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

# Розробка веб-платформи для винагороди за переробку сміття з використанням блокчейн-технологій ICP та Solana

Текстова частина до кваліфікаційної роботи
за спеціальністю
„Інженерія програмного забезпечення” - 121

**Керівник кваліфікаційної роботи**
Гороховський К.С.
_____(підпис)
“____” _____2025 року

**Виконав студент**
ІПЗ-4 Гаврилюк В.Д.
“____” _____2025 року

Київ 2025

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем

ЗАТВЕРДЖУЮ

_____ старший викладач Гороховський К.С.

“___” _____2024р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Гаврилюку Володимиру Дмитровичу

факультету інформатики 4 курсу бакалаврської програми

ТЕМА: **Розробка системи винагород за переробку сміття з використанням технологій блокчейн ICP і Solana**

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання

Вступ

Розділ 1. Дослідження та аналіз предметної області

Розділ 2. Аналіз технічного завдання

Розділ 3. Розробка застосунку

Розділ 4. Результат роботи

Висновки

Список використаної літератури

**Дата видачі „___” _____**2024 р. **Керівник_____**

(*підпис*)

**Завдання отримав_____**

(*підпис*)

**Календарний план виконання роботи**

| № | Назва етапу курсової роботи | Термін виконання етапу | Примітка |
|---|---|---|---|
| 1. | Отримання завдання на кваліфікаційну роботу | 01.10.2024 | |
| 2. | Огляд матеріалів за темою роботи | 15.11.2024 | |
| 3. | Проведення дослідження | 01.02.2025 | |
| 4. | Написання програмного застосунку | 15.03.2025 | |
| 5. | Написання текстової частини | 15.04.2025 | |
| 6. | Захист кваліфікаційної роботи | | |

Студент _____

Керівник _____“_____”_____2025р.

**Зміст**

**Перелік умовних позначень**

**ICP** – Internet computer protocol (Протокол Інтернет-комп'ютера)

**POR** – Proof of Recycling (Доказ переробки сміття)

**Анотація**

Кваліфікаційна робота присвячена аналізу та розробці веб-платформи для винагороди громадян за переробку відходів. Постійне зростання обсягів засмічених територій в Україні спричиняє велику кількість екологічних негараздів, отже, в цьому контексті, створення інструменту, що привертає увагу до цієї проблематики є дуже важливим.

Використання блокчейн-технологій надає можливість побудувати надійний додаток, котрий гарантує коректність даних, пов’язаних з переробкою, та дозволяє розробити прозору систему винагород за допомогою utility-токенів та NFT. Додатково, блокчейн-мережа Internet computer protocol робить створення повноцінних веб-платформ, що повністю працюють на децентралізованих серверах незалежних учасників, легким і доступним. Завдяки «reverse gas model», кінцеві користувачі не повинні мати токени для того, щоб користуватись застосунком – кошти за транзакцію має витрачати розробник. Також, розроблені програми на Solana пропонують гнучкість для вибору користувача, і надають змогу порівняти блокчейн-системи з різними моделями газу.

У роботі буде детально розглянутий процес розробки екологічно корисної ініціативи, враховуючи аргументацію обраних технологічних рішень, аналіз вимог до продукту, а також масштабованість та перспективи у майбутньому веб-додатку.

**Ключові слова:** екологія, блокчейн, ICP, Solana, переробка сміття, децентралізація

# Вступ

## Актуальність теми

Актуальність дослідження зумовлена значним збільшенням засмічених полігонів, площа яких перевищує територію природно-заповідного фонду нашої держави, що спричиняє серйозні наслідки для довкілля. Платформа, що має елементи гейміфікації та дозволяє отримати винагороду за ресайклінг, може суттєво вплинути на ставлення суспільства до екологічних проблем, і також збільшити відсоток утилізованих відходів в Україні, що є особливо важливо, наприклад, у контексті інтеграції до ЄС, де культура і стандарти переробки є дуже високими.

Істотне значення для розробки має також надійний механізм отримання винагород. Забезпечення прозорого отримання та зберігання компенсацій за екологічно корисну поведінку, використовуючи токенізацію та публічні смарт-контракти, сприяє довірі користувачів у рамках цього процесу.

Інноваційність роботи полягає в розробці та розгортанні повноцінного гейміфікованого веб-застосунку з клієнтською і серверною частинами на унікальному за потужностями блокчейні ICP та створенні кросчейнової інтеграції з екосистемою Solana. Винятковий спосіб здійснювати HTTPS-запити поза межами мережі ICP дозволяє налаштовувати API-інтеграції з іншими децентралізованими системами і Web2 сервісами.

Практична цінність дослідження полягає у розробці програмного додатку для заохочення переробляти відходи із системою винагород від брендів-партнерів та використанні найновітніших технологій і рішень. Важливим елементом при роботі з платформою є те, що кінцевий користувач не повинен мати токени, під’єднувати криптогаманці та загалом розумітись у Web3. Така платформа може бути корисною для брендів, які мають на меті залучити спільноту екоактивістів і продемонструвати свою чітку позицію в

підтримці збереження природи, а також для людей, що активно займаються переробкою, задля додаткової мотивації, і для тих, хто прагне спробувати зайнятись ресайклінгом і хоче отримати додаткову винагороду за свої зусилля.

Узагальнюючи, дослідження є актуальним і цінним не лише з наукової та технічної точок зору, а також з практичної і соціальної, надаючи у результаті інструмент сталої мотивації та створення екоспільноти відповідальних громадян.

**Об’єкт дослідження**

Переробки сміття в Україні та отримання винагород за цю активність.

**Предмет дослідження**

Імплементація та розгортання гейміфікованого веб-застосунку у децентралізованій мережі із системою винагород у вигляді токенів за ресайклінг.

**Мета дослідження**

Дослідити можливість використання блокчейн-системи ICP для розробки та розгортання клієнтської і серверної частин гейміфікованої веб-платформи з наявністю кросчейн-функціональності і надійним механізмом заохочення переробки відходів у вигляді побудованої логіки токенів на смарт-контрактах.

**Постановка задачі**

1. Аналіз ринку ресайклінгу та методів для винагород за переробку відходів.
2. Дослідження ICP.
3. Дослідження способів побудувати кросчейн функціональність і інтеграцію із сторонніми Web2-сервісами
4. Розробка власного масштабованого застосунку для мотивації ресайклінгу з елементами гейміфікації, що повністю працює на децентралізованих серверах в блокчейн-мережі.

**Структура роботи**

1. У першому розділі описано загальні відомості про предметну область, розглянуті наявні проблеми та запропоновано, та обґрунтовано підхід до їх вирішення. Крім цього, проаналізовано існуючі рішення на ринку.
2. У другому розділі проаналізовано технічне завдання: описано користувачів і їх функціональні вимоги.
3. У третьому розділі обґрунтовано вибір інструментів, детально описано реалізацію ключових компонентів системи з прикладами програмної реалізації.
4. У четвертому розділі наведено результати роботи програми та зроблені висновки.

# РОЗДІЛ 1. Дослідження та аналіз предметної області

## 1.1 Загальні відомості

Сортування і переробка сміття сьогодні – це не просто сучасний тренд, а справжній спосіб зберегти навколишнє середовище і зменшити руйнівний рівень засмічення. Переробка відіграє вирішальну роль у збереженні природних ресурсів, оскільки зменшує потребу в добуванні сировини з навколишнього середовища. Крім того, переробка також сприяє зменшенню обсягів відходів, що потрапляють на сміттєзвалища або спалюються. Це не лише зменшує навантаження на такі методи утилізації, але й знижує пов’язані з ними негативні наслідки для довкілля, зокрема викиди метану та забруднення ґрунтів.

За даними дослідження «Allied Market Research» [1], глобальний ринок переробки відходів оцінений у 51,7 мільярда доларів у 2023 році, і, за прогнозами, досягне 101,1 мільярда доларів до 2032 року, зростаючи зі середньорічним темпом у 7,8% у період з 2024 по 2032 рік. Як можна побачити, тенденції розвитку є дуже високими, і варто очікувати зріст і надалі.


***Рисунок 1.1*** *прогноз тенденцій зростання світового ринку переробки відходів*

У зв’язку із загальним піднесенням ринку, також зростають можливості для підприємців, котрі хочуть створити «зелений стартап». На прикладі нашої країни, можна знайти чимало грантових програм для фінансування таких ідей. Існують державні ініціативи від Міністерства економіки, що можуть надати до 8 мільйонів гривень для побудови переребного бізнесу [2], а також від приватних акселераторів, таких як «Greencubator» [3], що дозволяють отримати до 10 тисяч євро для стартапів на початковій стадії розробки.

Зростання ринку переробки відходів пришвидшилась у відповідь на глобальні зусилля, спрямовані на зменшення впливу на довкілля та впровадження практик сталого управління ресурсами. Така зацікавленість екологічним питанням від багатьох держав демонструє серйозність проблематики надмірного утворення відходів, що, безумовно, має великий вплив і на Україну.

## 1.2 Проблематика

Екологічні проблеми в нашій державі існують протягом довгого часу і мають серйозний характер. До прикладу, за статистикою 2019 року видання «USA today» Україна входить в рейтинг топ-10 країн, які виробляють найбільшу кількість сміття на 1 особу [4]. Відповідно до даних Міністерства розвитку громад та територій України у 2020 році було утворено близько 55 млн. $м^3$ побутових відходів, з яких 1,7% - спалили і приблизно 3-4% направили на сміттєпереробні лінії.

Варто зазначити, що приблизно 95% із загальної кількості сміття – це небезпечні відходи. Автори дослідження відзначають, що жодна із 105 країн, котрі надають такі дані, не виробляє їх настільки багато. Також, суттєвим фактором є загальний річний обсяг відходів: якщо брати до уваги всі країни

першого десятку цього рейтингу, то за цим показником ми займаємо 3 місце, поступаючись лише США і Канаді.


