майданчику для продажу NFT OpenSea NFT із зображенням, поділеним на чотири частини, на якому видно члена політичної партії “ОПЗЖ” Віктора Медведчука, в результаті аукціону вдалося заробити 1,4 тисячі долларів.

Рисунок 2.2.3 NFT від компанії “m81.studio”

Ще одним прикладом є NFT-колекція від української мережі кінотеатрів “Планета Кіно”, яка була створена у квітні 2022. Колекція містить шість смаків NFT-попкорну: «Мужності ЗСУ», «Сталевого характеру Володимира Зеленського», «Українського Криму», «Смузі від бандерівця», «Паляниці» та

«Руского корабля». Стартова ціна кожного з токенів становила 0,1 ETH. Кожен, хто купляв токен отримував довічний абонемент на попкорн у всій мережі кінотеатрів. В результаті вдалося виручити близько 676 тисяч гривень, які було перераховано на потреби ЗСУ.

Рисунок 2.2.4 NFT-попкорн «Мужності ЗСУ»

3. РОЗРОБКА ВЛАСНОЇ NFT-КОЛЕКЦІЇ

3.1 Аналіз проєкту UACatsDivision

UACatsDivision – це українська NFT-колекція, що була створена з метою збору коштів на 4 морських дрона для ВМС ЗСУ. Кіт-поліцейський, кіт-рятувальник, кіт-президент є одними з прикладів і ще багато інших, всі вони уособлюють тих, хто захищає Україну.

Як зазначає Дмитро Курикша, співзасновник проєкту UACatsDivision: «Наша місія — висвітлити героїзм нашого народу в боротьбі за свободу і незалежність України. Ми хочемо піднести котів ЗСУ у вигляді NFT-колекції, яка назавжди залишиться символом нескореності та незламності українського народу. »

Унікальність колекції в тому, що кожна NFT не намальована заздалегідь, а генерується у момент, коли користувач здійснює мінт. Кожна NFT є унікальною, оскільки має певну кількість ознак, що відрізняє їх один від одного. Задній фон, одяг, емоція на обличчі, колір хутра та багато інших. Всі ці елементи накладаються один на одного у визначеному порядку та в результаті формується NFT.

Для того, щоб провести детальний аналіз колекції, я зберу метадані всіх наявних NFT на даний момент. Для цього я використаю наступний код:

```python
import requests
import json
import csv

full_data = []

for i in range(1, 3520):
    url = f"https://ipfs.io/ipfs/bafybeihi32mshqevvdsloadhjns3a7vkeknwósav53prifdi7nxujifjye/{i}"
    response = requests.get(url)
    if response.status_code != 200:
        pass
    else:
        print(f"Parsed #{i}")
        data = json.loads(response.text)
        with open("data.csv", mode='a', newline='') as file:
            writer = csv.writer(file)
            data_row = [data["name"], data["description"], data["image"]]
            for attribute in data["attributes"]:
                data_row.append(attribute["trait_type"])
                data_row.append(attribute["value"])
            writer.writerow(data_row)
```

Рисунок 3.1.1 Код для збору метаданих колекції UACatsDivision

Код складається з циклу, кожен етап якого передбачає запит на інтернет-сторінку, що містить в собі метадані кожного NFT.
Вся інформація зберігається у файл «data.csv».
Посилання вказане у коді, це спеціальне сховище, на якому зберігається більшість NFT, детальніше про нього буде згадано далі у роботі.
Після успішного збору даних було отримано датасет розміром 3519 рядків на 19 колонок.
Розглянемо те, які унікальні ознаки може мати кіт з колекції UACatsDivision. Для цього буде використано наступний код:

```
[4]: lst_unique = []
     for i in range(1, 9):
         lst_unique.append(data[f"Trait_Type_{i}"].unique())
     lst_unique
```

```
[4]: [array(['Background'], dtype=object),
      array(['Fur'], dtype=object),
      array(['Clothes'], dtype=object),
      array(['Face'], dtype=object),
      array([nan, 'Headwear', 'Facewear', 'Armor', 'Hands'], dtype=object),
      array([nan, 'Hands', 'Headwear', 'Facewear'], dtype=object),
      array([nan, 'Hands', 'Headwear'], dtype=object),
      array([nan, 'Hands'], dtype=object)]
```

Рисунок 3.1.2 Код для перевірки унікальних ознак NFT з колекції UACatsDivision

Бачимо, що існують 8 унікальних характеристик, що може мати кіт.

1. Background - задній фон.
2. Fur – шерсть.
3. Clothes – одяг.
4. Face – обличчя.

Ці 4 ознаки є обов'язковими і є у кожного з котів, наступні 4 - опціональні.

5. Headwear - головний убір.
6. Facewear - предмет для обличчя: окуляри, око термінатора, лазер.
7. Hands - предмет, що тримається в руках: в основному різна зброя.
8. Armor – бронежилет.

Далі перевіримо, чи відрізняється кількість ознак для різних NFT, для цього використаємо наступний код:

```
[5]: amount_not_na_values = data[["Value_1","Value_2","Value_3","Value_4","Value_5", "Value_6","Value_7", "Value_8"]].notna().sum(axis=1)
     set(list(amount_not_na_values))

[5]: {4, 5, 6, 7, 8}

[6]: rows_per_count = amount_not_na_values.value_counts()

     plt.figure(figsize=(8, 6))
     bars = rows_per_count.plot(kind='bar', color='skyblue')
     plt.title('Кількість Котів для кожної к-ті ознак')
     plt.xlabel('К-ть ознак')
     plt.ylabel('К-ть котів')
     plt.xticks(rotation=0)
     plt.grid(axis='y', linestyle='--', alpha=0.7)
     for bar in bars.patches:
         plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
                  f'{round(int(bar.get_height())/len(data["Name"])*100, 1)}%',
                  ha='center', va='bottom', color='black')
     plt.tight_layout()
     plt.show()
```

Рисунок 3.1.2 Код для виводу графіка з к-тю ознак і їх відсотковим відношенням

В результаті було отримано наступну гістограму:

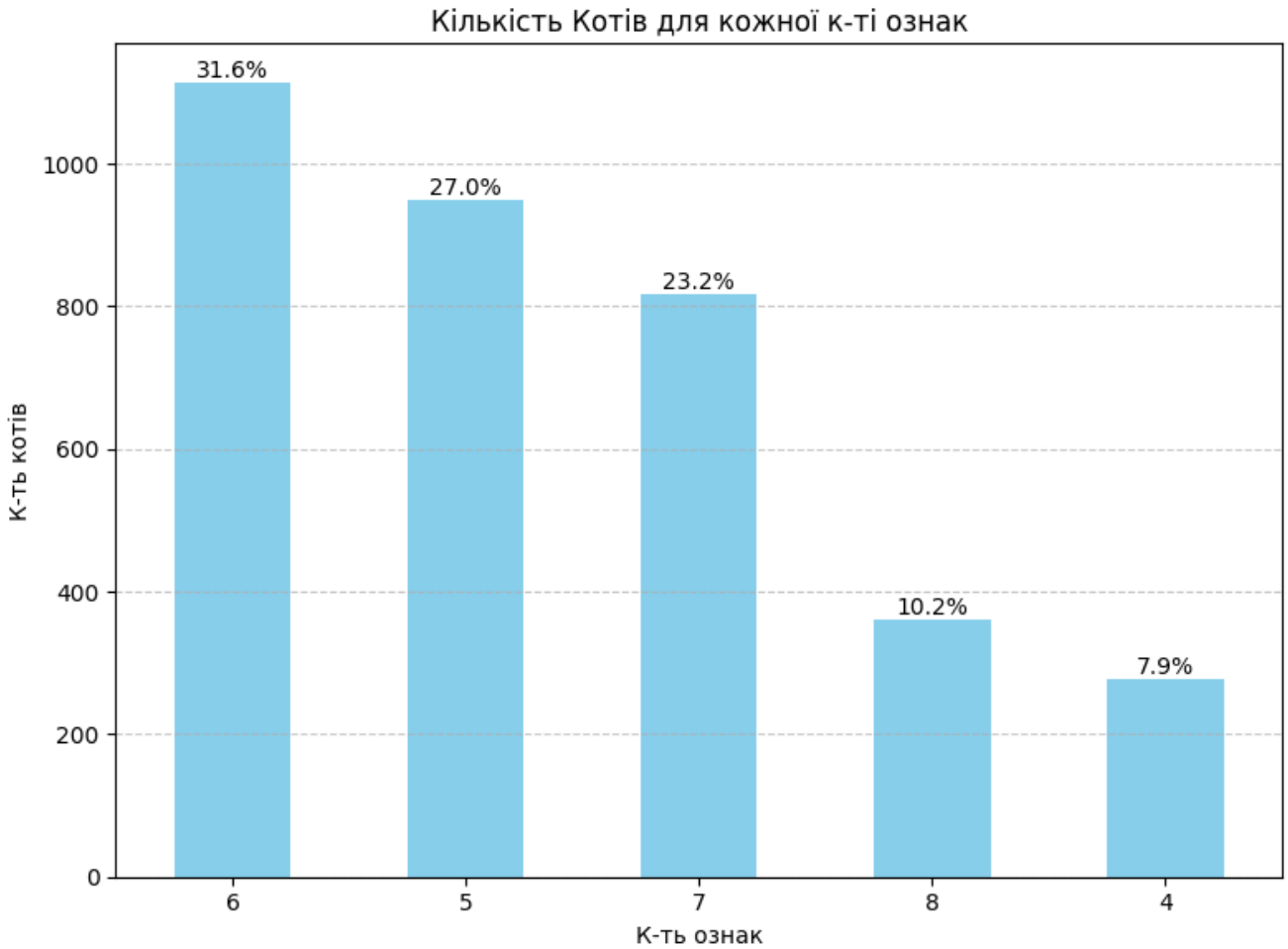

Рисунок 3.1.3 Гістограма к-ті ознак

Видно, що найменше котів, у яких взагалі немає додаткових характеристик - їх 278. Ненабагато більше і котів, що мають максимальну к-ть ознак, їх всього 360. Між кількістю котів, що мають 5, 6 та 7 ознак немає особливого розмаху, їх 950, 1113 та 818 відповідно. Далі спробуємо подивитися на розподіл цих даних, для цього створимо датасет з кількістю ознак у кожного з котів, далі базуючись на ньому створимо 5 окремих списків, в яких буде вказана кількість котів з певною кількістю ознак у проміжках в 153 кота. Для цього використаємо наступний код:

[164]:
```
chunk_size = 153
total_values = 3519
counts_of_4 = []
counts_of_5 = []
counts_of_6 = []
counts_of_7 = []
counts_of_8 = []

for i in range(0, total_values, chunk_size):
    chunk = amount_of_traits.iloc[i:i+chunk_size]
    count_4 = (chunk == 4).sum()
    count_5 = (chunk == 5).sum()
    count_6 = (chunk == 6).sum()
    count_7 = (chunk == 7).sum()
    count_8 = (chunk == 8).sum()
    counts_of_4.append(count_4)
    counts_of_5.append(count_5)
    counts_of_6.append(count_6)
    counts_of_7.append(count_7)
    counts_of_8.append(count_8)

count_4 = pd.DataFrame({'cnt': counts_of_4})
count_5 = pd.DataFrame({'cnt': counts_of_5})
count_6 = pd.DataFrame({'cnt': counts_of_6})
count_7 = pd.DataFrame({'cnt': counts_of_7})
count_8 = pd.DataFrame({'cnt': counts_of_8})
```

[165]:
```
plt.figure(figsize=(15, 12))
plt.subplot(3, 2, 1)  # 3 rows, 2 columns, plot position 1
plt.hist(count_4["cnt"])
plt.title("К-ть котів з 4 ознаками для проміжків у 153 кота")

plt.subplot(3, 2, 2)  # 3 rows, 2 columns, plot position 2
plt.hist(count_5["cnt"])
plt.title("К-ть котів з 5 ознаками для проміжків у 153 кота")

plt.subplot(3, 2, 3)  # 3 rows, 2 columns, plot position 3
plt.hist(count_6["cnt"])
plt.title("К-ть котів з 6 ознаками для проміжків у 153 кота")

plt.subplot(3, 2, 4)  # 3 rows, 2 columns, plot position 3
plt.hist(count_7["cnt"])
plt.title("К-ть котів з 7 ознаками для проміжків у 153 кота")

plt.subplot(3, 1, 3)  # 3 rows, 2 columns, plot position 3
plt.hist(count_8["cnt"])
plt.title("К-ть котів з 8 ознаками для проміжків у 153 кота")

plt.show()
```

Рисунок 3.1.4 Код для групування та виводу гістограм

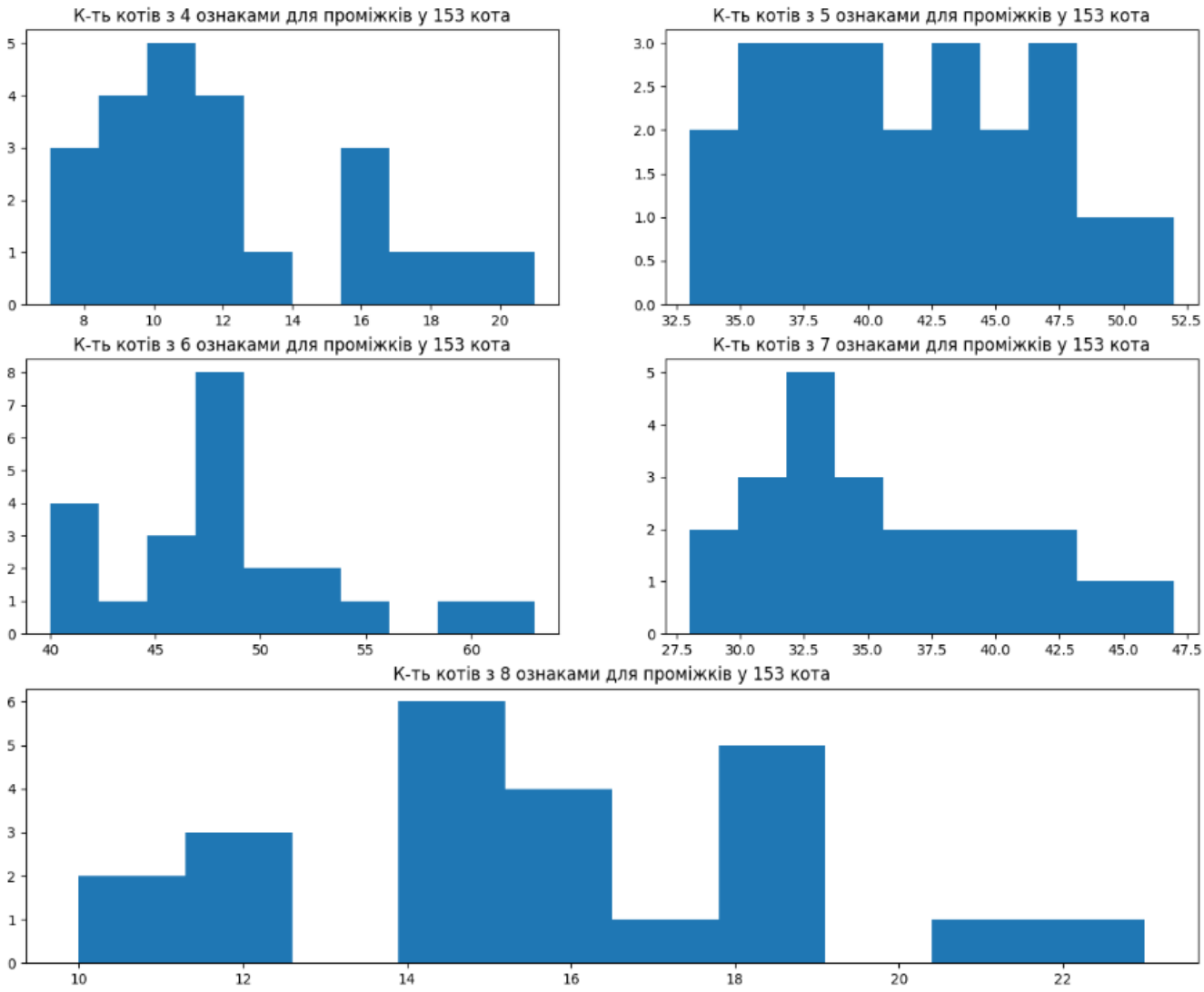

Рисунок 3.1.5 Гістограми після групування

Далі додатково виведемо статистичні показники, що допоможуть зробити висновок про розподіл:

Статистичні функції для котів з 4-ма ознаками:
Середньоквадратичне відхилення: 3.860205405253843
Середньоквадратична похибка: 0.804908468216458
Асиметрія: 0.7711851775473133
Ексцес:-0.3151733284551357

Статистичні функції для котів з 5-ма ознаками:
Середньоквадратичне відхилення: 5.252140538595112
Середньоквадратична похибка: 1.095147006950569

Асиметрія: 0.17459104263706476
Ексцес:-0.9125890043988227

Статистичні функції для котів з 6-ма ознаками:
Середньоквадратичне відхилення: 5.742498113624439
Середньоквадратична похибка: 1.1973936293862573
Асиметрія: 0.7220480090255984
Ексцес:0.3983171322775836

Статистичні функції для котів з 7-ма ознаками:
Середньоквадратичне відхилення: 5.255525983161323
Середньоквадратична похибка: 1.0958529209406151
Асиметрія: 0.4874532535366352
Ексцес:-0.744746288428376

Статистичні функції для котів з 8-ма ознаками:
Середньоквадратичне відхилення: 3.2278432942160946
Середньоквадратична похибка: 0.673051853161527
Асиметрія: 0.30423815828256706
Ексцес:-0.31586374950780627

Можна зробити наступні висновки:

1. Для котів з 4-ма ознаками:

    a) Значення середньоквадратичного відхилення ~3.86 вказує на те, що дані розподілені відносно широко навколо середнього значення.
    b) Значення середньоквадратичної похибки ~0.8 вказує на те, що середнє вибірки є досить точною оцінкою середнього значення генеральної сукупності.
    c) Значення асиметрії ~0.77 вказує на те, що розподіл даних скошений вправо, тобто більшість значень знаходиться лівіше від середнього значення.
    d) Значення ексцесу ~ -0.31 вказує на менш виражений пік даних та менш товсті хвости порівняно з нормальним розподілом.