***Рисунок 1.2*** *Рейтинг країн з найбільшою кількістю сміття на жителя*

Можна по-різному розглядати і аналізувати, чому стільки відходів утворюється в Україні, загалом причини є спільними для більшості світових держав. Через зростання урбанізації, стрімку індустріалізацію та постійне збільшення споживання, обсяги утворення відходів у глобальному масштабі досягли вражаючих розмірів.

Так чи інакше, проблема полягає не лише у великій кількості виробленого сміття, а у відсотку його переробки. Наприклад, серед десяти топ-10 країн вищезгаданого рейтингу, Україна випереджає за рівнем перевикористання залишків лише Сербію (0,8%) та Вірменію, котра взагалі не звітується щодо цього індикатору. Критичною така статистика стає у порівнянні з країнами ЄС, де в середньому 40% сміття йде на переробку, і відповідно до нової законодавчої ініціативи Європарламенту, до 2035 року

усі країни ЄС мають підвищити частку повернення відходів до 65%, отже збільшення обсягів переробки в Україні є важливим з огляду євроінтеграції.

Безперечно, проблеми засмічення не є абстрактними – вони мають вплив на життя українців. Наразі 7 відсотків українських земель займають сміттєзвалища, що співрозмірно площі Данії. Такі показники мають руйнівні наслідки для довкілля, спричиняючи негативний вплив на біорізноманіття, клімат, якість ґрунту, води і повітря. За статистикою Guardian [5], Україна займає перше місце у світі за кількістю смертей від забруденння повітря. Беручи до уваги важку воєнну ситуацію в країні, усі згадані проблеми будуть лише загострюватись.

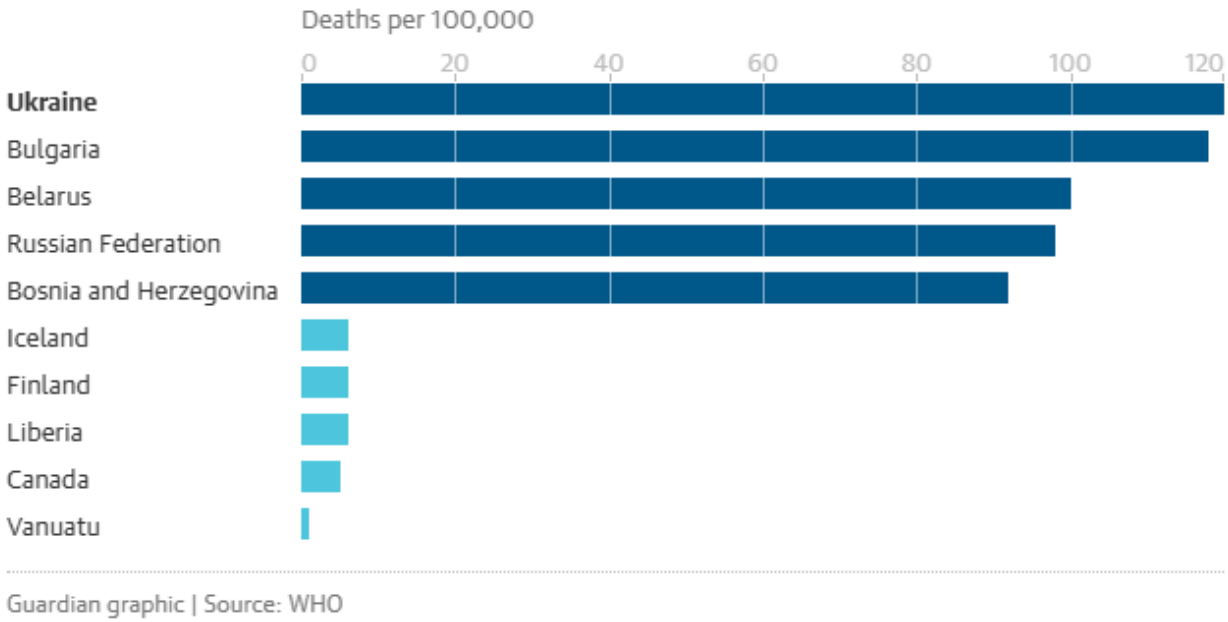



***Рисунок 1.3*** *Рейтинг країн за кількістю померлих від забруднення сміття*

Існує декілька основних причин, чому в Україні не є розвиненою культура сортування і переробки. Серед них можна відзначити [6]:

1. Відсутність належної інфраструктури та регулювання від держави. На відміну від європейських націй, де за неправильне сортування сміття, громадянам виписують значні штрафи, в нашій державі поки що законодавчо це ніяк не регламентується.

Також, існує мала кількість сміттєпереробних станцій і заводів, що робить процес ресайклінгу недоступним і важким.

2. В українців відсутня звичка і мотивація сортувати сміття. Через відсутність зрозумілої винагороди громадяни не бачать сенсу в надлишкових зусиллях, а наявні інструменти заохочення не працюють.

### 1.3 Аналіз наявних рішень

На сьогодні було розроблено декілька подібних рішень, що допомагають розвивати ековідповідальність у громадян України та світу.

До прикладу, вітчизняний застосунок «Кліматичні краплі» [7] пропонував систему відстеження і винагородження екоактивної спільноти за допомогою балів, котрі в застосунку називаються «краплі». Їх можна обміняти на знижки в кафе, ресторанах та інших закладах. «Кліматична крапля» відповідала одному кілограму парникового газу, який не потрапляє до атмосфери внаслідок екологічних дій користувачів. Додаток пропонував нагороду за різні види діяльності: за ходьбу, їзду на велосипеді, переробку і інші. В цілому, платформа мала потенціал, однак через низку недоліків, на жаль, не змогла стати успішною і зупинила роботу. Серед них, окремо виділяються:

1. Платформа була реалізована як мобільний застосунок і мала автономний характер. Інформація про зароблені бали зберігалась на девайсі користувача, отже, після зміни смартфону, він втрачав свої бали. Більше того, таке рішення дозволило зловживати можливостями отримати винагороду через технічні неточності в коді платформи. Баланс користувачів не був загальнодоступним,

тому деякі учасники додатку мали мільйони «накручених» балів, що, зі свого боку, підривало довіру до платформи.

2. Географія «Кліматичних крапель» обмежувалась декількома містами в Україні. Таким чином, відвідувачі системи з інших регіонів могли користуватись нею, однак, винагороду можна було отримати лише в зазначених локаціях.
3. Платформа не мала елементів гейміфікації, таким чином, учасники платформи не були зацікавлені щоденно відвідувати її, якщо не мали потребу.

Беззаперечно, існують і закордонні аналоги цієї ідеї, як-от іспанський «Reciclos» або італійський «Junker app», що пропонують бонуси за перероблення відходів, проте вони не використовують блокчейн-технології, отже, система винагород не є відкритою і прозорою.

## 1.4 Опис обраного підходу до вирішення поставленої задачі

Після детального аналізу наявних на ринку рішень, можна дійти до висновку, що використання блокчейн-технологій у застосунках обраної тематики є доцільним і допомагає вирішити деякі ключові проблеми, з якими стикались платформи з аналогічною ідеєю.

Як рішення було запропоновано реалізацію веб-платформи, що повністю працює в блокчейн-мережі ICP, та надає змогу отримати винагороду у вигляді токенів, завантаживши фотографію з переробної станції зі сміттям як підтвердження своєї екологічно корисної діяльності. Зображення має точно доводити факт того, що користувач здійснив переробку. Перевірка валідності даних користувача про ресайклінг буде здійснюватись на основі API-інтеграції зі штучним інтелектом. Токени, зі свого боку, є валютою, за котру можна придбати право на знижку у мережі

магазинів партнерів, шаблони цих купонів (NFT-токенів) партнер може добровільно розміщати в застосунку. Списання знижки відбувається з використанням згенерованих на основі NFT QR-коду. Стан купону (активний або неактивний) зберігається on-chain, що виключає можливість перевикористати знижку. Елементи гейміфікації у вигляді щоденних квестів і опитувань допомагають учасникам застосунку більше дізнатись про принципи сортування та отримати додаткові бонуси, що викликає зацікавленість з боку користувачів у частому відвідуванні додатку. Створена платформа має назву «**Proof of recycling**» («Доказ переробки»).

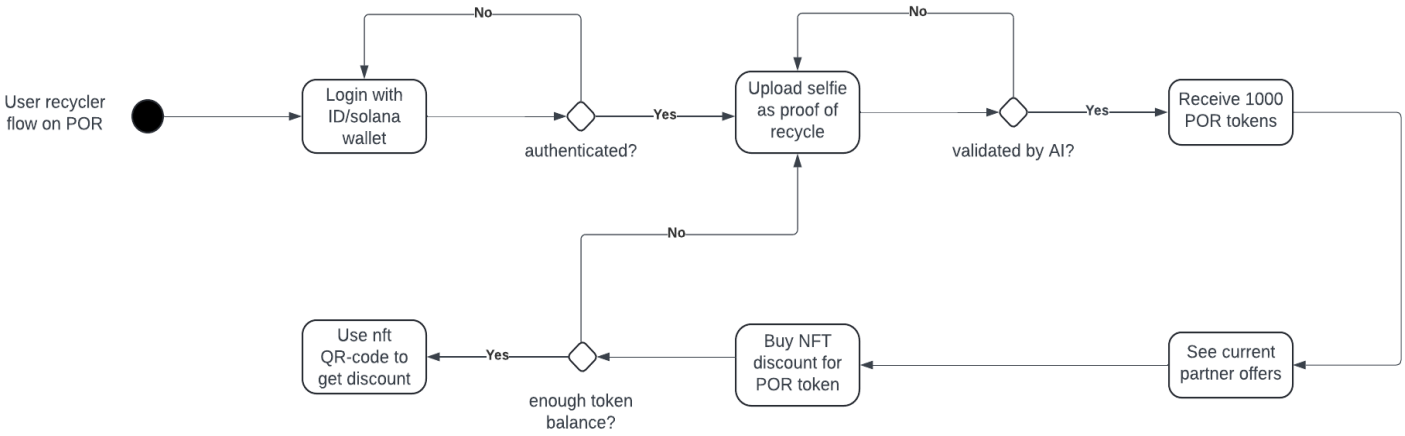



***Рисунок 1.4*** *UML-діаграма активностей базового прикладу роботи з платформою користувача, що переробляє сміття*

## 1.5 Обґрунтування вибору рішення

Застосовуючи децентралізований підхід, можна позбутись недоліків попередників: побудувати прозору і гнучку систему винагород на основі utility-токенів і NFT, що буде працювати на децентралізованих серверах, і якою неможливо зловживати. Таким чином, партнери, котрі надають можливість отримати знижку в додатку, можуть бути впевнені у достовірності права на бонуси, що є критично важливим в контексті їх бізнесу і позбавить ризику фінансових втрат. До того ж, децентралізований характер збереження даних дозволить користувачам отримувати інформацію

та докази переробки на будь-якому пристрої, вирішуючи проблематику автономності, при цьому джерело записів є відкритим і спільним для усіх.

Додатковим плюсом також є Web3-спільнота, члени якої ідеально підходять під опис цільової аудиторії: міська молодь з прогресивними поглядами і відповідальною позицією. Цей фактор може допомогти залучити перших користувачів на платформу.

### 1.5.1 Обґрунтування вибору екосистеми блокчейну ICP

Internet computer protocol (ICP) є ключовою технологією, що лягла в основу проєкту. ICP – це інноваційна блокчейн-мережа, розроблена фондом DFINITY, котра дозволяє повноцінно розробляти і розгортати клієнтську та серверну частини веб-застосунків на децентралізованих серверах. Вона є у мільйони разів потужнішою та здатна замінити хмарні сервіси й традиційні IT-системи [8]. Цю мережу - що працює на протоколі ICP - координує децентралізоване управління без дозволів, а розміщується вона на суверенних апаратних пристроях, якими керують незалежні учасники. Її мета - розширити публічний інтернет, додавши до нього вбудовану функціональність хмарних обчислень. ICP має велику кількість особливостей, що роблять його унікальним у своєму роді і дозволяють легко та швидко будувати гнучкі веб-застосунки. Серед головних переваг можна виділити:

1. Смарт-контракти в блокчейні ICP можуть використовувати сотні гігабайтів пам’яті з використанням «stable memory» [9] та обчислюватися з повною швидкістю сучасного процесора, що на кілька порядків перевищує можливості смарт-контрактів Ethereum.
2. На відміну від більшості децентралізованих систем, смарт-контракти в ICP можуть самостійно надсилати HTTPS-запити до зовнішніх сервісів,

тим самим вирішуючи «The oracle problem» [10]. Усунення потреби в зовнішніх оракулах надає значну кількість покращень, забезпечуючи детермінованість і дозволяючи напряму взаємодіяти з API, базами даних та смарт-контрактами інших блокчейнів.

3. ICP застосовує «reverse gas model», що пропонує розробникам попередньо оплачувати комісії за газ, наповнюючи свої смарт-контракти так званими «циклами». Це дає змогу користувачам взаємодіяти з додатком без потреби мати токени чи гаманець. Такий підхід спрощує вхід у Web3, забезпечуючи користувацький досвід, подібний до традиційних вебзастосунків, і сприяє легшому впровадженню технології.
4. Смарт-контракти, що працюють на ICP, є потужною еволюцією традиційних смарт-контрактів і їх називають каністри. Це обчислювальні одиниці, які поєднують у собі як код, так і дані. Каністри можна використовувати для найрізноманітніших цілей - від надання веб-сторінок для клієнтського застосунку, написаних на JS-фреймворку, до створення захищеного месенджера або реалізації децентралізованої біржі токенів. Вони можуть спілкуватися з користувачами або з іншими каністрами, а також управлятися такими структурами, як DAO (децентралізовані автономні організації). Каністри можуть бути написані різними мовами, котрі здатні компілюватись до стандарту WASM, основними є Rust і Motoko.

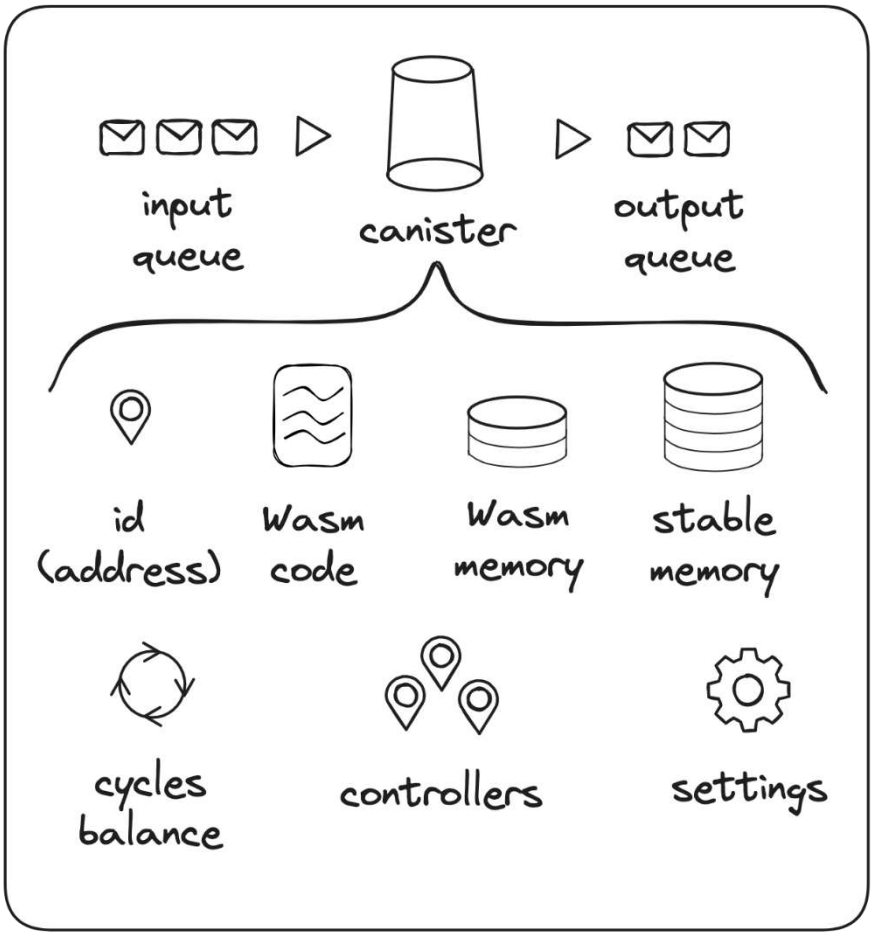



***Рисунок 1.5*** *Основні компоненти смарт-контракту ("каністри") в ICP*

5. Internet computer protocol має вбудовану систему автентифікації - Internet Identity, що керує сесіями на основі криптографії. Коли користувач підключається до децентралізованого застосунку, створюється нова пара відкритого та закритого ключів для сесії, яку Internet Identity засвідчує за допомогою поєднання WebAuthn [11] і порогового підпису (threshold signing).
6. Використовуючи інструмент збірки dfx, процес розгортання каністер в локальному середовищі є доволі простим, що дозволяє розробникам швидко демонструвати результати їх роботи. Водночас, перехід між локальною мережею і mainnet потребує виконання лише декількох термінальних команд та поповнення каністер циклами, що дає змогу легко переконфігурувати проєкт для промислового використання.

Після ґрунтовного аналізу потужних можливостей екосистеми Інтернет-комп’ютера, вона була обрана як основне архітектурне рішення для розробки і деплойменту веб-застосунку. Каністри в ICP дозволяють побудувати не лише смарт-контракти, а повноцінні бекенди зі сховищем даних, котре може мати розмір до 500 гігабайт. Здатність роздавати веб-сторінки робить можливим створення фронтенд частини застосунку, у тому числі з використанням популярних нині JS-фреймворків.

### 1.5.2 Обґрунтування вибору блокчейну Solana

Зі свого боку, Solana – це більш класична блокчейн-платформа, створена організацією «Solana labs» у 2017 році. Вона не надає можливості розробляти повноцінні веб-застосунки з клієнтською частиною та не дає змоги робити HTTPS-outcalls, проте пропонує модель побудови швидких децентралізованих додатків на базі смарт-контрактів і DAO [12].

Алгоритм консенсусу «Proof of History» [13] є ядром блокчейну Solana. Цей механізм відіграє центральну роль у забезпеченні високої пропускної здатності блокчейну ефективним способом. По суті, ця функція відповідає за перевірку проміжку часу між двома подіями. Вона підтримує створення міток часу та хешів у блокчейні. Такий підхід є унікальним порівняно зі звичайним послідовним процесом верифікації. Завдяки використанню «Proof of History» у поєднанні з технікою таймстампінгу ефективність системи значно підвищується. Додатковими перевагами блокчейну є надійність і надзвичайно висока швидкість обробки транзакцій, Solana теоретично може здійснювати обробку 65 тисяч паралельних операцій. Ця висока пропускна здатність досягається завдяки комбінації технологій «Proof of History» та Proof of Stake», які сприяють її швидкості та ефективності.

Як було зазначено, Solana не дає змогу створювати повноцінні веб-сайти, проте написані на ній програми доцільно використати у розробленій на ICP платформі, продемонструвавши можливість Інтернет-комп’ютера робити інтеграції з іншими децентралізованими системами. Ця технологія дозволяє написати блокчейн-логіку для надання винагороди користувачам, використовуючи відповідні стандарти SPL токенів. Крім того, інтеграція Solana в проєкт уможливлює порівняння блокчейнів з різними моделями газу, адже в ньому користувач самостійно платить за комісію для здійснення транзакцій. До того ж, на сьогодні спільнота Solana є дуже потужною і більш популярною аніж в ICP, як наслідок, інтеграція з цією технологією може привернути більше уваги від Web3-громади.

# РОЗДІЛ 2. Аналіз технічного завдання

## Головна мета застосунку

Забезпечити зручний, надійний і прозорий інструмент для отримання винагороди за переробку сміття у вигляді токенів, котрі можна обміняти на знижки партнерів. Імплементувати функціональність відслідковування екологічних активностей користувача. Гарантувати відсутність можливості повторно застосовувати раніше використану знижку.