2. Для котів з 5-ма ознаками:

    a) Значення середньоквадратичного відхилення ~5.25 так само вказує на широкий розкид навколо середнього

значення, навіть більше ніж у попередньої групи котів.
   b) Значення середньоквадратичної похибки ~ 1.09 є доволі точною оцінкою генеральної сукупності, проте менш ніж у котів з 4-ма ознаками.
   c) Значення асиметрії ~ 0.17 означає, що розподіл так само має невелику скошеність вправо, проте вона не є значущою.
   d) Значення ексцесу ~-0.91 означає, що розподіл має ще тонші хвости та плоскіший пік, ніж у попередньої групи.

3. Для котів з 6-ма ознаками:
   a) Значення середньоквадратичного відхилення ~5.74 є найбільшим з досліджуваних груп, дані розподілені ще ширше відносно середнього значення.
   b) Значення середньоквадратичної похибки ~1.19 є найбільшим з досліджуваних, що вказує на те, що середнє значення є ще меншою оцінкою генеральної сукупності.
   c) Значення асиметрії ~ 0.72 є доволі подібним до групи котів з 4-ма ознаками.
   d) Значення ексцесу ~0.39 вказує на те, що розподіл має більш товсті хвости та більш виражений пік, порівняно з нормальним розподілом. Також це є єдиним позитивним значенням зі всіх груп.

4. Для котів з 7-ома ознаками:
   a) Значення середньоквадратичного відхилення ~5.25 є аналогічним для групи котів з 5-ма ознаками.
   b) Значення середньоквадратичної похибки ~1.09 так само є однаковим з вибіркою котів з 5 ознаками.
   c) Значення асиметрії ~0.48 вказує на те, що розподіл має скошеність вправо, проте вона не є дуже великою, порівнюючи з групою котів з 4 та 6 ознаками.
   d) Значення ексцесу ~0.74 вказує на менш загострений пік та тонші хвости. Значення є другим найменшим з усіх груп.

5. Для котів з 8-ома ознаками:
   a) Значення середньоквадратичного відхилення ~3.22 вказує на те, що дані мають помірний розкид навколо середнього значення. Варто підмітити, що значення є

найменшим з усіх вибірок.

b) Значення середньоквадратичної похибки ~0.67 вказує на те, що середнє вибірки є найточнішою оцінки генеральної сукупності, порівнюючи з іншими групами котів.
c) Значення асиметрії ~ 0.30 вказує на невелику скошеність вправо, проте вона є не є особливо значущою.
d) Значення ексцесу ~ -0.31 є ідентичним до групи котів з 4-ма ознаками.

Загалом, можна зробити висновок, що в цілому всі значення дл я груп відрізняються між собою, що може свідчити про те, що не існує чіткого алгоритму розподілу. Можна підмітити, що в усіх групах значення частіше розташовані лівіше від середнього значення, оскільки значення асиметрії є позитивним.

Далі проведемо аналіз ймовірності появи різних ознак,

```
[18]: bg_color = data["Value_1"].value_counts()
      bg_color
```

почнемо з заднього фону.

Рисунок 3.1.6 Код для виводу типів та к-ті заднього фону

Було отримано наступні результати:

Ash Gray (Попелясто-сірий) - 388
Light Silver (Світло-сріблястий) - 366
Pastel Blue (Пастельно-блакитний) - 363
Camouflage Green (Камуфляжно-зелений) - 359
Yellow (Жовтий) - 350
Dark Vanilla (Темно-ванільний) - 347
Philippine Gray (Філіппінський сірий) - 346
Olivine (Олівін) - 341
Sky Blue (Небесно-блакитний) - 330
Ebony (Колір чорного дерева) - 329

Для ліпшого візуального сприйняття виведемо кругову діаграму видів заднього фону, для цього буде використано наступний код:

```
[19]: colors = {"Ash Gray" : "#B2BEB5" , "Light Silver": "#E1E1E1", "Pastel Blue": "#AEC6CF", "Camouflage Green": "#78866b",
          "Yellow": "#FFFF00", "Dark Vanilla": "#D1BEA8", "Philippine Gray": "#8C8C8C", "Olivine": "#9ab973", "Sky Blue": "#87CEEB",
          "Ebony": "#555D50"}
      plt.figure(figsize=(8, 8))
      plt.pie(bg_color, labels=bg_color.index, autopct='%1.1f%%', colors=[colors[val] for val in bg_color.index])
      plt.title('Кругова діаграма для кольорів заднього фону')
      plt.show()
```

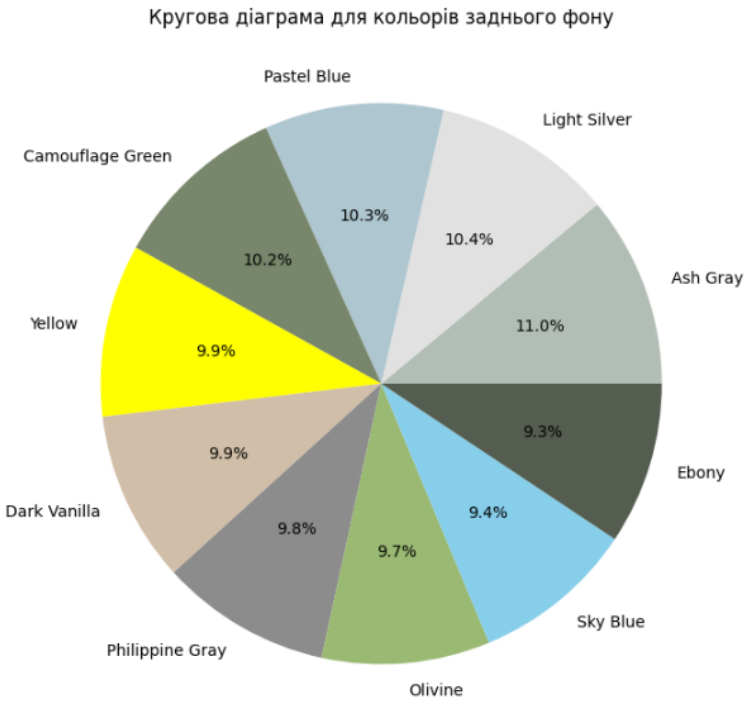

Рисунок 3.1.7 Код для виводу кругової діаграми кольорів заднього фону

Можна зробити висновок, що всі 10 кольорів ймовірніше за все мають шанс випадіння 10%.

Перейдемо до аналізу хутра котів.

```
[20]: fur_type = data["Value_2"].value_counts()
      fur_type
```

Рисунок 3.1.8 Код для виводу типів хутра та к-ті котів з кожним типом

Було отримано наступні результати:

Light Gray (Світло-сірий) – 759.

Gray (Сірий) - 747

Dark Gray (Темно-сірий) - 676

Red (Червоний) - 508

White Lynx (Колір білої рисі) - 202

Black (Чорний) - 200

Lynx (Колір рисі) – 167

Kitty Girl (Дівчина-Кішка) – 154

Leopard (Леопардовий) – 74

AZOVSTAL (Азовсталь) – 32

Так само виведемо кругову діаграму:

```
[21]: colors = {"Light Gray" : "#D3D3D3" , "Gray": "#808080", "Dark Gray": "#5A5A5A", "Red": "#FF0000",
              "White Lynx": "#F7F7F7", "Black": "#454545", "Lynx": "#b59378", "Kitty Girl": "#fadfec", "Leopard": "#a37319",
              "AZOVSTAL": "#aaa9ad"}
      plt.figure(figsize=(8, 8))
      plt.pie(fur_type, labels=fur_type.index, autopct='%1.1f%%', colors=[colors[val] for val in fur_type.index])
      plt.title('Кругова діаграма для кольорів хутра')
      plt.show()
```

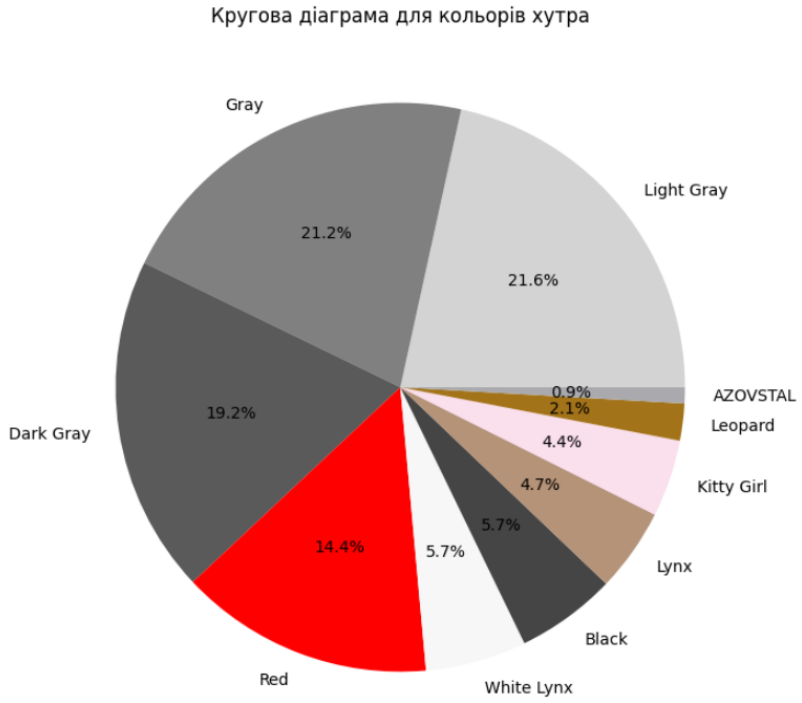

Рисунок 3.1.9 Код для виводу кругової діаграми видів хутра

Можна зробити висновок, що по шерсті вже існують певні задані відсотки ймовірності випадіння для 10 варіантів.
Э три основні шерсті: сіра, світло-сіра та темно-сіра, які мають ймовірність близько 20%. Далі йде червона, що має шанс приблизно 15%. Типи шерсті такі як: біла рись, чорний, рись, дівчина-кішка мають приблизно однаковий шанс ~5%.
Найбільш рідким є тип шерсті: ASOVSTAL, що має приблизно 1% і є всього у 32 котів.

Далі проаналізуємо типи одягу:

```
[22]: clothes_type = data["Value_3"].value_counts()
      clothes_type
```

Рисунок 3.1.10 Код для виводу типів одягу та к-ті котів з кожним типом

Отримали наступні результати:
Armed Forces T-shirt (Футболка Збройних сил) – 227
TRO T-shirt Yellow (Футболка члена ТРО жовта) – 203
Hoodie (Худі) - 182
White Embroidery (Біла вишиванка) - 101
Linen Embroidery (Льняна вишиванка) – 99
Police T-shirt (Поліцейська футболка) - 95
Rescuer Jacket (Куртка рятувальника) – 82
Green Embroidery (Зелена вишиванка) – 68
Medical T-shirt (Медична футболка) – 45
UNITED24 Hoodie (Худі UNITED24) – 33
General Coat (Генеральське пальто) – 20
Admiral Coat (Адміральське пальто) - 17
The President Hoodie (Президентське худі) - 1

Приклади одягу на бульбашковій діаграмі:

Рисунок 3.1.11 Бульбашкова діаграма з прикладами одягу

Обчислимо ймовірності з якими розподілені види одягу, для цього використаємо наступний код:

```
[23]: plt.figure(figsize=(14, 6))
bars = clothes_type.plot(kind='bar', color='skyblue')
plt.title('')
plt.xlabel('Тип одягу')
plt.ylabel('К-ть котів')
for bar in bars.patches:
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(),
             f'{round(int(bar.get_height())/len(data["Value_3"])*100, 2)}%',
             ha='center', va='bottom', color='black')
plt.show()
```

Рисунок 3.1.12 Код для графіку ймовірностей типів одягу

Отриманий результат:

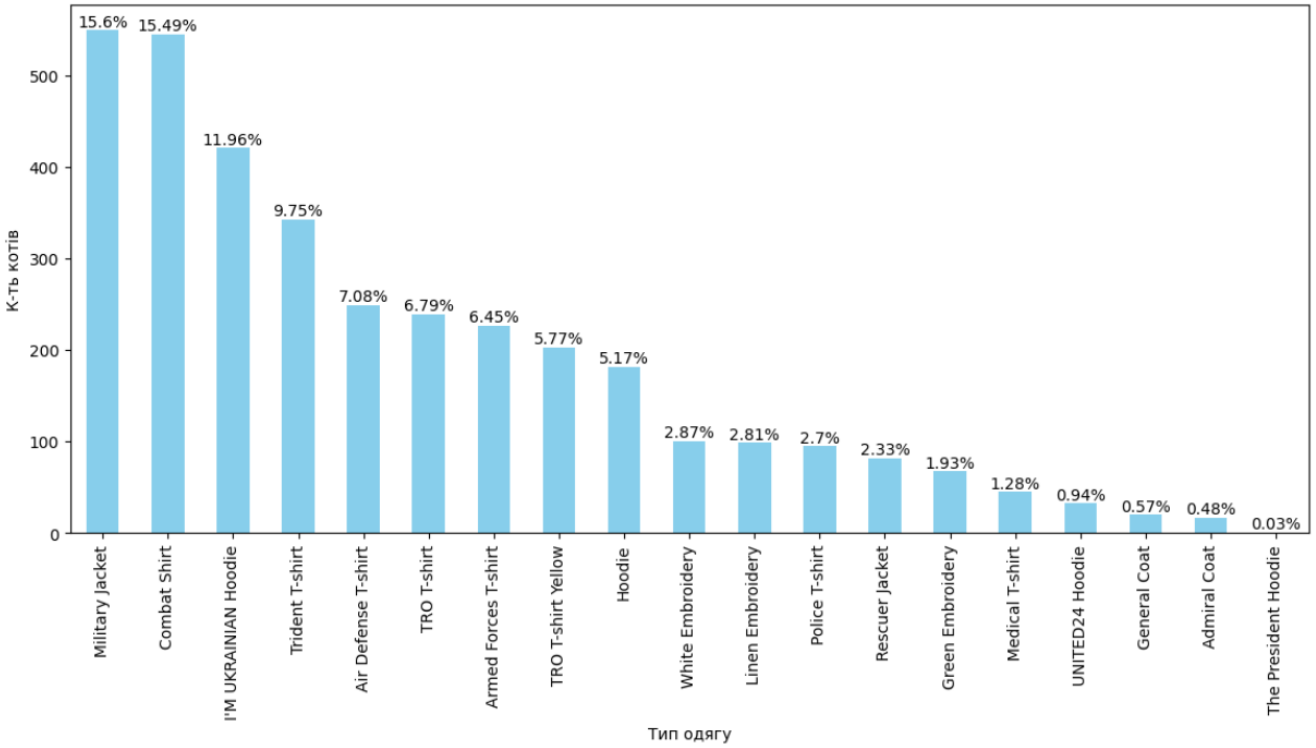

Рисунок 3.1.13 Гістограма ймовірностей типів одягу

Бачимо, що найбільшу ймовірність мають: військова куртка та бойова сорочка, по ~15%.
Далі йдуть худі I’M UKRAINIAN та футболка з тризубом, шо мають по 12 та 10 відсотків відповідно.
Наступною категорією з приблизно однаковим шансом йдуть: футболка протиповітряної оборони, футболка ТРО, футболка Збройних сил, футболка ТРО жовта, та худі, що мають від 5 до 7%.
Наступна категорія включає в себе: вишиванку, льняну вишиванку, поліцейську футболку, куртку рятувальника та зелену вишиванку, що мають від 2 до 3%.
І остання категорія, що має ймовірність не більше 1%, вона складається з: медичної футболки, худі UNITED24, генеральського пальто та адміральського мундиру.
Найунікальнішим предметом одягу є Президентське худі, що має шанс лише 0.03% та існує поки лише у одного кота.

Проведемо аналіз для типів обличчя/емоції:

```
[24]: faces_type = data["Value_4"].value_counts()
      faces_type
```

Рисунок 3.1.14 Код для виводу типів обличчя

Отримані типи:

Calm Yellow Eyes (Спокійні жовті) – 129, Sad Yellow Eyes (Сумні жовті) – 114,
Grin Green Eyes (Усміхнені зелені) – 113, Calm Green Eyes (Спокійні зелені) – 113,
Tongue & Yellow Eyes (Язик та жовті) – 109, Upset Blue Eyes (Сумні сині) – 107,
Cute Blue Eyes (Милі сині) – 106, Grin Yellow Eyes (Усміхнені жовті) – 105,
Funny (Смішні) – 104, Incredulous Blue Eyes (Неймовірні блакитні) – 103,
Fury Black (Люті чорні) – 103, Terrible Demon (Страшні демонські) – 103,
Grin Vampire Eyes (Усміхнені вампірські) – 103, Sad Blue Eyes (Сумні сині) – 102,
Terrible Sauron (Страшні сауронівські) – 100,
Tongue & Vampire Eyes (Язик і вампірні) – 98,
Tongue & Green Eyes (Язик і зелені) – 98, Upset (Засмучені) – 97,
Terrible Black (Страшні чорні) – 97, Calm Blue Eyes (Спокійні сині) – 96,
Sad Green Eyes (Сумні зелені) – 96, Incredulous Vampire (Неймовірні вампірні) -95,
Tongue & Blue Eyes (Язик і сумні) – 93, Cute Green Eyes (Милі зелені) – 92,
Incredulous (Неймовірні) – 91, Fury Yellow Eyes (Люті жовті) – 90,
Fury Sauron (Люті сауронівські) – 90, Upset Green Eyes (Сумні зелені) – 87,
Cute Yellow Eyes (Милі жовті) – 86, Leopard (Леопардові) – 74,
Fury Dragon (Люті дракончі) – 57, Incredulous Dragon (Неймовірні дракончі) –45,
Fury White Walker (Люті білі) – 41, Kittie Girl Bloody Eyes (Жіночі криваві) – 36,
Kittie Girl Green Eyes (Дівчачі зелені) – 36,
Kittie Girl Vampire Eyes (Дівчачі вампірські) – 31,
Y&B Fury Demon (Жовто-сині демонські люті) – 31,
Terrible Beast (Жахливі звірячі) – 30, Kittie Girl Blue Eyes (Жіночі сині) – 29,

Incredulous Beast (Неймовірні звірячі) – 29,

Terrible White Walker (Жахливі білі) - 28, Terrible Dragon (Жахливі драконячі) - 25,

Y&B Fury Beast (Жовто-блакитні люті звірячі) – 24, Fury Beast (Люті звірячі) – 23,

Kittie Girl Yellow Eyes (Жіночі жовті) – 22,

Y&B Terrible Walker (Жовто-блакитні жахливі) - 21

Y&B Terrible Demon (Жовто-блакитні демонські) – 20.

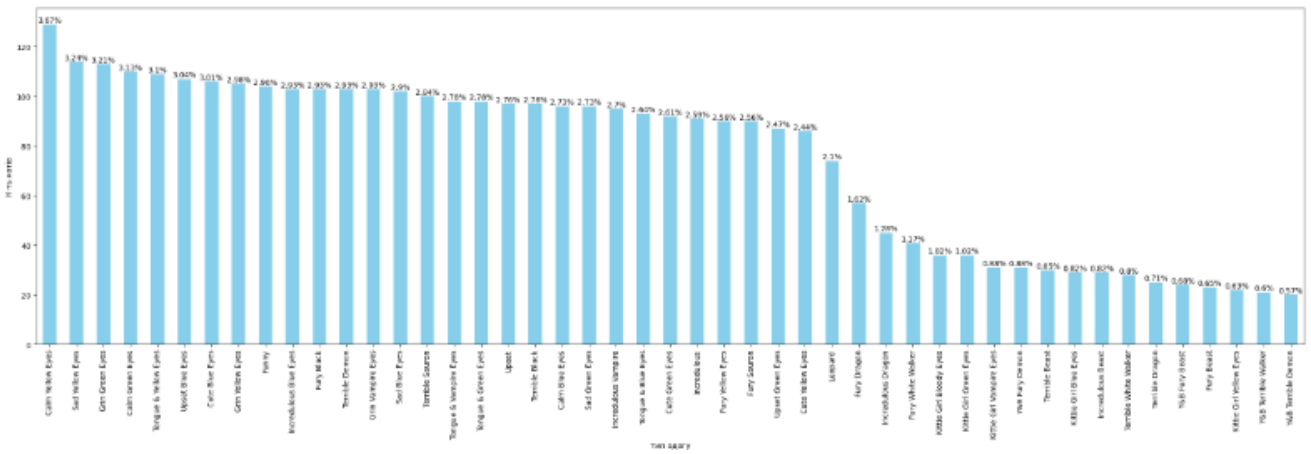

Рисунок 3.1.15 Гістограма ймовірностей типів очей

Бачимо, що з існуючих 47 видів обличчя 36 з них мають приблизно однаковий шанс - 2,5%. А інші 11 мають шанс ~1%. Найбільш рідкими є наступні:

Y&B Fury Beast. 24 екземпляри

Fury Beast. 23 екземпляри

Kittie Girl Yellow Eyes. 22 екземпляри

Y&B Terrible Walker. 21 екземпляр

Y&B Terrible Demon. 20 екземплярів

Рисунок 3.1.16 NFT з найбільш рідкими типами очей

Для аналізу додаткових ознак було проведено процес обробки даних, оскільки ознаки були розташовані у різних стовпцях. В результаті було сформовано окремі списки для додаткових ознак. Для ознаки Armor (Броня) було отримано наступні типи та кількість осіб:

1. Armor Plate (Бронепластина) - 554
2. Medical Armor Plate (Медична бронепластина) - 242
3. Rescue Armor Plate (Рятувальна бронепластина) - 107
4. Press Armor Plate (Бронепластина для представників преси) – 24

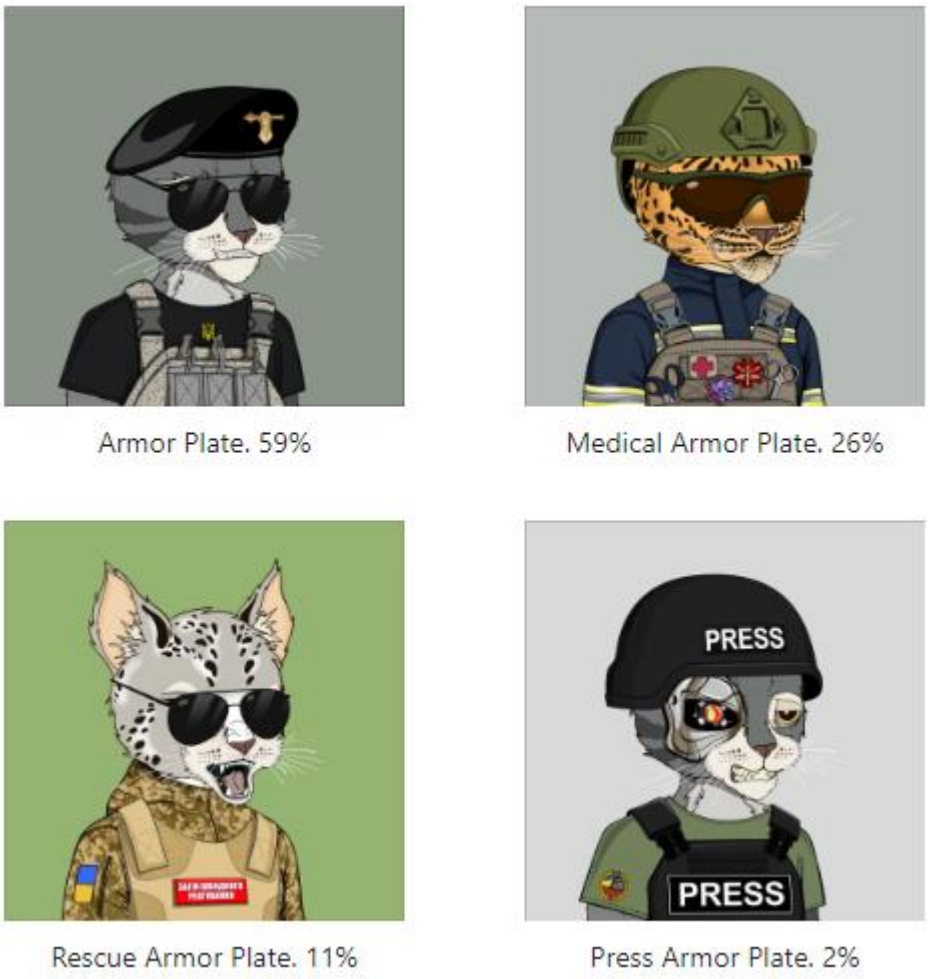

Рисунок 3.1.17 Приклади з усіма видами броні

Загалом бронежилет є у 927/3519 котів, що становить 27%.

Щодо ознаки Facewear (Аксесуари для обличчя) існують наступні типи:

Glasses (Окуляри) - 517

Aviator Glasses (Окуляри-авіатори) - 423

Tactical Glasses (Тактичні окуляри) - 306

Mask (Маска) - 221

Terminator Eye (Око термінатора) - 44

Laser Eyes (Лазер з очей) - 39

Рисунок 3.1.18 Приклади з усіма видами аксесуарів для обличчя

Загалом аксесуари для обличчя є у 1550/3519 котів, що становить 44%.

Розглядаючи наступну ознаку: Headwear (Головні убори), можна виділити наступні 37 типів:

Separate Regiment Beret (Берет Окремого Полку) - 82
Engineer Troops Beret (Берет інженерних військ) - 72
Special Operations Beret (Берет Сил Спеціальних операцій) - 71
Air Force Beret (Берет Повітряних сил) - 66
Mechanized Troops Beret (Берет механізованих військ) - 63
Tactical Helmet (Тактичний шолом) - 63
General Air Force Beret (Берет генерала Повітряних сил) - 62
General Ground Forces Beret (Берет генерала Сухопутних військ) - 61
Marine Corps Beret (Берет морської піхоти) - 60
Navy Beret (Берет спеціальних сил морської піхоти) - 59
Police Cap (Поліцейський кашкет) - 59
Tank Troops Beret (Берет танкових військ) - 57
Missile Forces and Artillery Beret (Берет Ракетних військ та артилерії) - 57
Mountain Infantry Beret (Берети гірсько-штурмових військ) - 57
Radio Technical Forces Beret (Берет радіотехнічних військ) - 56
General Navy Beret (Берет генерал ВМС) - 54
Press Helmet (Шолом для представників преси) - 53
Military Cap (Кашкет військовий) - 52
Military Police Beret (Берет Військової служби правопорядку) - 52
Air Defense Beret (Берет військ протиповітряної оборони Сухопутних військ - 52
Separate Security Brigade Beret (Берет Окремої охоронної бригади) - 48
Airborne Beret (Десантний берет) - 44
Black Helmet (Чорний шолом) - 43
Tankman Helmet (Шолом танкіста) - 43
Basic Helmet (Базовий шолом) - 43
Helmet with Addons (Шолом із доповненнями) - 39
Firefighter Helmet (Шолом пожежника) - 38
Helmet with Mask (Шолом із маскою) - 38
Admiral Cap (Кашкет адмірала) - 37
Helicopter Pilot Helmet (Шолом пілота гелікоптера) - 35
Panama (Панама) - 33
Medical Helmet (Медичний шолом) - 31
Poppies Wreath (Маковий вінок) - 30
Pilots Helmet (Шолом пілота) - 22
General Cap (Генеральський кашкет) - 20
Sunflower Wreath (Соняшниковий вінок) - 19
Kalush Band Hat (Шапка гурту «Калуш») - 1

Існує 37 видів головних уборів, з яких найбільш рідкими є:

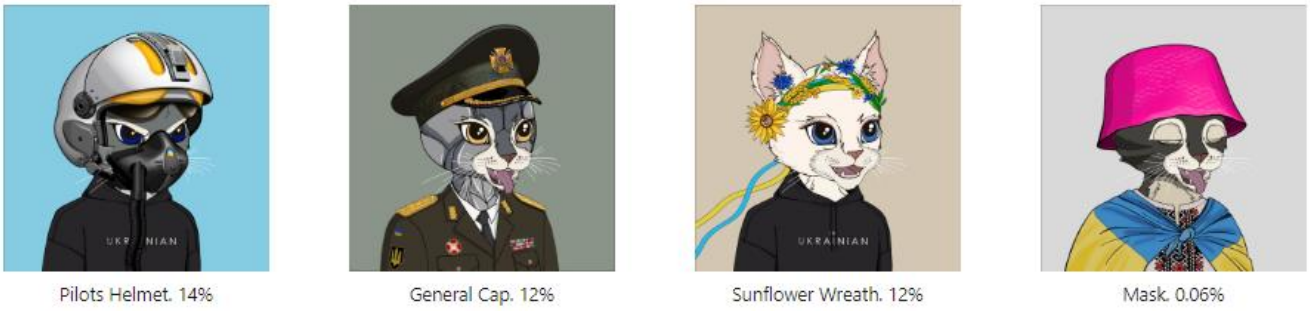

Рисунок 3.1.19 Приклади з найрідшими головними уборами

Щодо останньої ознаки – Hands (Предмети в руках), було виокремлено наступні типи:

Drone (Дрон) - 307
M16 Rifle (Гвинтівка M16) - 257
Steyr AUG Rifle (Гвинтівка AUG) - 257
Javelin (Джавелін) - 244
Gun (Пістолет) - 231
AK-74 Modernized (Гвинтівка AK-74) - 231
Gun with Silencer (Пістолет з глушником) - 217
Molotov Cocktail (Коктейль Молотова) - 182
Fuck you (Жест) - 175
Holding Grenade (Тримаючи гранату) - 125
Mine Detector (Детектор мін) - 118
Combat Knife (Бойовий ніж) - 117
Flag of Ukraine (Флаг України) - 108
Watermelon from Kherson (Херсонський кавун) - 84
Giving Grenade (Віддаючи гранату) - 63
Cossack Saber (Козацька шабля) - 60
Potato Masher (Картоплем’ялка) - 44
Victory Sign (Жест перемоги) - 1

Існує 18 видів предметів, що знаходяться у лапках кота, найбільш рідкі з яких:

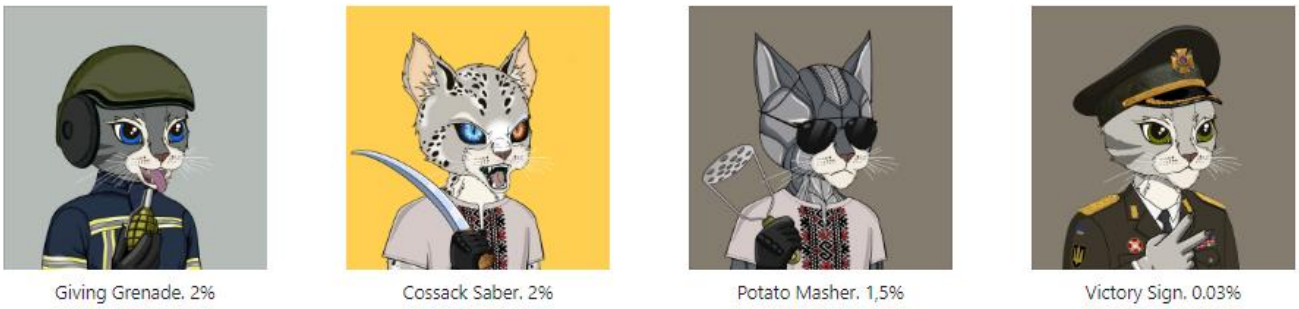

Рисунок 3.1.20 Приклади з найрідшими жестами/предметами у руках

3.2 Імплементація алгоритму генерування випадкових чисел та аналіз початкових параметрів

Першим етапом створення колекції є вибір алгоритму генерації контенту кожної NFT. У рамках цієї роботи натхненням стали дві колекції: «UACatsDivision» та «CryptoPunks». Кожна NFT буде генеруватись шляхом накладання один на одного піксельних об’єктів. В результаті контентом для NFT буде кіт 32x32 пікселі з українською символікою.