## 2.1 Опис ролей в системі

На сьогодні застосунок має дві основні ролі – Recycler та Partner, а також супер роль – Admin.

## 2.2 Технічні вимоги до ролей, набутих користувачем системи

**Recycler:**

Користувач, котрий надає докази переробки сміття і отримує за це токени. Його можливості включають:

- Завантажити фото-доказ ресайклінгу з переробної станції із описом, у тому числі, з автоматично визначеною геолокацією
- Отримати токени у разі підтвердження від AI, що фото є справді підтвердженням перероблення відходів
- Отримати токени у випадку успішного проходження щоденних квестів, включаючи тестування знань про екологію
- Переглянути доступні пропозиції від партнерів на сторінці бонусів для обміну балансу токенів на NFT-знижки

- Переглянути власну NFT-колекцію із їх поточними статусами (використані чи доступні)

**Partner:**

Користувач, котрий може розміщувати NFT, що дає право на знижку, на платформі. Йому доступні такі функції:

- Створювати шаблон NFT, котрий з'явиться на сторінці пропозицій і буде доступний для мінту користувачу «**recycler**»
- Сканувати QR-коди NFT для знижки, роблячи їх недоступними для повторного використання
- Перегляд статистики мінту NFT-токенів та використання знижок

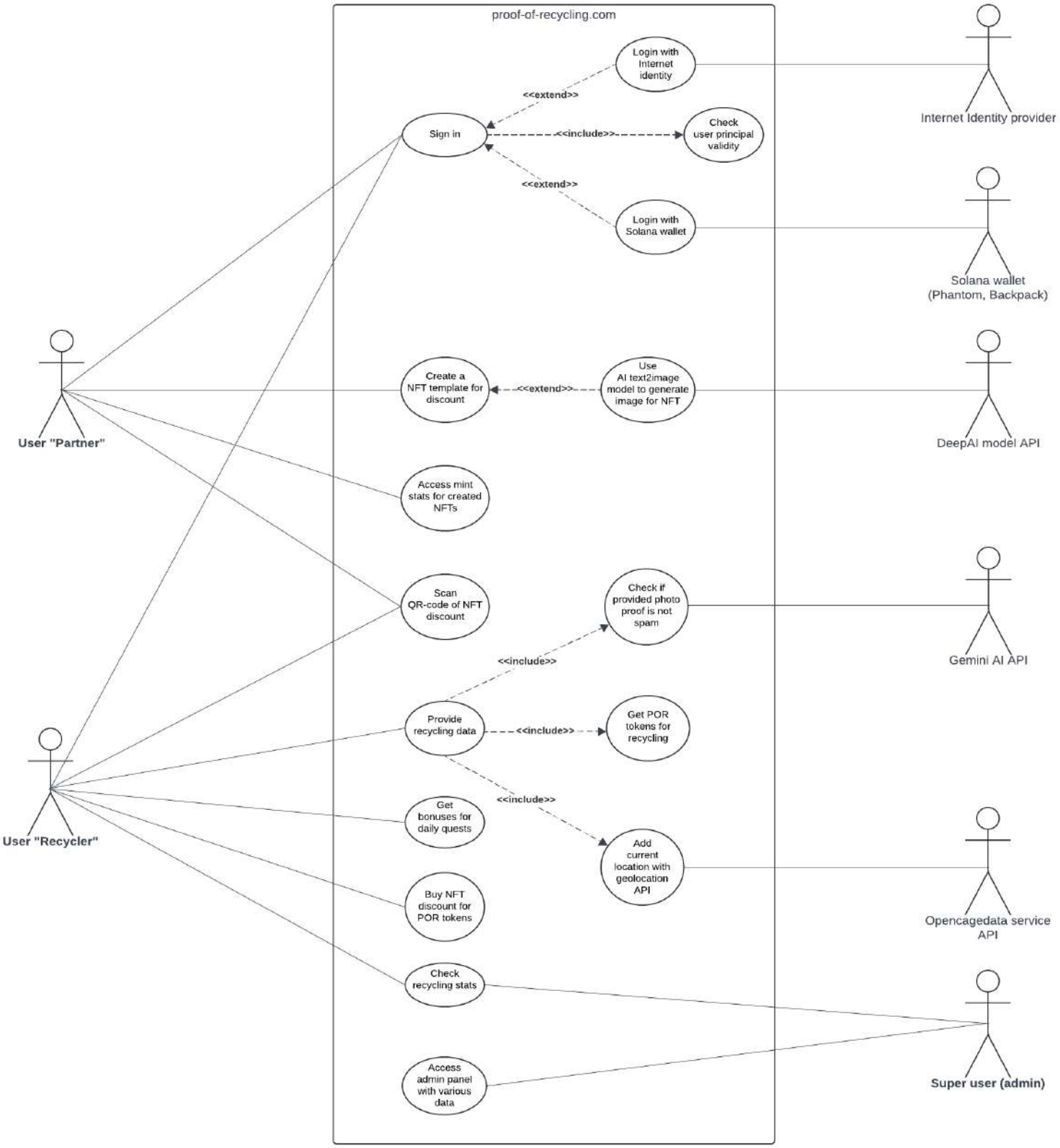



***Рисунок 2.1*** *UML use-case діаграма для різних користувачів системи*

# РОЗДІЛ 3. Розробка застосунку

## 3.1 Опис загальної архітектури

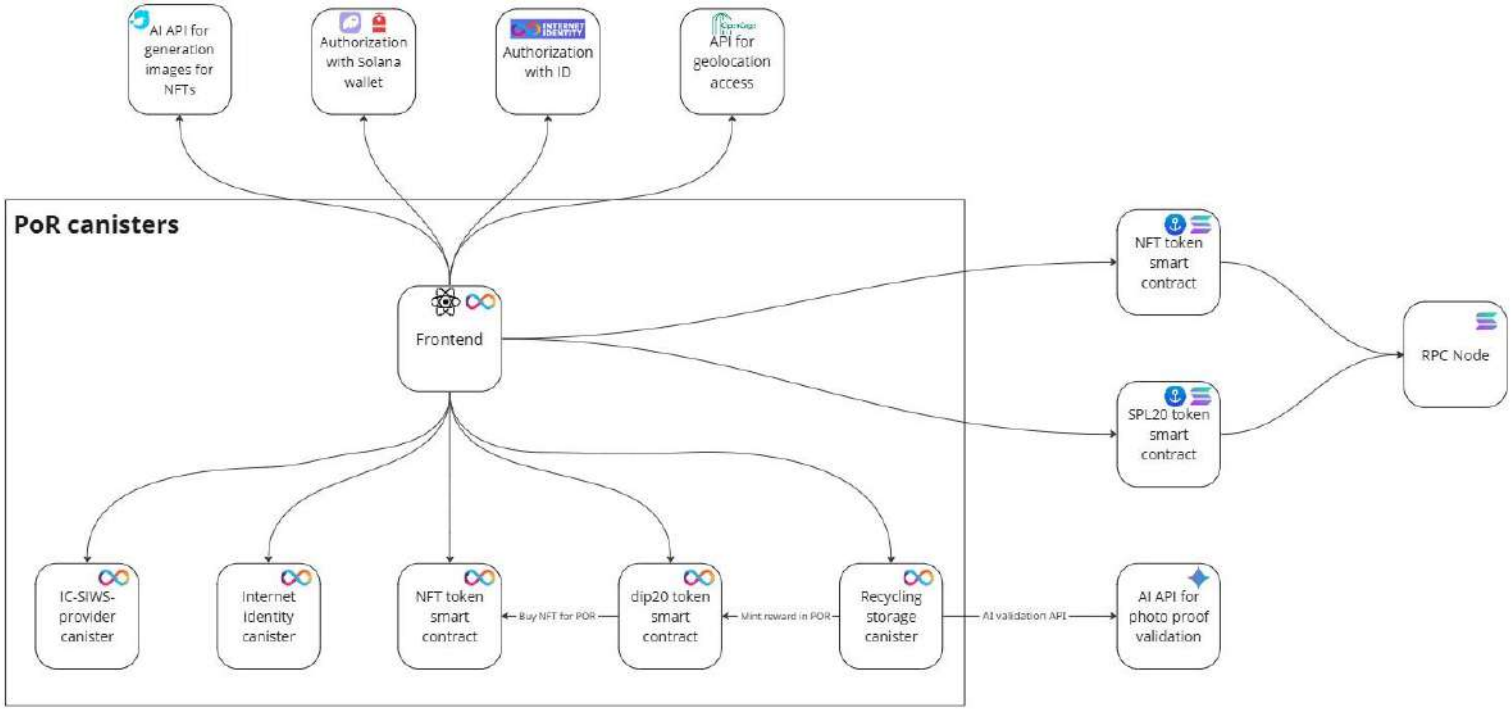



***Рисунок 3.1*** *Загальна архітектура застосунку*

Веб-застосунок використовує класичний варіант клієнт-серверної взаємодії, маючи окремий SPA-застосунок, що працює в браузері, а також розділені сервіси для серверної та блокчейн логіки. Основа платформи була написана на ICP та розгорнута в каністрах. У цьому контексті, каністру можна розглядати як сервіс у мікросервісній архітектурі Web2 застосунків, котрий може взаємодіяти з іншими частинами системи або зі сторонніми API.

Клієнтська частина, написана на React.js, також розміщена в мережі Інтернет-комп’ютера, хоча сьогодні більшість децентралізованих застосунків (dApps), побудованих на інших блокчейнах, покладаються на централізованих хмарних провайдерів (наприклад, AWS, GCP) для хостингу фронтенду та інших компонентів, що створює єдину точку відмови. Типовий

підхід у галузі полягає в тому, що смарт-контракт виконує певні обчислення, однак фронтенд часто розміщується на сервері з Node.js [14]. Це не лише створює ризики безпеки через можливе втручання у фронтенд, але й призводить до того, що децентралізована платформа, яка працює на централізованому сервері, може бути вимкнена в будь-який момент, що суперечить самій природі блокчейну як захищеного від цензури середовища.