Для цього було розроблено наступні типи елементів:

1. Колір заднього фону – 9 варіантів
2. Колір форми кота – 7 варіантів
3. Колір хутра – 7 варіантів
4. Обличчя – 3 варіанта
5. Окуляри – 4 варіанти
6. Головний убір – 9 варіантів
7. Предмет на хвості – 10 варіантів

Рисунок 3.2.1 Приклади елементів для генерації контенту для NFT

Для того, щоб кожна NFT була унікальною, було розроблено алгоритм для вибору елементів. Його суть полягає в наступному:

1. 4 елемента є обов'язковими: колір заднього фону, колір форми кота, колір хутра та тип обличчя. Ці елементи присутні у кожного з «котів».
2. Наступні три елементи: окуляри, головний убір та предмет на хвості є опціональними. NFT може мати як один з цих елементів, всі або жодного.
3. Для кожного елемента є задана ймовірність вибору, що визначена числом від 0 до 1000. Детальніше про алгоритм вибору буде описано у наступному абзаці.
4. Для елементів, що є вибірковими, є задана ймовірність за якими вони будуть чи не будуть додані до фінального контенту NFT. Ця ймовірність визначена числом від 0 до 100 та співпадає з приблизною ймовірністю, що була обрахована для колекції «UACatsDivision».

Щодо алгоритму вибору чисел було обрано лінійний конгруентний метод. Це є одним з найстаріших та найбільш відомих алгоритмів для генерації послідовності псевдовипадкових чисел. Формула, за якою генерується елемент послідовності наступна:

$X_{n+1} = (a * X_n + c)\ mod\ m$, де

$m -$ модуль, $0 < m$,

$a -$ множник, $0 < a < m$,

$c -$ приріст, $0 \leq c < m$,

$X_0 -$ початкове значення, $0 \leq X_0 < m$

Оскільки алгоритм генерує число без прив'язки до певних діапазонів, буде запропонована наступна формула для генерації чисел у заданому діапазоні.

Нехай

$X_n \notin [R_{min};\ R_{max}]$, тоді $X_n = \frac{X_n}{m-1} * (R_{max} - R_{min}) + R_{min}$, де

$X_n -$ число з послідновності згенерованих чисел,

$R_{min}, R_{max} -$ відповідно нижня та верхня межа діапазону

$m -$ модуль, $0 < m$

Оскільки $X_n \in [0; m-1]$, то $\frac{X_n}{m-1}$ є нормалізацією числа до діапазону $[0; 1]$ і далі помноживши результат на розкид діапазону і додавання нижньої межі отримуємо число, що є масштабованим числом для означеного проміжку.

Лінійний конгруентний метод базується на певному початковому числі, яке використовується для генерації наступних елементів послідовності. Для вибору цього числа було використано python-бібліотек os та time.

Формула для вибору початкового числа:

$$X_0 = \frac{os.getpid + time.time}{2034259200000000000\ +\ 4294967295} * m$$

os.getpid() – цей метод повертає ціле число, що позначає ідентифікатор поточного процесу в операційній системі. Для операційної системи Windows 11 це значення знаходиться у діапазоні $[0; 4294967295]$.

time.time_ns() – цей метод повертає кількість наносекунд на поточний момент, починаючи від дати 1 січня 1970-го року. Для означення діапазону приймемо найбільшим значенням – к-ть наносекунд на момент 1 червня 2034-го року.

Отже, діапазоном значень для цієї функції буде $[0; 20342592000000000000]$.

Для того, щоб початкове значення X_o знаходилось у діапазоні $[0; m]$ використовується аналогічний алгоритм нормалізації.

```python
import os
import time

5 usages
class LCG:
    def __init__(self, a=48271, c=0, m=2**31-1, seed=None):
        self.a = a
        self.c = c
        self.m = m

        if seed is None:
            self.x0 = (os.getpid() + time.time_ns() / (20342592000000000000 + 4294967295) * self.m)
        else:
            self.x0 = seed

        self.x_prev = (self.a * self.x0 + self.c) % self.m

    7 usages
    def generate(self, num_range=None):
        self.x_prev = (self.a * self.x_prev + self.c) % self.m
        if num_range is None:
            return self.x_prev
        else:
            return int((self.x_prev / (self.m - 1) * (num_range[1] - num_range[0]) + num_range[0]))
```

Рисунок 3.2.2 Лінійний конгруентний метод реалізований у Python

Тепер постає питання у виборі початкових параметрів a, c та m для лінійного конгруентного методу. Серед великої кількості параметрів, що існує було обрано 4 на порівняння.

1. C++11's minstd_rand: $a = 48271, c = 0, m = 2^{31} - 1$
2. Java java.util.Random: $a = 25214903917, c = 11, m = 2\text{^}48$
3. RANDU: $a = 65539, c = 0, m = 2\text{^}31$
4. Musl: $a = 6364136223846793005, c = 1, m = 2\text{^}64$

Для вибору найбільш оптимальним параметрів для вибору атрибутів для NFT було проведене наступне дослідження. Перед тим як перейти до нього буде означено процес вибору атрибутів та їх значень.

Для кожного з атрибутів є визначена ймовірність вибору, що означена числом від 0 до 1000 та зберігається у масиві з ймовірностями інших значень елемента. За допомогою лінійного конгруентного методу генерується спочатку 4 числа, що визначають вибір значень для перших чотирьох обов’язкових елементів. Для кожного зі згенерованих чисел вибирається найближче число з кожного масиву, що є більшим за згенероване. Індекс цього значення використовується далі для вибору відповідного атрибуту з наявних. Далі йде процедура вибору опціональних атрибутів. Для кожного з них процес виглядає так:

1. Генерується значення 0-100
2. Якщо значення є більшим за певне число, що визначене як імовірність цього елемента у оригінальній колекції, тоді генерується число 0-1000, що буде використано для вибору значення опціонального елементу. Інакше записується значення 0.

Нижче представлена блок-схема, яка демонструє означений алгоритм.

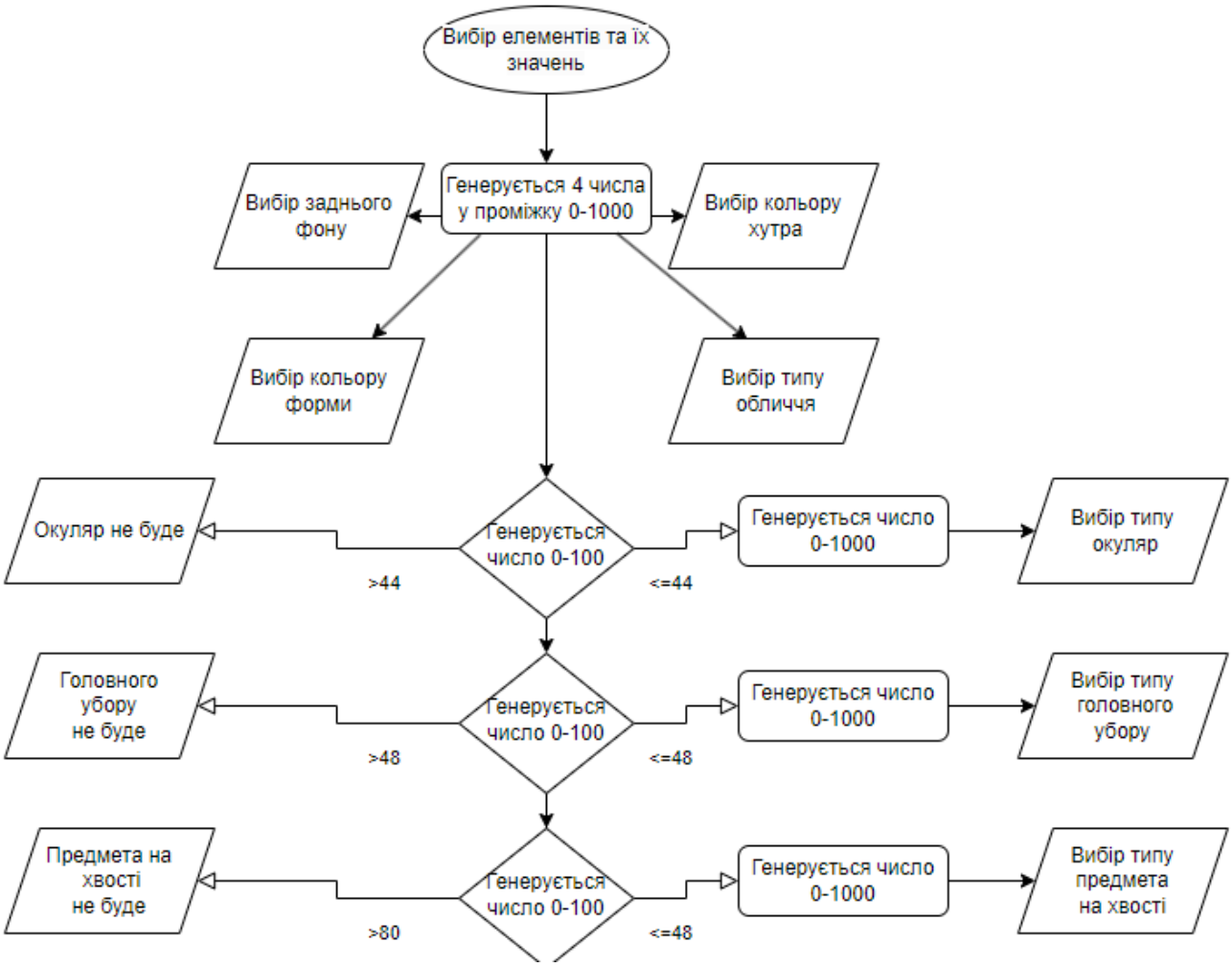

Рисунок 3.2.3 Блок-схема алгоритму вибору атрибутів для NFT.

Задані ймовірності для всіх атрибутів визначені наступним чином:

```
{
  "background" : [111, 222, 333, 444, 555, 666, 777, 888, 1000],
  "face": [333, 666, 1000],
  "facewear": [235, 470, 970 ,1000],
  "shape": [123, 357, 502, 580, 636, 825, 1000],
  "fur": [285, 531, 692, 791, 853, 916, 1000],
  "headwear": [149, 397, 536, 623, 735, 828, 903, 969, 1000],
  "tail": [182, 344, 480, 621, 677, 734, 838, 902, 953, 1000]
}
```

Рисунок 3.2.4 Визначені ймовірності для всіх атрибутів

Перейдемо до аналізу параметрів. Дослідження полягало у тому, щоб декілька разів зробити тестовий контент для колекції розміром 10 тисяч та проаналізувати при яких параметрах фактична ймовірність появи атрибутів буде найближчою до заданої. Для цього тричі було згенеровано метадані для кожного елемента колекції і зібрано інформацію про ймовірності появи значень кожного з атрибутів.

Background	Ash Gray	Camouflage Green	Dark Vanilla	Ebony	Light Silver	Olivine	Pastel Blue	Philippine Gray	Sky Blue
Визначені ймовірності	11,11	11,11	11,11	11,11	11,11	11,11	11,11	11,11	11,11
Ймовірності c++11 №1	10,1	11,6	13,5	11,5	9,9	11,8	10,8	9,8	11
Ймовірності java №1	10,5	11,5	10,9	11,7	12	10,2	11,9	9,8	11,6
Ймовірності randu №1	10,4	10,6	11,6	11,7	11,4	11,1	10,7	11	11,5
Ймовірності musl №1	10,2	9,8	11,4	10,9	11	10,4	11,2	13,6	11,5
Ймовірності c++11 №2	11,5	13,2	12,4	10,1	11,3	10,3	11,4	10,1	9,7
Ймовірності java №2	11,1	11,1	11	12	10,4	11	10,5	11,3	11,6
Ймовірності randu №2	14,2	10,4	10,7	11,2	10,1	10,8	9,8	11,9	10,9
Ймовірності musl №2	12,4	12	11	10,3	10,8	10,7	11,9	11,3	9,6
Ймовірності c++11 №3	12	11,3	12,9	11,5	11	9,7	10,3	10,4	10,9
Ймовірності java №3	10	10,1	12	11,8	9,2	10	12,7	11,6	12,6
Ймовірності randu №3	11,4	12,2	10,3	13,1	11,5	9,6	10,7	10,5	10,7
Ймовірності musl №3	10,7	10,5	11,5	10,4	12,1	11,5	11,3	11	11

Таблиця 3.2.4 Зібрані дані про ймовірність появи типів заднього фону

Shape	Red	Black	Pink	Orange	Blue	Purple	Yellow
Визначені ймовірності	12,3	23,4	14,5	7,8	5,6	18,9	17,5
Ймовірності c++11 №1	11,8	24,5	15	6,7	6,8	16,7	18,5
Ймовірності java №1	10,9	23,4	14,4	8,5	6,6	20,3	15,9
Ймовірності randu №1	12,4	24	17,3	7,3	5,7	18,5	14,8
Ймовірності musl №1	14,7	23,2	13,5	8,5	5	18,2	16,9
Ймовірності c++11 №2	13,6	25,3	14,3	5,6	4,6	19,6	17
Ймовірності java №2	10,8	25	13,9	7,5	6,8	21	15
Ймовірності randu №2	12,4	22,4	14,4	8,3	5,5	19	18
Ймовірності musl №2	11,7	24,7	13,1	7,9	4,5	19	19,1
Ймовірності c++11 №3	12,7	21,5	13,7	9	5,1	20,2	17,8
Ймовірності java №3	13,4	24,8	13,8	7,1	5,7	17,5	17,7
Ймовірності randu №3	13,6	24,1	15,1	7,9	5,3	20,1	13,9
Ймовірності musl №3	11,2	25,2	14,1	7,3	6,1	18,2	17,9

Таблиця 3.2.5 Зібрані дані про ймовірність появи різних кольорів форми

Fur	Gray	Red	Orange	Leopard	Blue&Yellow	Black&Red	Pink w w dots
Визначені ймовірності	28,5	24,6	16,1	9,9	6,2	6,3	8,4
Ймовірності c++11 №1	29,5	21,8	17,5	10,3	6,6	5,8	8,5
Ймовірності java №1	27,7	24,2	16,8	9,8	5,8	7,7	8
Ймовірності randu №1	31,3	23,6	15,8	9,9	5,4	7,3	6,7
Ймовірності musl №1	28,6	24,5	16,3	10,4	6,7	4,9	8,6
Ймовірності c++11 №2	25,1	25,8	19,4	10,1	6,1	6,3	7,2
Ймовірності java №2	28,1	23,7	17	9	7,1	6,4	8,7
Ймовірності randu №2	30	24,5	15	11,3	5,5	5,4	8,3
Ймовірності musl №2	29,9	25,2	14,1	9,9	8	6,7	8
Ймовірності c++11 №3	31,2	24,4	14,6	9	6,4	5,7	8,7
Ймовірності java №3	26,6	26,8	15,9	8,8	7,6	6,6	7,7
Ймовірності randu №3	26,3	24,4	17,8	9,7	6,4	7,3	8,1
Ймовірності musl №3	29	24,4	16,2	10,8	6,2	6,1	7,3

Таблиця 3.2.6 Зібрані дані про ймовірність появи різних кольорів хутра

Face	Sad	Happy	Face w ibrows
Визначені ймовірності	33,33	33,33	33,33
Ймовірності c++11 №1	32,9	34,5	32,6
Ймовірності java №1	31,8	36,6	31,6
Ймовірності randu №1	30,1	34,3	35,6
Ймовірності musl №1	33,9	33	33,1
Ймовірності c++11 №2	33,1	33,7	33,2
Ймовірності java №2	34,4	30	35,6
Ймовірності randu №2	33,9	33	33,1
Ймовірності musl №2	32,3	34	33,7
Ймовірності c++11 №3	36,8	35,6	27,6
Ймовірності java №3	37,1	32,3	30,6
Ймовірності randu №3	32,2	34,8	33
Ймовірності musl №3	36,2	31	32,8

Таблиця 3.2.7 Зібрані дані про ймовірність появи різних типів обличчя

Facewear	3D	FPV Drone	Black glasses	Terminator
Визначені ймовірності	23,5	23,5	50,0	3,0
Ймовірності c++11 №1	18,2	27,4	51,8	2,6
Ймовірності java №1	24,0	24,4	49,3	2,3
Ймовірності randu №1	28,6	20,6	48,1	2,7
Ймовірності musl №1	28,1	22,6	46,5	2,9
Ймовірності c++11 №2	22,7	23,4	51,9	2,1
Ймовірності java №2	26,9	24,1	47,9	1,1
Ймовірності randu №2	23,8	22,7	22,4	2,3
Ймовірності musl №2	24,4	24,0	50,0	1,6
Ймовірності c++11 №3	24,7	23,3	48,2	3,8
Ймовірності java №3	26,6	23,3	46,7	3,5
Ймовірності randu №3	24,6	21,5	51,0	2,9
Ймовірності musl №3	24,0	24,0	49,0	3,1

Таблиця 3.2.7 Зібрані дані про ймовірність появи різних типів окуляр

Headwear	SR beret	ET beret	RTF beret	AFU G helmet	Police cap	AFU B helmet	FireF helmet	Medic helmet	Kalush
Визначені ймовірності	14,9	24,8	13,9	8,7	11,2	9,3	7,5	6,6	3,1
Ймовірності c++11 №1	14,5	24,2	14,5	9,3	11,1	9,7	8,3	5,1	3,4
Ймовірності java №1	15,5	24,7	14	7,5	11,5	9,2	8,6	6,3	2,7
Ймовірності randu №1	14,8	24	13,3	7,9	11,2	9,4	9	6,9	3,4
Ймовірності musl №1	12,1	27,1	15	9	11,5	11,7	5,7	4,7	3,1
Ймовірності c++11 №2	16	27,3	11,7	7,6	10,2	8,2	7,6	8,6	2,9
Ймовірності java №2	12,9	28,3	14,2	7,7	9,6	8,1	9,2	7,1	2,9
Ймовірності randu №2	14	26,2	15,2	8,7	10,6	8,8	7,7	6,2	2,7
Ймовірності musl №2	13,3	23,9	14,3	8,4	10,2	9,8	8,8	7,8	3,3
Ймовірності c++11 №3	17,9	25,3	14,5	9,4	9,8	7,9	8,3	4,3	2,8
Ймовірності java №3	15,9	22,2	16,4	10,3	12,4	10,1	5,7	5,7	1,5
Ймовірності randu №3	15,9	23,7	16,9	8,4	11,4	7	7,6	6,2	3
Ймовірності musl №3	14,5	24,8	15,5	10,6	8,5	9,5	5,6	7,7	3,3

Таблиця 3.2.7 Зібрані дані про ймовірність появи різних типів головних уборів

Tail	Handgun	M416 Rifle	Molotov	Knife	UA Flag	Watermelon	Grenade	Cossack saber	Javelin	Jar of cucumbers
Визначені ймовірності	18,2	16,2	13,6	14,1	5,6	5,7	10,4	6,4	6,6	3,2
Ймовірності c++11 №1	18,6	17,2	12,3	12,7	5,5	6,2	10,9	7,4	5	4,2
Ймовірності java №1	18,1	13,9	12,5	14,4	5	8	10,2	7,5	5,7	4,7
Ймовірності randu №1	16,9	16,8	14,3	14,7	5,3	5,5	11,2	5	4,7	5,6
Ймовірності musl №1	18,2	18,9	12,7	14,9	4,4	6	10,3	6,8	3,7	4,2
Ймовірності c++11 №2	16,7	15,5	16,1	15,2	5,1	5,4	10	6,3	5,3	4,4
Ймовірності java №2	19	16,3	11,6	14,3	5,1	5,8	9,5	8,4	5,8	4,3
Ймовірності randu №2	19,5	16,5	11,2	15,2	6,2	4,5	10,7	6,5	4,9	4,9
Ймовірності musl №2	18,5	16,9	13,4	14,6	6,1	4,9	9,9	7,4	3,9	4,5
Ймовірності c++11 №3	17	15,8	12,6	16,1	4,9	6,1	9,8	6,9	5,8	5,1
Ймовірності java №3	17,2	15,7	13,8	13	6,8	5,9	10,1	6,8	5	5,7
Ймовірності randu №3	19,9	15,1	14,8	12,5	5,5	5,2	12,3	5,6	5,4	3,7
Ймовірності musl №3	14,8	16,2	13,7	14,7	5,7	4,9	12	7,2	5,5	5,2

Таблиця 3.2.7 Зібрані дані про ймовірність появи різних типів предметів на хвості

Далі для кожного значення атрибуту обраховується середнє з трьох та для кожного набору параметрів обраховується кількість найближчих значень до заданих. Який набір параметрів має найбільше найближчих значень, вважається найефективнішим.

Background	Ash Gray	Camouflage Green	Dark Vanilla	Ebony	Light Silver	Olivine	Pastel Blue	Philippine Gray	Sky Blue	
Визначені ймовірності	11,11	11,11	11,11	11,11	11,11	11,11	11,11	11,11	11,11	
Ймовірності c++11 AVG	11,2	12,03	12,93	11,03	10,73	10,60	10,83	10,10	10,53	4th
Ймовірності java AVG	10,53	10,90	11,30	11,83	10,53	10,40	11,70	10,90	11,93	3rd
Ймовірності randu AVG	12,00	11,07	10,87	12,00	11,00	10,50	10,40	11,13	11,03	W
Ймовірності musl №1	11,10	10,77	11,30	10,53	11,30	10,87	11,47	11,97	10,70	2nd
Найближче число	11,1	11,07	11,3	11,03	11	10,87	10,83	11,13	11,03	
2 найближче число	11,2	10,9	11,3	10,53	11,3	10,6	11,47	10,9	10,7	
3 найближче число	10,53	10,77	10,87	11,83	10,73	10,5	11,7	11,97	10,53	

Таблиця 3.2.8 Аналіз найефективнішого методу для кольору заднього фону

Shape	Red	Black	Pink	Orange	Blue	Purple	Yellow	
Визначені ймовірності	12,3	23,4	14,5	7,8	5,6	18,9	17,5	
Ймовірності c++11 AVG	12,70	23,77	14,33	7,10	5,50	18,83	17,77	W
Ймовірності java AVG	11,70	24,40	14,03	7,70	6,37	19,60	16,20	4th
Ймовірності randu AVG	12,80	23,50	15,60	7,83	5,50	19,20	15,57	2nd
Ймовірності musl №1	12,53	24,37	13,57	7,90	5,20	18,47	17,97	3rd
Найближче число	12,53	23,5	14,33	7,83	5,5	18,83	17,77	
2 найближче число	12,7	23,77	14,03	7,7	5,5	19,2	17,97	
3 найближче число	12,8	24,37	13,57	7,9	5,2	18,47	16,2	

Таблиця 3.2.9 Аналіз найефективнішого методу для кольору форми

Fur	Gray	Red	Orange	Leopard	Blue&Yellow	Black&Red	Pink w w dots	
Визначені ймовірності	28,5	24,6	16,1	9,9	6,2	6,3	8,4	
Ймовірності c++11 AVG	28,60	24,00	17,17	9,80	6,37	5,93	8,13	W
Ймовірності java AVG	27,47	24,90	16,57	9,20	6,83	6,90	8,13	4th
Ймовірності randu AVG	29,20	24,17	16,20	10,30	5,77	6,67	7,70	2nd
Ймовірності musl №1	29,17	24,70	15,53	10,37	6,97	5,90	7,97	3rd
Найближче число	28,6	24,7	16,2	9,8	6,37	5,93	8,13	
2 найближче число	29,17	24,9	16,57	10,3	5,77	6,67	8,13	
3 найближче число	29,2	24,17	15,53	10,37	6,83	5,9	7,97	

Таблиця 3.2.10 Аналіз найефективнішого методу для кольору хутра

Face	Sad	Happy	Face w ibrows	
Визначені ймовірності	33,33	33,33	33,33	
Ймовірності c++11 AVG	34,27	34,60	31,13	4th
Ймовірності java AVG	34,43	32,97	32,60	2nd
Ймовірності randu AVG	32,07	34,03	33,90	3rd
Ймовірності musl №1	34,13	32,67	33,20	**W**
Найближче число	34,13	32,97	33,2	
2 найближче число	34,27	32,67	33,9	
3 найближче число	34,43	34,03	32,6	

Таблиця 3.2.11 Аналіз найефективнішого методу для типу обличчя

Facewear	3D	FPV Drone	Black glasses	Terminator	
Визначені ймовірності	23,5	23,5	50,0	3,0	
Ймовірності c++11 AVG	21,87	24,70	50,63	2,83	**W**
Ймовірності java AVG	25,83	23,93	47,97	2,30	3rd
Ймовірності randu AVG	25,67	21,60	40,50	2,63	4th
Ймовірності musl №1	25,50	23,53	48,50	2,53	2nd
Найближче число	21,87	23,53	50,63	2,83	
2 найближче число	25,5	23,93	48,5	2,63	
3 найближче число	25,67	24,7	47,97	2,53	

Таблиця 3.2.12 Аналіз найефективнішого методу для типу окуляр

Headwear	SR beret	ET beret	RTF beret	AFU G helmet	Police cap	AFU B helmet	FireF helmet	Medic helmet	Kalush	
Визначені ймовірності	14,9	24,8	13,9	8,7	11,2	9,3	7,5	6,6	3,1	
Ймовірності c++11 AVG	16,13	25,60	13,57	8,77	10,37	8,60	8,07	6,00	3,03	3rd
Ймовірності java AVG	14,77	25,07	14,87	8,50	11,17	9,13	7,83	6,37	2,37	**W**
Ймовірності randu AVG	14,90	24,63	15,13	8,33	11,07	8,40	8,10	6,43	3,03	2nd
Ймовірності musl №1	13,30	25,27	14,93	9,33	10,07	10,33	6,70	6,73	3,23	4th
Найближче число	14,9	24,63	13,57	8,77	11,17	9,13	7,83	6,73	3,03	
2 найближче число	14,77	25,07	14,87	8,5	11,07	8,6	8,07	6,43	3,03	
3 найближче число	16,13	25,27	14,93	8,33	10,37	8,4	8,1	6,37	3,23	

Таблиця 3.2.12 Аналіз найефективнішого методу для типу головного убору

Tail	Handgun	M416 Rifle	Molotov	Knife	UA Flag	Watermelon	Grenade	Cossack saber	Javelin	ar of cucumbers	
Визначені ймовірності	18,2	16,2	13,6	14,1	5,6	5,7	10,4	6,4	6,6	3,2	
Ймовірності c++11 AVG	17,43	16,17	13,67	14,67	5,17	5,90	10,23	6,87	5,37	4,57	W
Ймовірності java AVG	18,10	15,30	12,63	13,90	5,63	6,57	9,93	7,57	5,50	4,90	2nd
Ймовірності randu AVG	18,77	16,13	13,43	14,13	5,67	5,07	11,40	5,70	5,00	4,73	3rd
Ймовірності musl №1	17,17	17,33	13,27	14,73	5,40	5,27	10,73	7,13	4,37	4,63	4th
Найближче число	18,1	16,17	13,67	14,13	5,63	5,9	10,23	6,87	5,5	4,57	
2 найближче число	18,77	16,13	13,43	13,9	5,67	5,27	10,73	5,7	5,37	4,63	
3 найближче число	17,43	15,3	13,27	14,67	5,4	5,07	9,93	7,13	5	4,73	

Таблиця 3.2.12 Аналіз найефективнішого методу для типу предмету на хвості

Для того, щоб визначити загальний найефективніший метод створимо таблицю з загальними результатами та проаналізуємо завдяки різним критеріям голосування. Нехай c+11, java, randu, musl це a,b,c,d відповідно, тоді:

К-ть виборів	1	2	1	1	1	1
К-ть балів						
3	c	a	d	a	b	a
2	d	c	b	d	c	b
1	b	d	c	b	a	c
0	a	b	a	c	d	d

Таблиця 3.2.13 Голосування 7 “виборців” за чотири альтернативи

Застосуємо наступні критерії:

1. Відносна більшість. Рахуються лише альтернативи, що мають 3 бали. Перемагає та, що має найбільшу кількість голосів.
2. Критерій Кондорсе. Для кожної пари альтернатив вираховується к-ть разів, коли елемент пари отримує більшу к-ть очок ніж інший елемент. Якщо існує кандидат, що в кожній з пар отримав к-ть голосів, що є більшої за половину виборців, він перемагає.

3. Критерій Де Борда. Для кожного з кандидатів рахується загальна к-ть очок, що вони отримали від виборців. Кандидат з найбільшою к-тю очок перемагає.
4. Критерій Копленда. Для кожної пари альтернатив вираховується к-ть разів, коли елемент пари отримує більшу к-ть очок ніж інший елемент і ця кількість має бути більше половини числа виборців, в такому випадку кандидат отримує +1 бал. Якщо менше половини, то -1 бал. Інакше – 0 балів. Перемагає кандидат з найбільшою к-тю балів.
5. Критерій Сімпсона. Для таблиці отриманої з критерію Кондорсе кожній альтернативі присвоюється число найменшою к-ть голосів в рядку. Альтернатива з найбільшим числом перемагає.