Web3 вимагає, щоб усі частини децентралізованого застосунку реалізовувалися у вигляді смарт-контрактів. Крім того, функціональність Web3 можлива лише в тих блокчейн-застосунках, які повністю працюють у децентралізованому середовищі — тобто включають логіку смарт-контрактів, зберігання даних на ланцюгу та подання інтерфейсу користувача (фронтенду) безпосередньо у браузер користувача.

Написані смарт-контракти на Інтернет-комп'ютері, окрім логіки розподілення токенів, додатково дають змогу створювати повноцінні бекенди зі сховищем даних, використовуючи «Stable memory». Система авторизації Internet identity також впроваджена в застосунку і гарантує надійний механізм безпеки від неавторизованого доступу.

Окрім раніше згаданих сервісів була розроблена Solana-програма з використанням фреймворку Anchor. Її логіка є аналогічною до логіки отримання винагорода на ICP, однак користувач має попередньо підключити свій криптогаманець до платформи, щоб сплачувати комісію за транзакції. Смарт-контракти на Solana працюють незалежно від основної частини застосунку і їх інструкції викликаються з фронтенду.

До того ж, налаштовані API-інтеграції до зовнішніх сервісів, що спрощує роботу користувача на платформі і демонструє можливість Інтернет-комп'ютера робити зовнішні HTTPS-виклики.

### 3.2 Основні засоби розробки та їхнє використання

### 3.2.1 ICP-каністри

Як раніше згадувалось, смарт-контракти, що написані та розгорнуті в мережі Інтернет-комп'ютера мають назву «каністри» [15]. Каністри - це вдосконалена форма традиційних смарт-контрактів. Вони мають багато спільних властивостей із традиційними смарт-контрактами, наприклад, їх стан можна перевірити та криптографічно засвідчити, однак вони відрізняються від типових програм на блокчейні, маючи такі переваги:

- WASM-модуль каністри може бути оновлений до нової версії для додавання нових функцій або виправлення помилок;
- каністри можуть підписувати та надсилати транзакції безпосередньо в інші блокчейн-мережі;
- до того ж, вони можуть звертатися до зовнішніх сервісів, щоб отримати дані і використовувати їх.

Для застосунку було розроблено 3 основні каністри (і використані 2 готові). Каністра «**storage**» відповідає за збереження інформації користувача про ресайклінг, а також робить запити до інших контрактів і до Gemini API для валідації фотографії переробки. Смарт-контракт «**dip20**» імплементує стандарт взаємозамінних токенів для ICP, що застосовуються у платформі у вигляді POR-токенів, котрі є внутрішньою валютою додатку. Каністра «**nft**» реалізує типовий смарт-контракт для логіки невзаємозамінних токенів, що використовуються як доказ користувача на право знижки у партнера.

Смарт-контракти в ICP можуть бути написані мовами програмування, однак найбільш підтримуваною є Rust, тому я вирішив використати саме її. Rust - це мова системного програмування, що протягом останніх років стає все більш популярною. Вона надає пріоритет продуктивності, безпеці роботи з пам'яттю та паралелізму. Rust відомий своєю сильною системою типів, яка

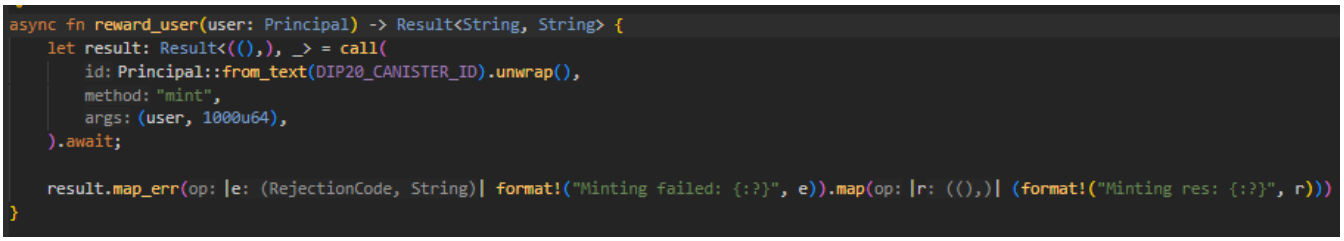
допомагає виявляти помилки на етапі компіляції, та інноваційним механізмом перевірки запозичень (borrow checker), що запобігає помилкам, пов'язаним із пам'яттю, без використання garbage collector.

Для розробки додатків на ICP за допомогою Rust я використовую Rust Canister Development Kit (CDK). CDK взаємодіє з IC SDK, щоб надати мові програмування функціональність для створення, розгортання, виклику та керування каністерами. Rust CDK та інші Rust бібліотеки забезпечують підтримку повного набору функцій ICP, включно з HTTP-сертифікацією, композитними запитами та статистикою запитів, тоді як інші CDK можуть мати обмежену підтримку деяких розширених можливостей.

```rust
use ic_cdk::{api::management_canister::http_request::{self, http_request, CanisterHttpRequestArgument, HttpHeader, HttpMethod}, init, query, update};
use serde_json::json;
use std::collections::HashMap;
use std::sync::Mutex;
use ic_cdk::api::caller;
use ic_cdk::api::call::call;
use lazy_static::lazy_static;
use candid::Principal;

type PrincipalId = String;


#[derive(CandidType, Deserialize, Clone)]
4 implementations
struct RecycleData {
    image: Vec<u8>,
    comment: String,
    location: String,
    principal_id: PrincipalId,
    created_at: u128 ,
}

#[update]
async fn store_data(image: Vec<u8>, comment: String, location: String, created_at: u128) -> Result<String, String> {
    if !validate_image_with_gemini(image.clone(), &location, &comment).await? {
        return Err("AI rejected the image as invalid.".to_string());
    }
    let principal_id: String = caller().to_string();
    let recycle_data: RecycleData = RecycleData { image, comment, location, principal_id: principal_id.clone(), created_at };

    let mut storage: MutexGuard<'_, HashMap<String, ...>> = STORAGE.lock().unwrap();
    let user_records: &mut Vec<RecycleData> = storage.entry(principal_id.clone()).or_insert(vec![]);
    user_records.push(recycle_data);
    reward_user(caller()).await;
    Ok(principal_id)
}
```

***Рисунок 3.2*** *Використання IC CDK і функція зберігання даних про переробку*

Крім того, IC CDK надає можливість викликати методи інших каністер за допомогою метода «call», таким чином, можна побудувати функціональну бекенд-систему на взаємопов'язаних смарт-контрактах.



```
async fn reward_user(user: Principal) -> Result<String, String> {
    let result: Result<((),), _> = call(
        id: Principal::from_text(DIP20_CANISTER_ID).unwrap(),
        method: "mint",
        args: (user, 1000u64),
    ).await;

    result.map_err(op: |e: (RejectionCode, String)| format!("Minting failed: {:?}", e)).map(op: |r: ((),)| (format!("Minting res: {:?}", r)))
}
```

***Рисунок 3.3*** *Виклик методу каністри "dip20" з "storage"*

Великою перевагою смарт-контрактів на ICP є опція застосовувати їх не лише для розробки логіки, а як і сховище даних. Під час розробки проєктів на ICP існують два основні типи сховищ даних, які можуть використовувати каністри. Перший тип - це heap-пам’ять каністри, іноді її називають основною пам’яттю каністри. Heap-пам’ять є тимчасовою, і будь-які дані, що зберігаються в heap-пам’яті каністри, очищуються щоразу, коли каністру перевстановлюють або оновлюють. Другий тип сховища, доступний каністрі, - це стабільна пам’ять. Стабільна пам’ять - це унікальна особливість ICP, яка визначає окреме сховище даних, відмінне від звичайної Wasm heap-пам’яті каністри. Це другорядний тип пам’яті, який можна використовувати для довготривалого зберігання даних, оскільки всі дані, що зберігаються у стабільній пам’яті, зберігаються між усіма процесами каністри. Реалізація «stable memory» залежать від вибору мови розробника, наприклад, для Rust потрібно імпортувати крейт «ic-stable-structures» [16], що робить можливим використання звичайних структур даних, таких як вектор чи TreeMap, однак при цьому загальний ліміт стабільної пам’яті в одній каністрі може досягати 500 гігабайт. Для мого проєкту більш доцільне використання саме постійного сховища даних. Також, «stable memory» робить можливим зберігання файлів у вигляді байтів, до прикладу, саме таким способом я зберігаю фотографію-доказ переробки.

```rust
use ic_cdk_macros::init;
use ic_stable_structures::memory_manager::{MemoryId, MemoryManager, VirtualMemory};
use ic_stable_structures::{StableBTreeMap, DefaultMemoryImpl, Storable};
use std::cell::RefCell;
use candid::Principal;
use serde::{Serialize, Deserialize};

type Memory = VirtualMemory<DefaultMemoryImpl>;


thread_local! {
    static MEMORY_MANAGER: RefCell<MemoryManager<DefaultMemoryImpl>> = RefCell::new(
        MemoryManager::init(DefaultMemoryImpl::default())
    );

    static USERS: RefCell<StableBTreeMap<Principal, User, Memory>> = RefCell::new(
        StableBTreeMap::init(
            MEMORY_MANAGER.with(|m| m.borrow().get(MemoryId::new(0)))
        )
    );

    static BRANDS: RefCell<StableBTreeMap<String, Brand, Memory>> = RefCell::new(
        StableBTreeMap::init(
            MEMORY_MANAGER.with(|m| m.borrow().get(MemoryId::new(1)))
        )
    );
}
```