Реалізувавши ці критерії за допомогою Python, використовуючи наступні вхідні дані:

```
list_of_candidates_start=["a", "b", "c", "d"]
main_arr=[
    ["", 1, 2, 1, 1, 1, 1],
    [3, "c", "a", "d", "a", "b", "a"],
    [2, "d", "c", "b", "d", "c", "b"],
    [1, "b", "d", "c", "b", "a", "c"],
    [0, "a", "b", "a", "c", "d", "d"]]

all_cand = 7
```

Рисунок 3.2.14 Вхідні дані для оцінки 5-ьома критеріями

Отримано наступні результати:

```
Переможець за критерієм відносної більшості: ({'a': 4, 'b': 1, 'c': 1, 'd': 1}, 'a')
Переможець критерієм Кондорсе:  ({'a': [0, 4, 4, 5], 'b': [3, 0, 4, 2], 'c': [3, 3, 0, 5], 'd': [2, 5, 2, 0]}, 'a')
Переможець критерієм Де Борда:  ({'a': 13, 'b': 9, 'c': 11, 'd': 9}, 'a')
Переможець критерієм Копленда:  ({'a': 3, 'b': -1, 'c': -1, 'd': -1}, 'a')
Переможець критерієм Сіпмсона:  ({'a': 4, 'b': 2, 'c': 3, 'd': 2}, 'a')
```

Рисунок 3.2.15 Результати оцінок

Бачимо, що всі п’ять критеріїв обрали переможцем кандидата a, тобто параметри C++11's minstd_rand:

$$a = 48271, c = 0, m = 2^{31} - 1$$

Далі згенеруємо 1000 NFT та їх метадані, використовуючи обрані вище параметри. Для того, щоб не було дублікатів, перед тим як зберігати NFT існує перевірка чи є вже ідентичні метадані у базі згенерованих, якщо є, то процес генерації проходить ще раз.

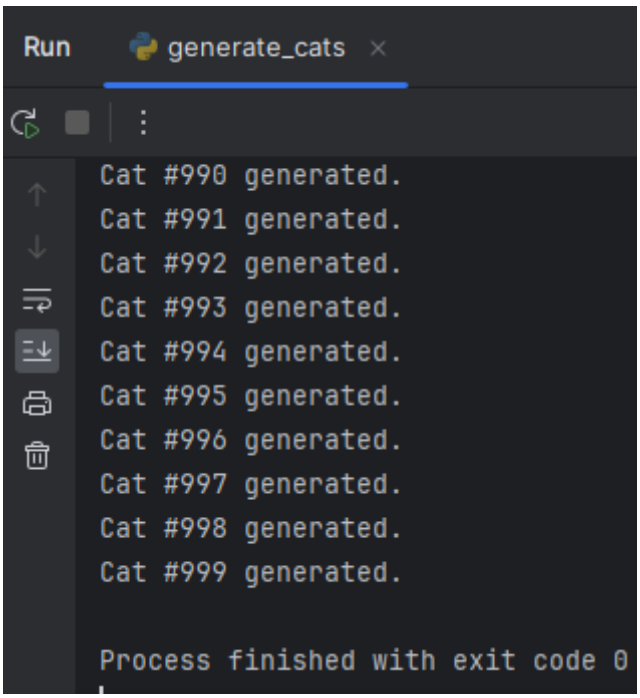

Рисунок 3.2.16 Підтвердження згенерованих 1000 NFT

```
{
  "name": "Pixel UA Cat #783",
  "description": "Cat from collection PixelUACatsDivision",
  "image": "",
  "content_url": "",
  "attributes": [
    {
      "trait_type": "background",
      "value": "Ebony"
    },
    {
      "trait_type": "shape",
      "value": "Black"
    },
    {
      "trait_type": "fur",
      "value": "Red"
    },
    {
      "trait_type": "face",
      "value": "Happy face"
    },
    {
      "trait_type": "facewear",
      "value": "FPV Drone glasses"
    },
    {
      "trait_type": "headwear",
      "value": "Firefighter helmet"
    }
  ]
}
```


Рисунок 3.2.17 Приклад створеного зображення та метаданих

3.3 Розгортання колекції на блокчейн-платформі TON

Для того, щоб була можливість створити NFT з цих зображень та підкріпити метадані, потрібно зробити їх доступними по посиланню. Одним з популярних варіантів є завантаження, використовуючи ipfs - це протокол, гіпермедіа та файлообмінна пірингова мережа для зберігання та обміну даними в розподіленій файловій системі. Використовуючи контент-адресацію, IPFS унікально ідентифікує кожен файл у глобальному просторі імен, який з'єднує хости IPFS, створюючи стійку систему зберігання та обміну файлами. Одним з доступних варіантів для завантаження є сервіс Pinata.

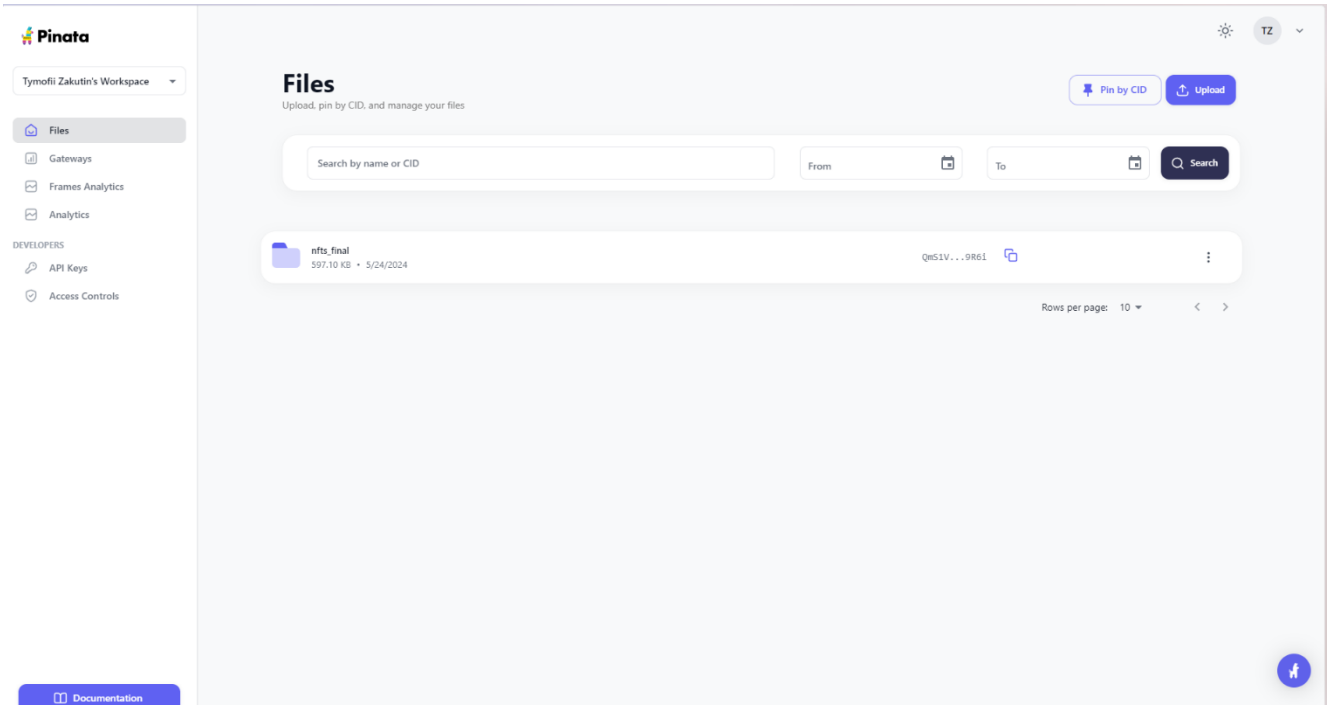

Рисунок 3.2.18 Завантаження контенту NFT на сервіс Pinata

Після завантажування зображень, потрібно оновити метадані, щоб вони містили актуальні посилання на потрібні зображення.

```python
import json

for i in range(0, 1000):
    with open(f"meta_final/{i}.json", "r") as mt:
        meta = json.load(mt)
    meta['image'] = f'https://bafybeiddevnzmeoq45msiesjspjfmfrljixovvol7bsojzc2wphtkvbox4.ipfs.dweb.link/{i}.png'
    with open(f"meta_final/{i}.json", 'w') as file:
        json.dump(meta, file, indent=4)
```

Рисунок 3.2.19 Код для оновлення метаданих

Після того, як метадані оновлені, завантажуємо їх на Pinata.

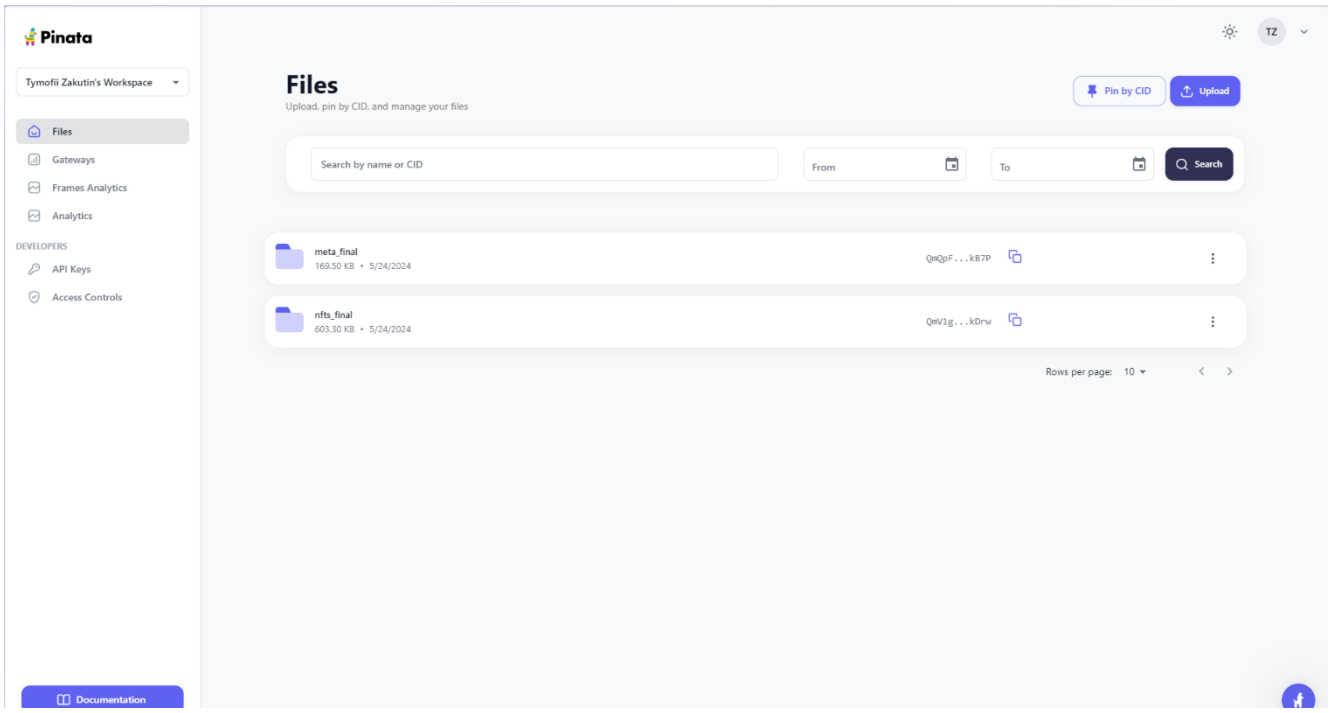

Рисунок 3.2.20 Завантаження метаданих NFT на сервіс Pinata

Далі перейдемо до створення самої колекції. Для цього потрібно створити гаманець у блокчейні TON, активувати його, та запустити смартконтракт створення NFT колекції. Для всіх цих дій будемо використовувати python бібліотеку tonsdk.

Створюємо гаманець:

```python
from tonsdk.contract.wallet import Wallets, WalletVersionEnum

mnemonics, pub_k, priv_k, wallet = Wallets.create(version=WalletVersionEnum.v4r2, workchain=0)

wallet_address = wallet.address.to_string(True, True, True)

if __name__ == '__main__':
    for mn in mnemonics:
        print(mn, end=" ")
    print("\n", wallet.address.to_string(True, True, True))
```

Рисунок 3.2.21 Створення гаманця у блокчейні TON

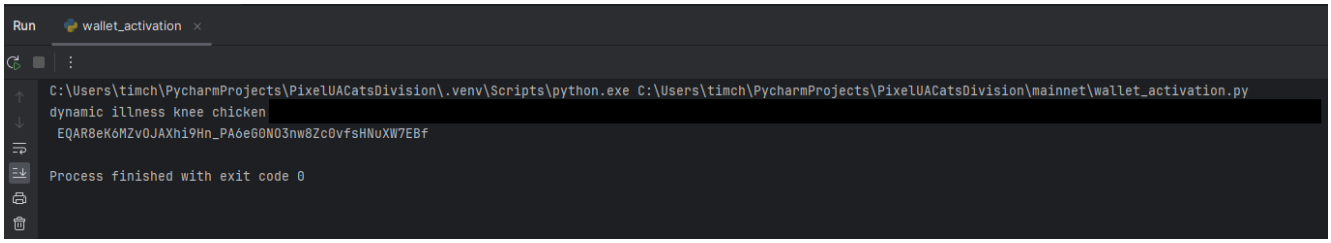

Рисунок 3.2.22 Гаманець та 24 секретних слова

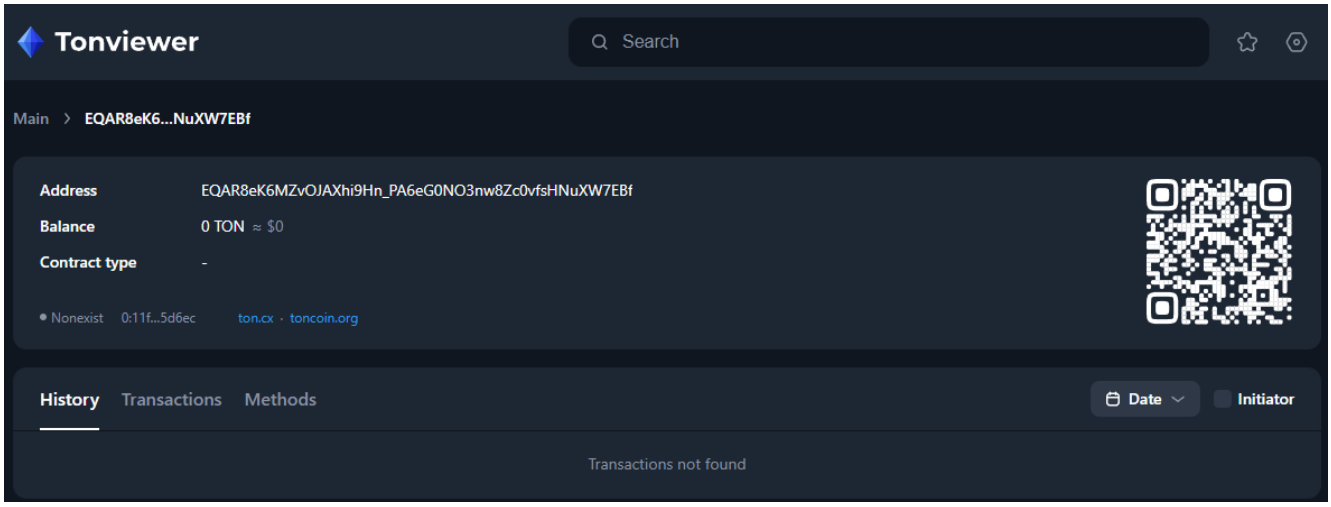

Рисунок 3.2.23 Інформація про гаманець на Tonviewer - це функціональний оглядач блокчейну, спеціально розроблений для TON.

При створенні гаманець має статус «Nonexist», тобто не існуючий/не активований. Для активації потрібно поповнити гаманець і після того зробити перший переказ, що автоматично викликає смартконтракт активації гаманця.

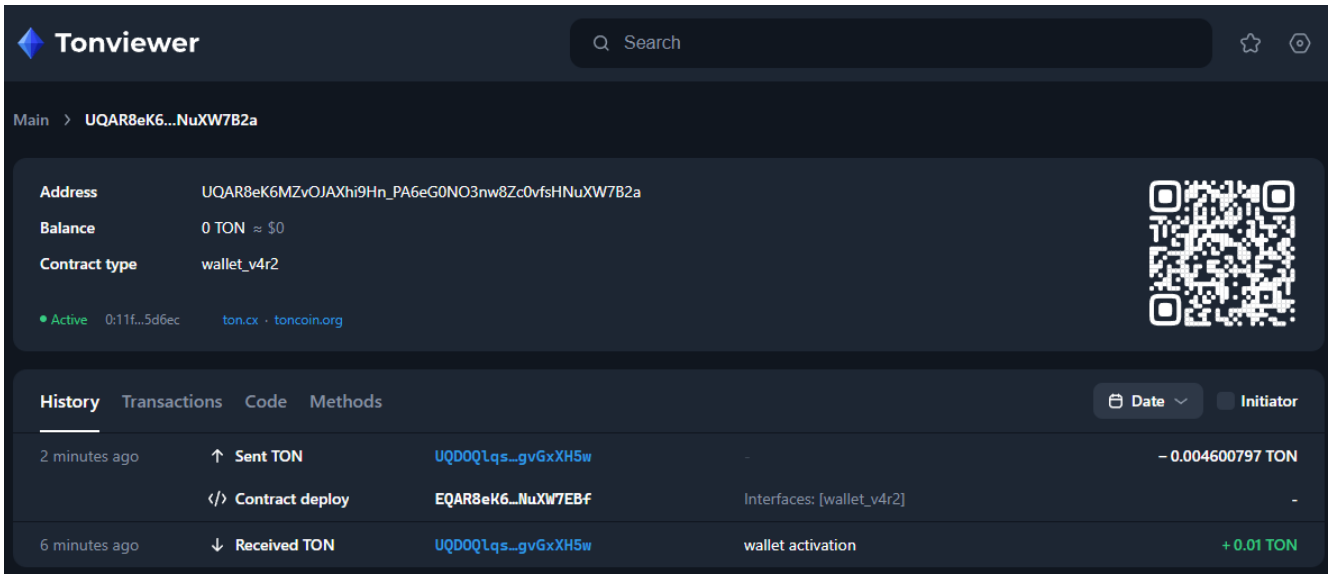

Рисунок 3.2.24 Інформація на Tonviewer про активований гаманець

Для того, щоб викликати смартконтракт NFT колекції використовується наступний код:

```python
async def deploy_collection():

    collection = create_collection_mint()
    state_init = collection.create_state_init()['state_init']

    client = await get_client()

    seqno = await get_seqno(client, wallet_address)

    query = wallet.create_transfer_message(to_addr=collection.address.to_string(),
                                           amount=to_nano( number: 0.05,  unit: 'ton'), seqno=seqno, state_init=state_init)

    await client.raw_send_message(query['message'].to_boc(False))
```

```python
from tonsdk.contract.token.nft import NFTCollection, NFTItem
from tonsdk.contract import Address
from tonsdk.utils import to_nano

3 usages
def create_collection_mint():
    royalty_base = 1000
    royalty_factor = 0.055



    collection = NFTCollection(royalty_base=royalty_base,
                               royalty=royalty_factor,
                               royalty_address=Address('UQAR8eK6MZvOJAXhi9Hn_PA6eG0NO3nw8Zc0vfsHNuXW7B2a'),
                               owner_address=Address('UQAR8eK6MZvOJAXhi9Hn_PA6eG0NO3nw8Zc0vfsHNuXW7B2a'),
                               collection_content_uri='https://bafybeibezqk4q6xqoqnhqwyixt42fdmezfsftfhl4dkp3uzlx2djamvifq.ipfs.dweb.link/meta.json',
                               nft_item_content_base_uri='https://bafybeibezqk4q6xqoqnhqwyixt42fdmezfsftfhl4dkp3uzlx2djamvifq.ipfs.dweb.link/',
                               nft_item_code_hex=NFTItem.code)

    return collection
```

Рисунок 3.2.25 Деплой NFT-колекції

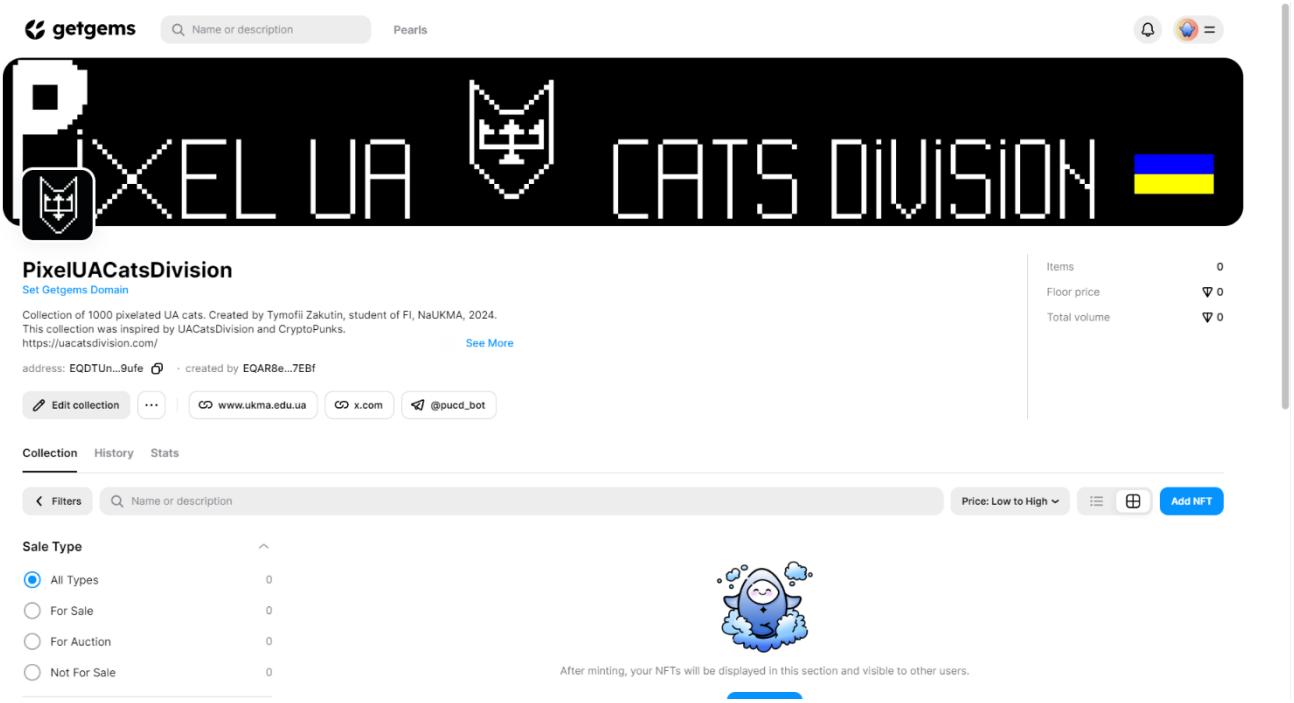

Рисунок 3.2.26 Колекція на NFT-маркетплейсі Getgems.io

Наступним етапом стала розробка телеграм-бота, в якому користувач матиме змогу купити свою NFT.
Бота було створено, використовуючи python-бібліотеку telebot.
Блок-схема роботи телеграм-бота:

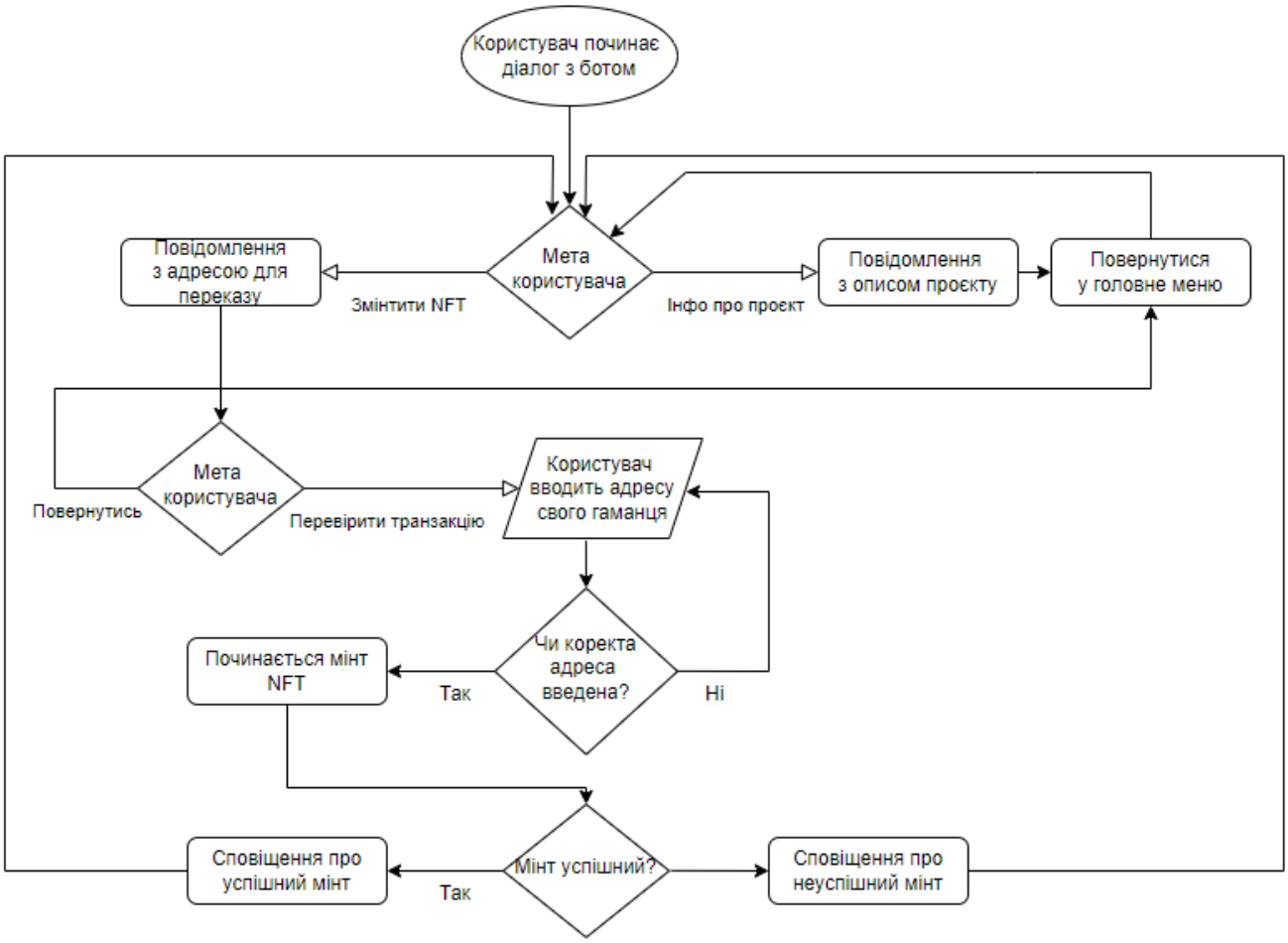

Рисунок 3.2.27 Блок-схема роботи бота

Далі перейдемо до прикладу покупки NFT.

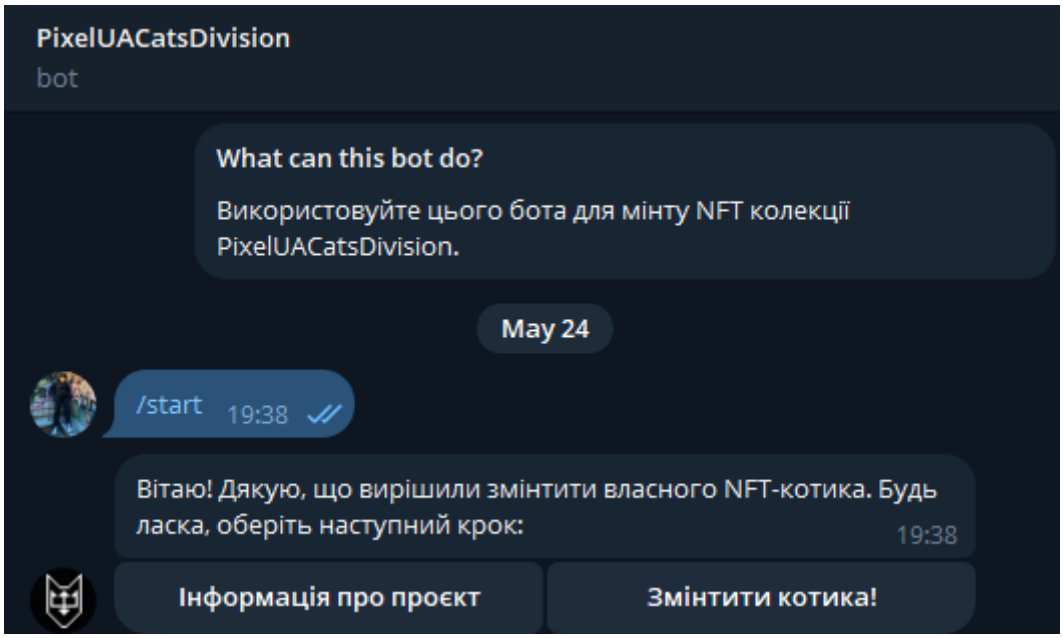

Рисунок 3.2.28 Вітальна сторінка

При натисканні інформація про проєкт, отримуємо:

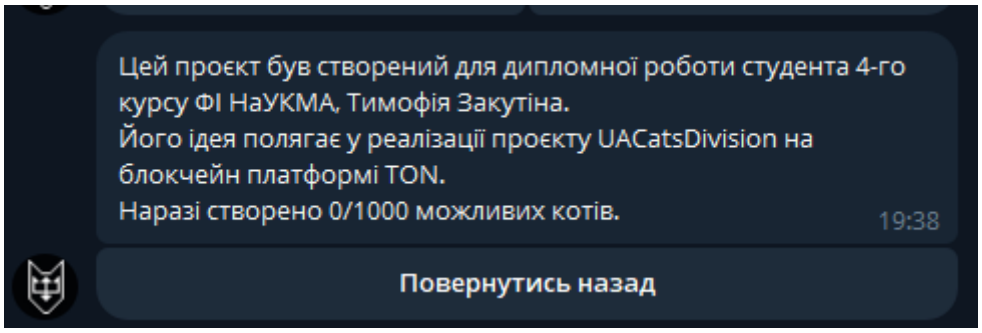

Рисунок 3.2.29 Інформація про проєкт

Для того, щоб здійснити мінт, натискаємо на «Змінтити котика! », здійснюємо переказ та надсилаємо свою адресу, підтверджуємо її правильність.