***Рисунок 3.4*** *Застосування структури з постійною пам'яттю StableBTreeMap в проєкті*

Як уже зазначалось раніше, одним з найбільш плюсів Інтернет-комп’ютера є його здатність робити HTTPS-виклики поза власною мережею, не використовуючи оракли. Я використовую цю функцію, щоб делегувати перевірку фото-доказу користувача, коли він надає інформацію про переробку, як наслідок, якщо користувач надав будь-яке невідповідне зображення, то він не зможе отримати винагороду.

```rust
async fn validate_image_with_gemini(
    let body: String = serde_json::to_string(&payload).unwrap();

    let request: CanisterHttpRequestArgument = CanisterHttpRequestArgument {
        url: format!(
            "https://generativelanguage.googleapis.com/v1beta/models/gemini-pro-vision:generateContent?key={}",
            GEMINI_API_KEY
        ),
        method: HttpMethod::POST,
        headers: vec![HttpHeader {
            name: "Content-Type".to_string(),
            value: "application/json".to_string(),
        }],
        body: Some(body.into_bytes()),
        max_response_bytes: Some(3000),
        transform: None,
    };

    let (response: HttpResponse,) = http_request(arg: request, cycles: 200).await Result<(HttpRespon…
    .map_err(op: |e: (RejectionCode, String)| format!("HTTP request error: {:?}", e))?;

    if response.status != Nat::from(200u32) {
        return Err(format!("Gemini returned status: {}", response.status));
    }

    let response_text: String = String::from_utf8(vec: response.body) Result<String, FromUtf8Error>
        .map_err(op: |_| "Invalid UTF-8 in response".to_string())?;
    let response_json: serde_json::Value = serde_json::from_str(&response_text) Result<Value, Error>
        .map_err(op: |_| "Invalid JSON in Gemini response".to_string())?;
    let text_response: String = response_json["candidates"][0]["content"]["parts"][0]["text"] Value
        .as_str() Option<&str>
        .unwrap_or(default: "") &str
        .to_lowercase();

    Ok(text_response.contains("yes"))
} fn validate_image_with_gemini
```

***Рисунок 3.5*** *Частина коду функції для формування запиту на перевірку доказу переробки з Gemini API*

### 3.2.2 Internet identity

Internet Identity – це вбудована, система автентифікації на блокчейні, побудована на криптографічних засадах і підтримується платформою ICP. На відміну від більшості блокчейнів, де потрібно проходити автентифікацію щоразу при виконанні запиту, Інтернет-комп’ютер дозволяє тимчасово й безпечно делегувати автентифікацію децентралізованого застосунку після підключення. Це стало можливим завдяки створенню сесій під час використання застосунку, що ґрунтується на криптографії з використанням chain-key.

У своїй роботі я застосовую цю технологію, адже я хочу надати доступ до своїх смарт-контрактів лише авторизованим користувачам. Для інтеграції з Internet Identity потрібно розгорнути в проєкті pre-built каністру, додати деякі конфігурації і, до того ж, встановити на JS-клієнті бібліотеку «dfinity/auth-client», яка дозволяє керувати доступами користувача в застосунку.

```
import toastNotifications from '../../utils/toastNotifications.utils';
import { AuthClient } from '@dfinity/auth-client';

const network = process.env.DFX_NETWORK;
const identityProvider =
  network === 'ic'
    ? 'https://identity.ic0.app'
    : 'http://rdmx6-jaaaa-aaaaa-aaadq-cai.localhost:4943';

function LoginPage() {
  const { login: loginWithSolana, loginStatus, identity: solanaIdentity } = useSiws();
  const { setSolanaIdentity } = useAuth();
  const wallet = useWallet();
  const [isConnectingIdentity, setIsConnectingIdentity] = useState(false);

  async function loginWithInternetIdentity() {
    setIsConnectingIdentity(true);
    const authClient = await AuthClient.create();
    await authClient.login({
      identityProvider,
      onSuccess: () => {
        updateActor().then(() => {
          navigate(ApplicationRoutes.Profile);
        });
        console.log('Login Successful!');
      },
    });
    setIsConnectingIdentity(false);
  }
```

***Рисунок 3.6*** *Використання авторизації за допомогою Internet Identity*

### 3.2.3 Solana

Доповненням до основної частини платформи є написання Solana-програм та їх інтеграції в застосунок. Для написання смарт-контрактів на Solana я обрав фреймворк Anchor на мові Rust. Anchor - це провідна платформа для розробки програм на Solana, котра спрощує процес

написання, тестування, розгортання та взаємодії з програмами [17]. З використанням цього фреймворку, мені вдалось написати функціональність винагород аналогічну до тієї, що спроєктована в ICP каністрах «**dip20**» і «**nft**». Таким чином, якщо при вході на платформу обрати спосіб авторизації з криптогаманцем (Phantom, Backpack чи іншим), то клієнтська частина веб-застосунку буде звертатись до Solana-програми, і як наслідок, нагорода буде нараховуватись на баланс обраного криптогаманця.

Головний смарт-контракт відповідає за логіку NFT-токену, що використовується для отримання знижок в закладах партнерів. Інформація про автора токена і його статус зберігаються on-chain, що унеможливлює зміну цих данів без необхідних дозволів. Для метаданих NFT в проєкті використовується Metaplex - це набір інструментів і смартконтрактів для створення, зберігання та управління NFT на блокчейні Solana. Він забезпечує стандарт для NFT через Token Metadata Program, що дозволяє зберігати всю інформацію про токен: назву, опис, зображення, атрибути тощо. JSON із потрібними властивостями для метаданих зберігається в децентралізованому сховищі Pinata. Програма була задеплоєна в devnet і має адресу «*4jwVxfKwLtXhAkPxpWuiZMHB5deB4U7At6mkCrZUGonX*».

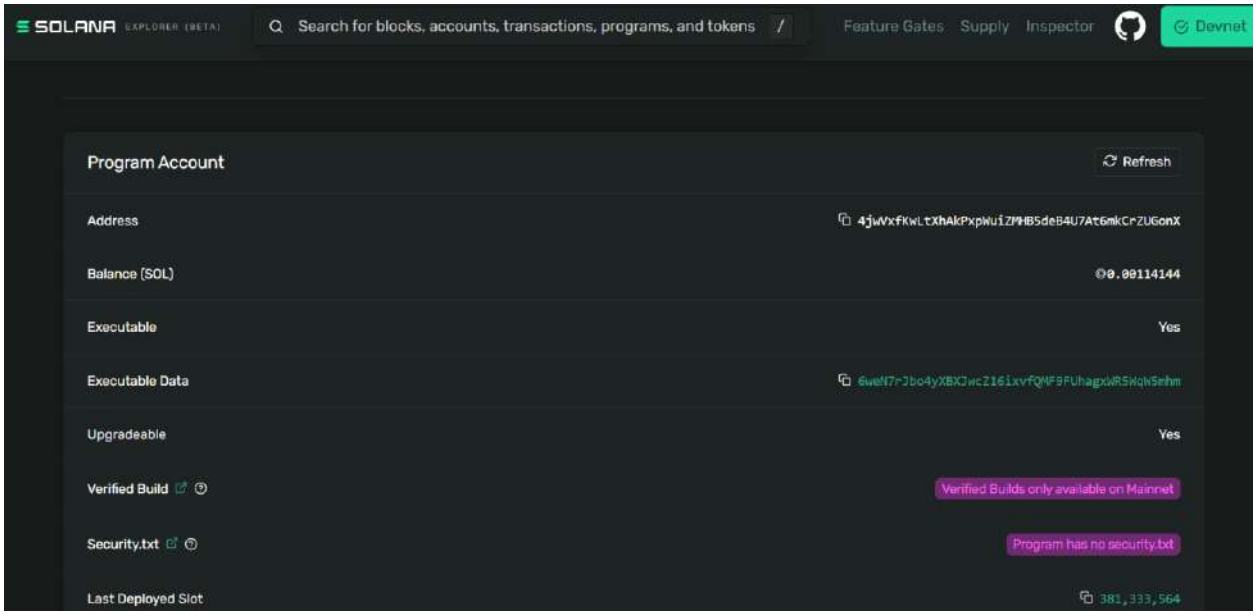



***Рисунок 3.7*** *Програма в Solana explorer*

```rust
use mpl_token_metadata::{
    instructions::{
        CreateMasterEditionV3, CreateMasterEditionV3InstructionArgs, CreateMetadataAccountV3,
        CreateMetadataAccountV3InstructionArgs,
    },
    types::DataV2,
};

declare_id!("4jwVxfKwLtXhAkPxpWuiZMHB5deB4U7At6mkCrZUGonX");

#[program]
pub mod por_nft {
    use super::*;

    #[derive(Accounts)]
    pub struct UseNft<'info> {
        #[account(mut, has_one = owner, has_one = mint)]
        pub nft_status: Account<'info, NftStatus>,
        pub owner: Signer<'info>,
    }

    pub fn use_nft(ctx: Context<UseNft>) -> Result<()> {
        let status = &mut ctx.accounts.nft_status;
        require!(!status.is_used, CustomError::NftAlreadyUsed);

        status.is_used = true;
        Ok(())
    }

    pub fn mint_nft(
        ctx: Context<MintNft>,
        name: String,
        symbol: String,
        uri: String,
    ) -> Result<()> {
        let authority_seeds: &[&[&[u8]]] = &[&[b"mint-authority", &[ctx.bumps.program_authority]]];

        create(CpiContext::new(
            ctx.accounts.associated_token_program.to_account_info()
```

***Рисунок 3.8*** *Код смарт-контракту для NFT на Anchor*

### 3.2.4 Інтеграція криптогаманця Solana в ICP-проєкт

Для того, щоб взаємодіяти з каністрами, користувач повинен бути авторизованим і мати прив’язану до його сесії Internet Identity. Однак, якщо користувач буде авторизовуватись через криптогаманець Solana, то він не зможе автоматично викликати методи ICP смарт-контрактів, для цього потрібен мапінг публічної адреси гаманця до новоствореної Internet Identity. Саме тому я використовую в своєму застосунку рішення від «ic-siws» - це проєкт, який забезпечує автентифікацію на основі гаманця Solana для застосунків на ICP (Internet Computer). Метою цього проєкту є покращення взаємодії між Solana та платформою Internet Computer, що дозволяє

розробникам створювати застосунки, які поєднують переваги обох екосистем. Застосування цієї бібліотеки гарантує, що вхід через Solana-гаманець завжди використовує один і той самий Principal (ідентифікатор ICP), незалежно від клієнта або пристрою, створюючи однозначну відповідність між Solana-адресами і Principal-ідентифікатором.

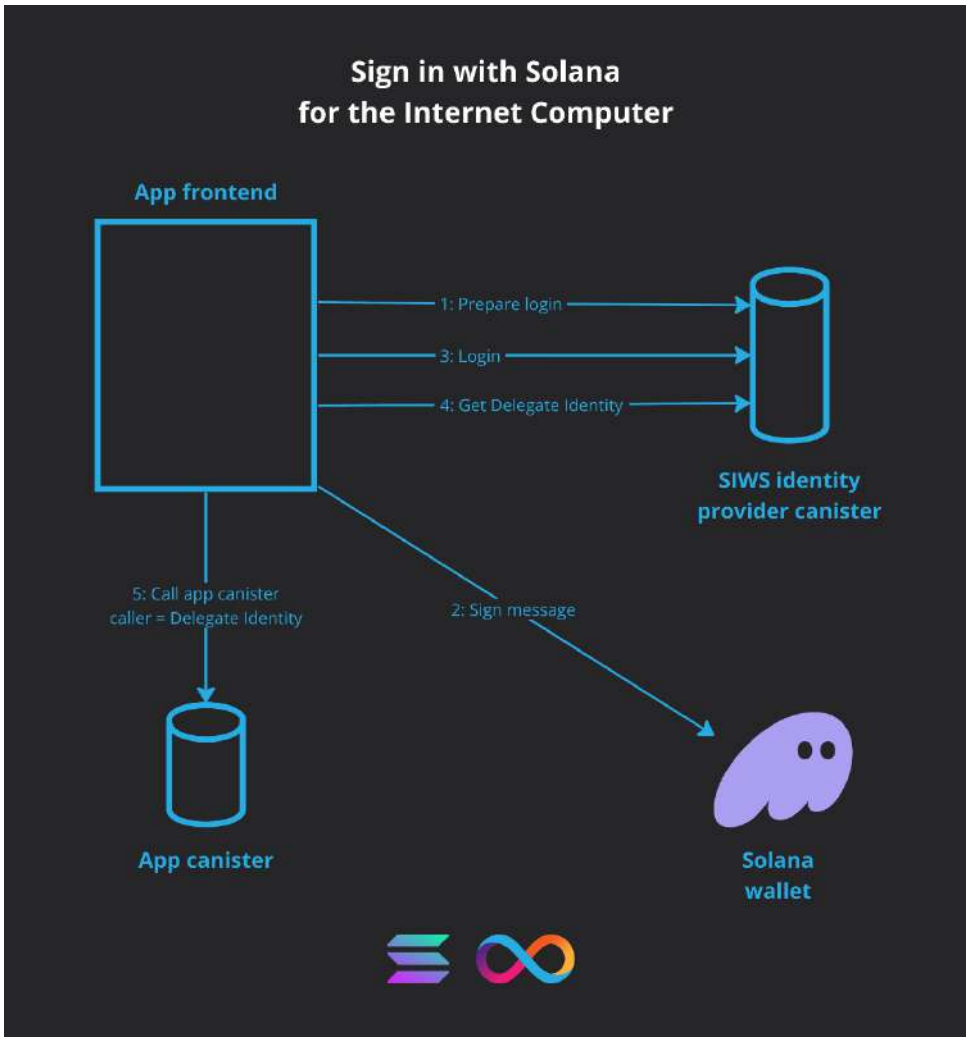



***Рисунок 3.9*** *Алгоритм роботи проєкту ic-siws*

На верхньому рівні цей механізм працює так [18]:

1. Фронтенд за допомогою ic-siws-js надсилає запит до каністри для отримання SIWS-повідомлення для входу.
2. Каністра генерує SIWS-повідомлення та повертає його.

3. ic-siws-js запитує у користувача підписати повідомлення через його гаманець Solana.
4. Підписане повідомлення надсилається назад до каністри через ic-siws-js.
5. Каністра перевіряє підпис і видає делеговану ICP-ідентичність (Principal).
6. Фронтенд використовує делеговану ідентичність для автентифікованих викликів до інших каністр.

Отож, проаналізувавши алгоритм роботи цього проєкту, я розгорнув у своїй платформі pre-built каністру «ic-siws-provider» та імпортував залежність для фронтенда з підтримкою SIWS (Sign-In With Solana), котра має назву «ic-siws-js». Вона надає підтримку хуку «useSiws», що робить менеджмент авторизації простим і зручним. Крім того, щоб мати змогу викликати криптогаманець на клієнті, потрібно скористатись бібліотеками сімейства «@solana/wallet-adapter». Вони надають готовий компонент WalletProvider, в котрий потрібно огорнути весь проєкт, після чого користувач зможе взаємодіяти зі своїм гаманцем за допомогою попередньо встановленого у нього розширення браузеру чи мобільний застосунок.

```
import { WalletMultiButton } from '@solana/wallet-adapter-react-ui';
import { useWallet } from '@solana/wallet-adapter-react';
import { useSiws } from 'ic-siws-js/react';
import { useAuth } from '../../hooks/auth.hooks';
import toastNotifications from '../../utils/toastNotifications.utils';

const network = process.env.DFX_NETWORK;
const identityProvider =
  network === 'ic'
    ? 'https://identity.ic0.app'
    : 'http://rdmx6-jaaaa-aaaaa-aaadq-cai.localhost:4943';

function LoginPage() {
  const { login: loginWithSolana, loginStatus, identity: solanaIdentity, clear } = useSiws();
  const { setSolanaIdentity } = useAuth();
  const wallet = useWallet();
```

***Рисунок 3.10*** *Використання авторизації з Solana*

```
{wallet.publicKey && (
  <button
    className='my-5 w-full flex items-center justify-center gap-2 py-3 px-4 rounded-md font-medium primary-button'
    disabled={loginStatus === 'logging-in'}
    onClick={() => {
      loginWithSolana()
      .then(async (delegetaionIdentity) => {
        setSolanaIdentity(delegetaionIdentity as any);
        updateActor().then(() => {
          navigate(ApplicationRoutes.Profile);
        });
      })
      .catch((e) => {
        toastNotifications.error(e.message || 'unexpected error occurred while signing in with solana')
      });
    }}
  >
    {loginStatus === 'logging-in' ? 'Signing in…' : 'Sign in with Solana'}
  </button>
)}
```

***Рисунок 3.11*** *Виклик функції логіну з гаманцем*

### 3.2.5 Фронтенд-частина

Для створення користувацького інтерфейсу я обрав популярний фреймворк ReactJS. Цей інструмент має багато переваг, серед них можна відзначити декларативність, використання віртуального DOM (що сильно оптимізує роботу додатку в браузері) і компонентну структуру, яка гарно підходить для перевикористання однакових елементів на сайті і пришвидшує розробку та масштабованість.

Щоб полегшити написання стилів для компонентів була обрана бібліотека TailwindCSS - CSS-фреймворк з підходом «utility-first», який дозволяє швидко створювати кастомні користувацькі інтерфейси. TailwindCSS дотримується принципу «mobile-first» [19] системи брейкпоінтів, що значно спрощує процес побудови адаптивного дизайну. Використовуючи цей принцип, мені вдалось створити адаптивний веб-інтерфейс, що враховує особливості розмірів різних типів пристроїв.

Важливим також є механізм поєднання фронтенд-каністри з іншими. Це можливо за допомогою спеціальних Candid-інтерфейсів. Candid - це мова

опису інтерфейсів (interface description language), призначена для визначення публічного інтерфейсу сервісу. Згенерувати їх для каністри можна за допомогою інструменту збірки dfx.

storage.did M

storage > storage.did

```
type Result_1 = variant { Ok : User; Err : text };
type User = record {
  name : text;
  description : text;
  created_at : nat;
  image : text;
  principal_id : text;
  location : text;
};
service : () -> {
  create_brand : (Brand) -> (Result);
  create_or_get_user : (text) -> (User);
  get_all_recycle_data : () -> (vec RecycleDataWithoutImage) query;
  get_brand_with_employees : (text) -> (opt record { Brand; vec User }) query;
  get_brands_by_principal : (text) -> (vec Brand) query;
  get_recycle_data : (text) -> (vec RecycleDataWithoutImage) query;
  get_user : (text) -> (opt User) query;
  store_data : (blob, text, text, nat) -> (text);
  update_user : (text, text, text, text) -> (Result_1);
  user_works_for_brand : (text, text) -> (bool) query;
}
```

***Рисунок 3.12*** *Candid-інтерфейс для каністри "storage"*

```
const byteArray = new Uint8Array(arrayBuffer);
const authClient = await AuthClient.create();
const identity = authClient.getIdentity();
const storageActor = createStorageActor(storageCanisterId, {
  agentOptions: {
    identity,
  },
});

await storageActor.store_data(byteArray, formData.comment, formData.location, new Date().getTime());

toastNotifications.success('Recycling data stored successfully!');
```

***Рисунок 3.13*** *Приклад виклику функції каністри "storage"*

### 3.2.6 Інтеграції із зовнішніми сервісами

Для імплементації деяких функцій на платформі були налаштовані інтеграції з API сторонніх сервісів на клієнтській і серверній частинах.

Раніше вже описувався процес валідації зображення на основі «Gemini API». Gemini - це сімейство мультимодальних великих мовних моделей (LLMs), розроблених компанією Google DeepMind. Ця модель є мультимодальною, отже може працювати не лише з текстом, а й з зображеннями, аудіо, відео та кодом. Це дає змогу вирішувати складніші завдання, ніж традиційні текстові LLM. API має безкоштовну і платну версії [20].

Для автоматизації процесу визначення геолокації був використаний сервіс «Opencagedata». Він надає способи геокодування (перетворення координат у адреси або назви місць) по всьому світу на основі відкритих даних через REST API з безкоштовним доступом [21]. Дозвіл на отримання значення широти і довготи надає користувач за бажанням.

Ще одним сервісом, котрий був впроваджений у застосунку, є «DeepAI» - платформа, що пропонує різноманітні інструменти на базі штучного інтелекту, включно з генерацією зображень. Надає REST API для роботи з «Text-to-image» моделлю, котра може створити зображення на основі наданого промпт-запиту. Початковий план для використання сервісу починається від 5 доларів [22].