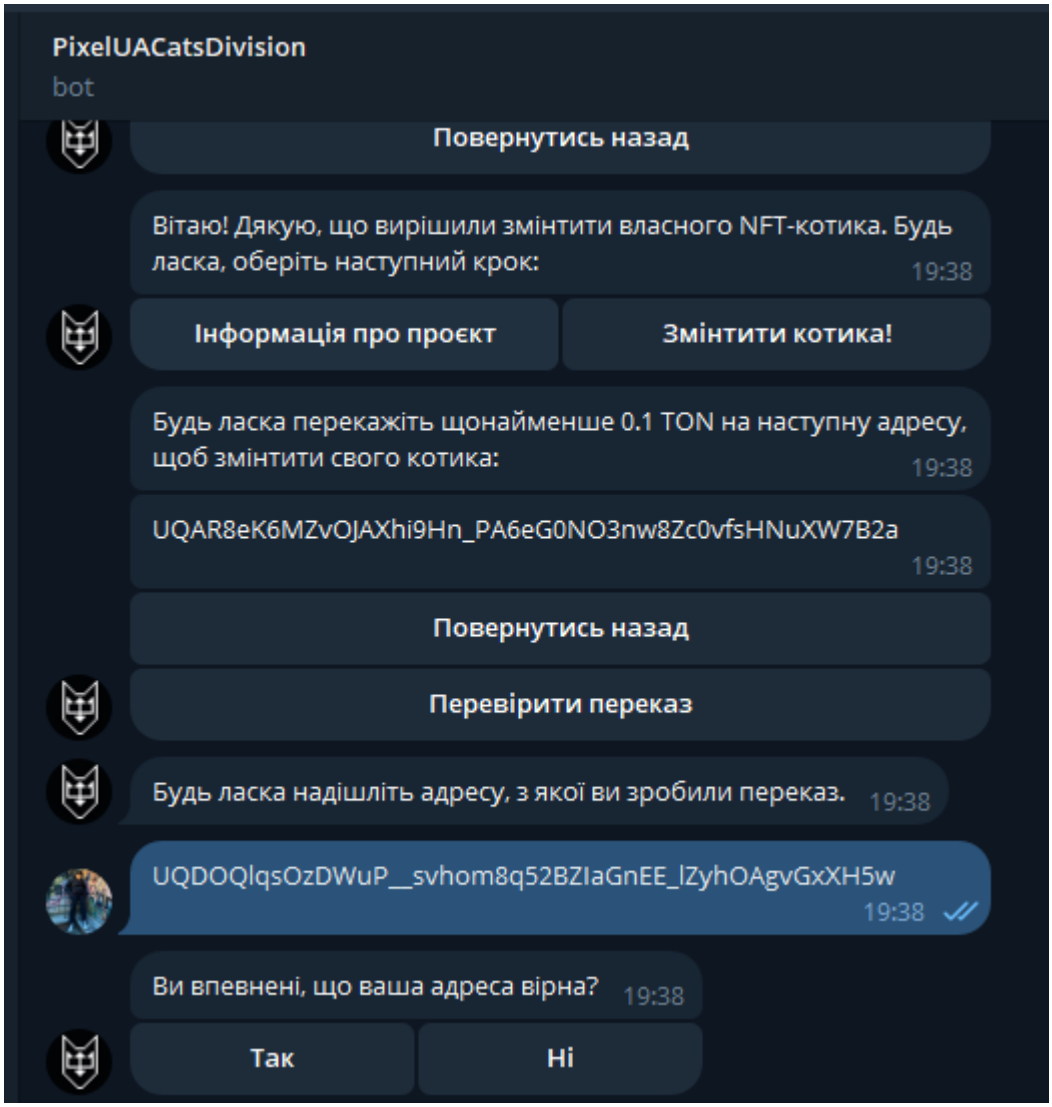

Рисунок 3.2.30 Приклад мінту NFT

Якщо, транзакція було успішно підтверджена, отримаємо відповідне сповіщення.

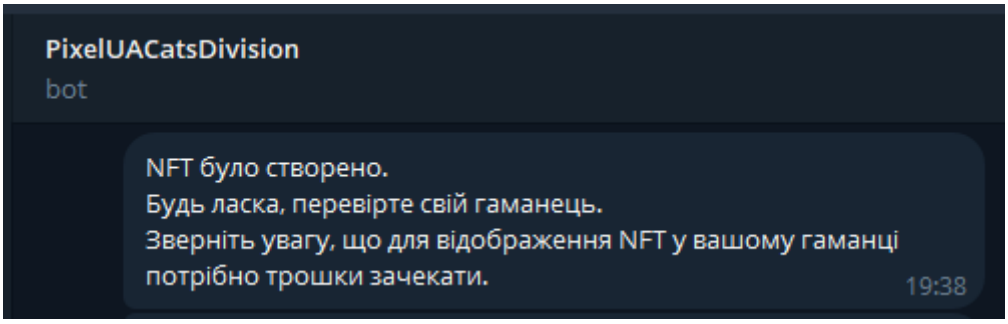

Рисунок 3.2.30 Приклад сповіщення про успішний мінт.

Можемо перевірити, що мінт відбувся успішно та користувач отримав NFT на свій гаманець.

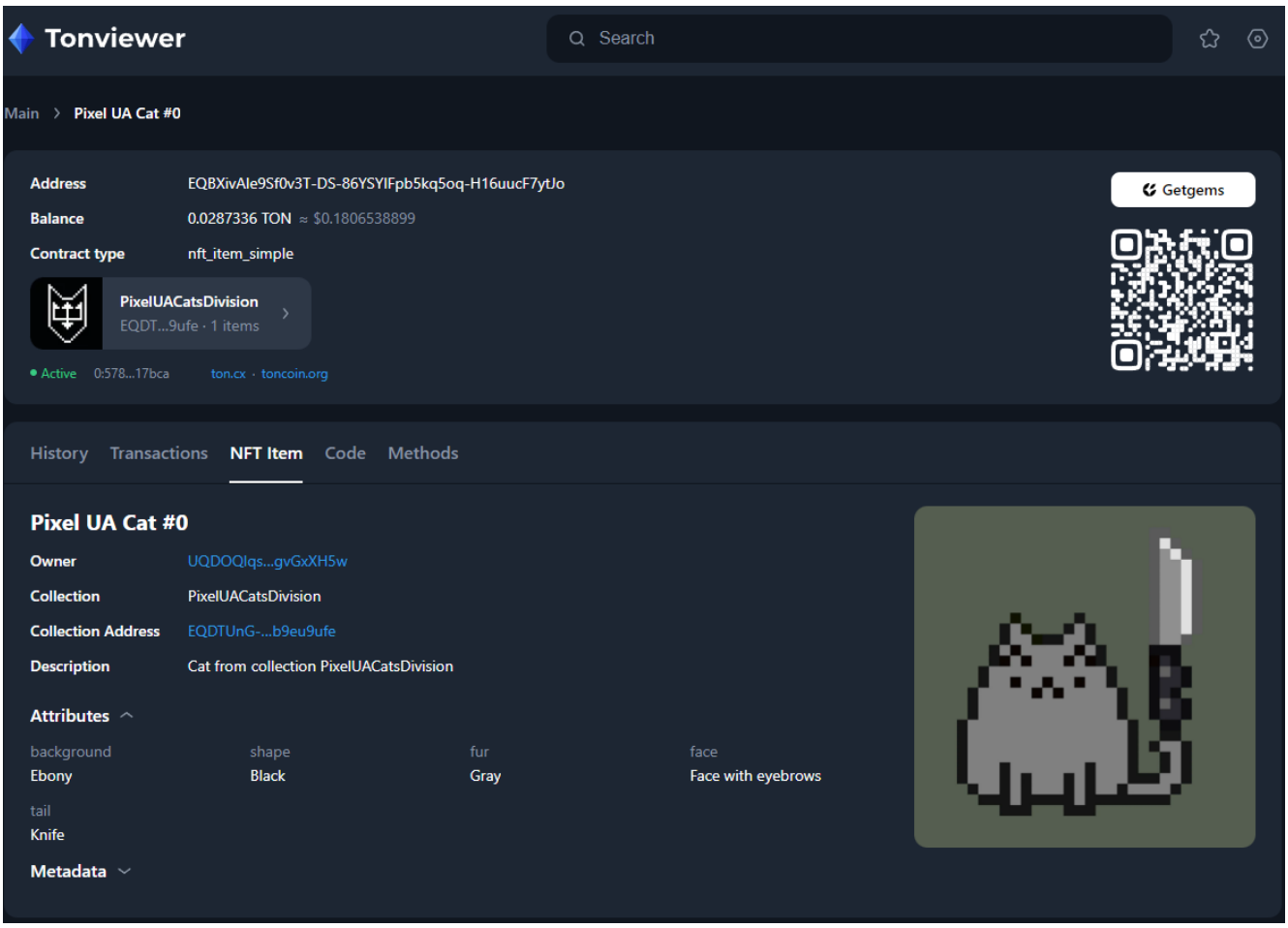

Рисунок 3.2.31 Інформація про NFT на платформі Tonviewer

Для того, щоб користувач не міг змінтити декілька NFT перевіряючи одну і ту саму транзакцію, після кожного мінту зберігається хеш.

Хеш генерується наступним чином: адреса користувача додається до часу здійснення транзакції і отриманий рядок хешується за допомогою алгоритму SHA-256.

Принцип алгоритму:

1. Вхідний рядок перетворюється у двійкову форму. В кінець додається 1, поки дані не будуть кратні 448. Додати 64 біти в кінець, де ці біти це число «від старшого до молодшого», що відповідають довжині оригінального рядка у двійковій формі.
2. Ініціалізація 8 хеш значень $h_0 - h_7$. Це визначені константи, які представляють перші 32 біти дробових частин квадратних коренів з перших 8 простих чисел: 2, 3, 5, 7, 11, 13, 17, 19.
3. Ініціалізація 64 констант. Кожне значення (0-63) - це перші 32 біти дробових частин кубічних коренів з перших 64 простих чисел (2 - 311).
4. Наступні кроки відбуватимуться для кожного 512-бітового шматка даних з вхідного рядка. На кожній ітерації циклу буде відбуватись зміна хеш-значення h0-h7, які і будуть кінцевим результатом.
5. Копіюються вхідні дані з кроку 1 у новий масив, кожен елемент якого є 32-бітним словом. Додається ще 48 слів, ініціалізованих нулем, так, щоб отримати масив w[0...63]. Змінюються нульові індекси в кінці масиву за наступним алгоритмом:

   Для і з w[16...63]:

   $$s_0 = ROTR(w[i-15], 7) \oplus ROTR(w[i-15], 18) \oplus (w[i-15] \gg 3)$$

   $$s_1 = ROTR(w[i-2], 17) \oplus ROTR(w[i-2], 19) \oplus (w[i-2] \gg 10)$$

   $$w[i] = w[i-16] + s_0 + w[i-7] + s_1$$

   .

   В результаті отримуємо 64 слова.

6. Ініціалізуються змінні a, b, c, d, e, f, g, h та встановлюються рівними поточним значенням хешу відповідно. h0, h1, h2, h3, h4, h5, h6, h7
   Запускається цикл стиснення. Цикл стиснення буде змінювати значення a-h. Цикл стиснення виглядає наступним чином:
   для i від 0 до 63
   $S_1 = ROTR(e,6) \oplus ROTR(e,11) \oplus ROTR(e,25)$
   $Ch = (e \wedge f) \oplus (\neg e \wedge g)$
   $temp1 = h + \Sigma_1 + Ch + k[i] + w[i]$
   $S_0 = ROTR(a,2) \oplus ROTR(a,13) \oplus ROTR(a,22)$
   $maj = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$
   $temp2 = \Sigma_0 + maj$
   $h = g$
   $g = f$
   $f = e$
   $e = d + temp1$
   $d = c$
   $c = bi$
   $b = a$
   $a = temp1 + temp2$
7. Після циклу стиснення, але все ще в циклі для 512-бітових шматків даних, змінюються хеш-значення, до них додаються відповідні дані a-h.
8. Всі дані $h_0 - h_7$ поєднуються між собою, і в результаті отримуємо фінальне хеш-значення.

При спробі повторно змінтити NFT по минулій транзакції, користувач отримує наступне повідомлення:

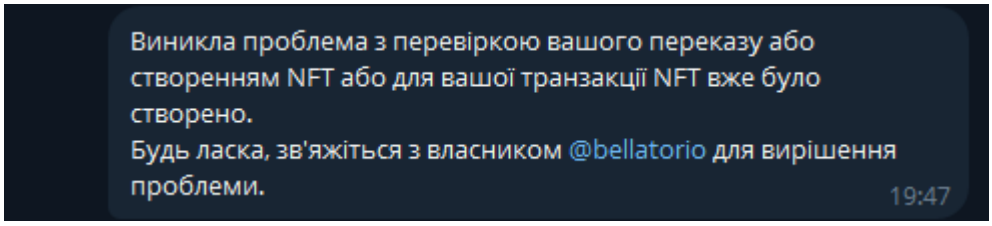

Рисунок 3.2.32 Сповіщення при спробі повторно змінтити NFT по першій транзакції

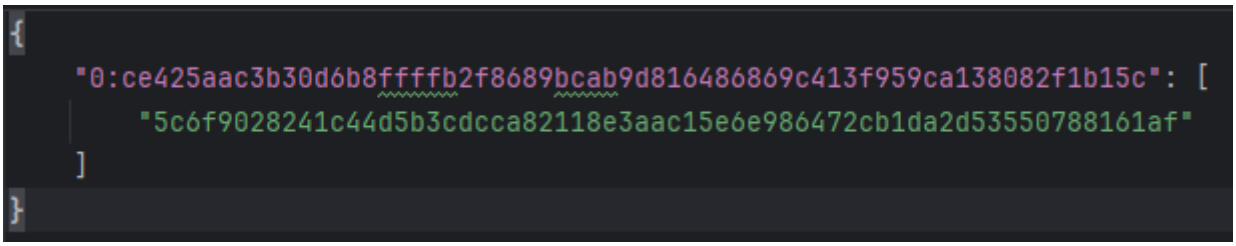

Рисунок 3.2.33 Приклад збереження хешу для кожного гаманця

Висновки

У ході виконання дипломної роботи було розглянуто поняття блокчейну, консенсусу, а також були окреслені переваги та недоліки для різних блокчейн-платформ та типів консенсусу. Також були окреслено термін NFT, та проведено дослідження щодо існуючих українських NFT-колекцій.

Був проведений статистичний аналіз NFT-колекції UACatsDivision з метою визначити алгоритм генерування зображень та ймовірності вибору різних елементів для створення контенту.

Для створення власної колекції були створені унікальні піксельні зображення, що в подальшому використовувались для створення NFT-контенту. Для алгоритму генерування зображень було використано лінійний конгруентний метод, що був імплементований у мові програмування Python. Було проведено дослідження з метою вибору найбільш оптимальних початкових параметрів для вищезгаданого методу. Для цього були використані 5 критеріїв голосування, що також були імплементовані у Python.

Використовуючи бібліотеку tonsdk, були написані методи для створення криптогаманця, створення NFT, NFT-колекції та для їх розміщення на блокчейні TON.

Для покупки та мінту NFT був розроблений телеграм-бот, що перевіряє транзакцію від користувача, використовуючи TON HTTP API та у разі успішного переказу, починає процес створення NFT одразу на гаманець користувача. З метою захисту від спроб користувачів створювати декілька NFT перевіряючи одну і ту ж транзакцію був розроблений алгоритм хешування даних транзакція, використовуючи хеш-функцію SHA-256, що була відтворена у Python.

Підсумовуючи, можна сказати, що було досягнуто мети роботи та розроблено продукт, що може стати прикладом для реалізації подібних проєктів на блокчейн платформі TON.

Список використаних джерел

1. UACatsDivision [електронний ресурс] - Режим доступу до ресурсу: https://uacatsdivision.com/
2. Non-Fungible Token (NFT): What It Means and How It Works [електронний ресурс] - Режим доступу до ресурсу https://www.investopedia.com/non-fungible-tokens-nft-5115211
3. All Major Blockchain Consensus Algorithms Explained [електронний ресурс] - Режим доступу до ресурсу https://medium.com/@learnwithwhiteboard_digest/all-major-blockchain-consensus-algorithms-explained-6934b4f5d47a
4. Історія блокчейну [електронний ресурс] - Режим доступу до ресурсу https://academy.binance.com/uk/articles/history-of-blockchain
5. A Definitive Guide to Learn The SHA-256 (Secure Hash Algorithms) [електронний ресурс] - Режим доступу до ресурсу https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm
6. The Complete Guide for Types of Blockchain! [електронний ресурс] - Режим доступу до ресурсу https://www.simplilearn.com/tutorials/blockchain-tutorial/types-of-blockchain
7. TON Center [електронний ресурс] - Режим доступу до ресурсу https://toncenter.com/
8. Як бренди заробляють на NFT і метавсесвіті [електронний ресурс] - Режим доступу до ресурсу https://laba.ua/blog/3255-kak-brendy-zarabatyvayut-na-nft-i-metavselennoy
9. TONSDK [електронний ресурс] - Режим доступу до ресурсу https://github.com/tonfactory/tonsdk
10. Linear congruential generator [електронний ресурс] - Режим доступу до ресурсу https://www.wikiwand.com/en/Linear_congruential_generator
11. Tonviewer [електронний ресурс] - Режим доступу до ресурсу https://tonviewer.com/
12. The Open Network [електронний ресурс] - Режим доступу до ресурсу https://docs.ton.org/learn/introduction

13. How Do I Upload Files To IPFS? [електронний ресурс] - Режим доступу до ресурсу https://www.pinata.cloud/blog/how-do-i-upload-files-to-ipfs