Після впровадження блокчейну Solana в додаток, з’явилась потреба зберігати метадані і зображення NFT у децентралізованому сховищі і мати доступ до них за URL адресою. Для цього було обрано сервіс «Pinata», котрий допомагає керувати файлами в IPFS (децентралізована система зберігання файлів). IPFS є корисним для Web3-платформ, бо дозволяє

зберігати NFT-метадані та зображення без ризику втрати через централізований сервер [23].

```
export const uploadJsonToPinata = async (metadata: Metadata): Promise<string> => {
  const res = await fetch('https://api.pinata.cloud/pinning/pinJSONToIPFS', {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      pinata_api_key: PINATA_API_KEY,
      pinata_secret_api_key: PINATA_SECRET_API_KEY!,
    },
    body: JSON.stringify(metadata),
  });

  if (!res.ok) {
    throw new Error(`Failed to upload JSON metadata: ${res.statusText}`);
  }

  const data = await res.json();
  return data.IpfsHash;
};

export const getFileUrl = (fileCID: string) => `https://gateway.pinata.cloud/ipfs/${fileCID}`;
```

***Рисунок 3.14*** *Метод для завантаження JSON в "Pinata" і отримання URL-адреси для файлу через збережений хеш*

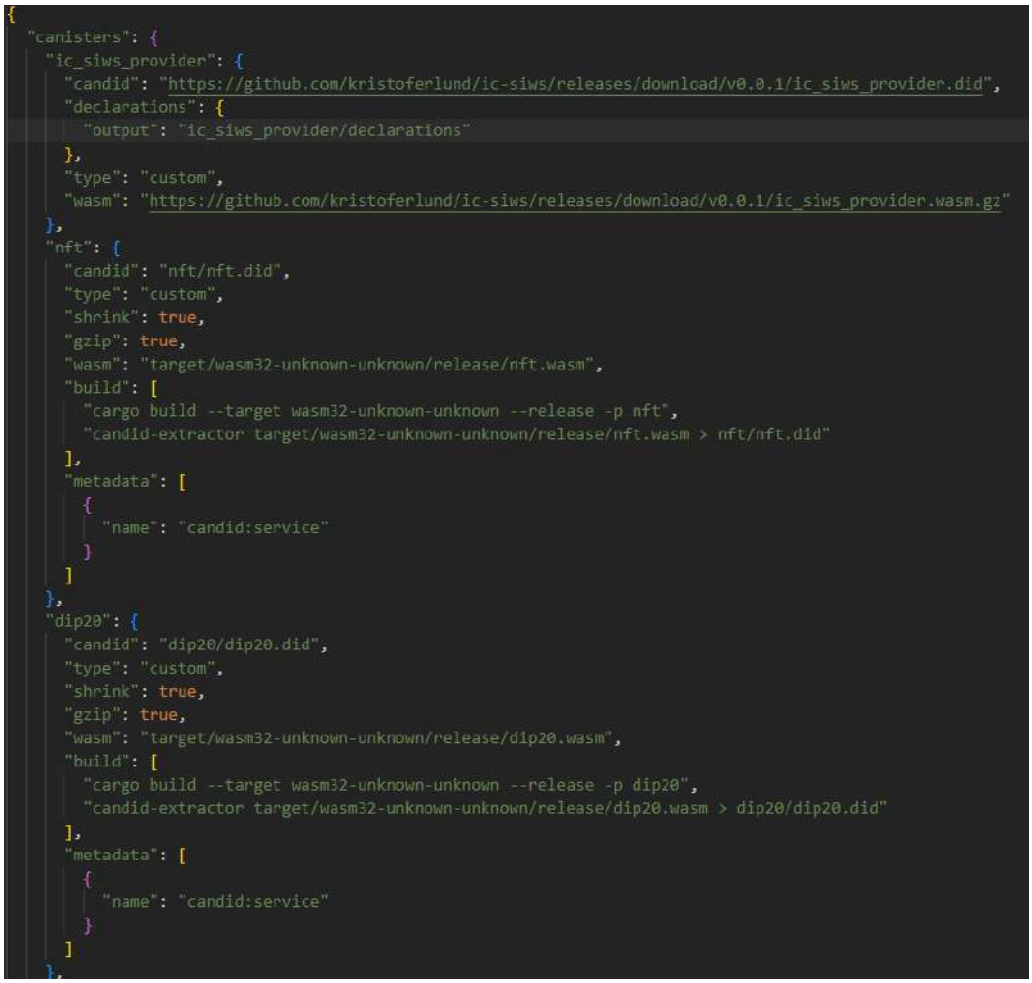
### 3.2.7 Розгортання

Після створення усіх потрібних каністер, вони можуть бути задеплоєні:

- у локальному середовищі, що створене за допомогою dfx (основний інструмент для збірки ICP проєктів);
- у кастомній локальній мережі;
- у mainnet - основна мережа, призначена для продакшн-використання. Розгортання у mainnet потребує циклів (обчислювальних ресурсів).

Для локальної розробки, зручно скористатись першим варіантом. Для того, щоб зібрати програму, потрібно створити файл dfx.json. Файл dfx.json використовується для налаштування параметрів проєкту, зокрема дозволяє вказати перелік каністр у застосунку; їхні назви, типи та вихідні файли;

параметри збірки за замовчуванням і багато інших опцій. По суті, можна порівняти файл dfx.json з файлами docker-compose, котрі так само декларативно визначають список сервісів системи.



```
{
  "canisters": {
    "ic_siws_provider": {
      "candid": "https://github.com/kristoferlund/ic-siws/releases/download/v0.0.1/ic_siws_provider.did",
      "declarations": {
        "output": "ic_siws_provider/declarations"
      },
      "type": "custom",
      "wasm": "https://github.com/kristoferlund/ic-siws/releases/download/v0.0.1/ic_siws_provider.wasm.gz"
    },
    "nft": {
      "candid": "nft/nft.did",
      "type": "custom",
      "shrink": true,
      "gzip": true,
      "wasm": "target/wasm32-unknown-unknown/release/nft.wasm",
      "build": [
        "cargo build --target wasm32-unknown-unknown --release -p nft",
        "candid-extractor target/wasm32-unknown-unknown/release/nft.wasm > nft/nft.did"
      ],
      "metadata": [
        {
          "name": "candid:service"
        }
      ]
    },
    "dip20": {
      "candid": "dip20/dip20.did",
      "type": "custom",
      "shrink": true,
      "gzip": true,
      "wasm": "target/wasm32-unknown-unknown/release/dip20.wasm",
      "build": [
        "cargo build --target wasm32-unknown-unknown --release -p dip20",
        "candid-extractor target/wasm32-unknown-unknown/release/dip20.wasm > dip20/dip20.did"
      ],
      "metadata": [
        {
          "name": "candid:service"
        }
      ]
    },
```

***Рисунок 3.15*** *Файл dfx.json в проєкті*

Відповідно, для деплою каністер потрібно скористатись командою «dfx deploy». Якщо розміщення відбувається в mainnet, то розробник попередньо має поповнити баланс каністри «циклами». Цикли (cycles) - це одиниця обчислювального ресурсу. Вони виконують роль "палива" для каністр, подібно до того, як газ використовується в Solana. Лише за один долар можна придбати близько одного терациклу, що еквівалентно 1,000,000,000,000 циклів. Такої кількості вистачить, аби зберігати 1 гігабайт даних у каністрі протягом року, при цьому виконуючи мільйони викликів у межах платформи [24].

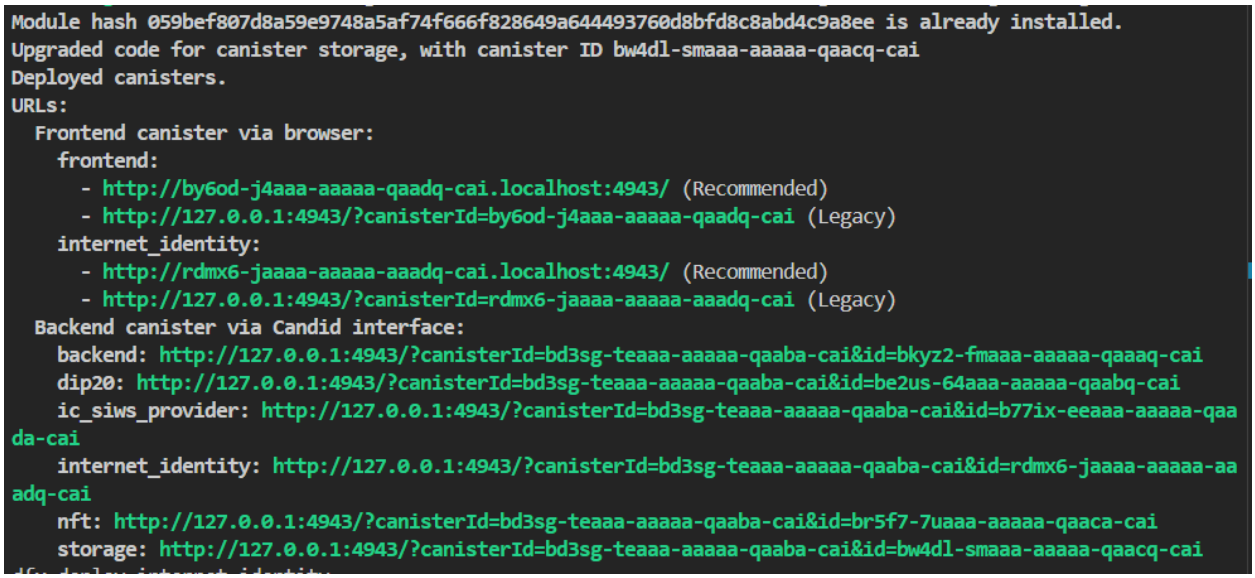



***Рисунок 3.16*** *Приклад розгортання застосунку в локальній мережі dfx*

Після розгортання застосунку в mainnet, він стає публічно доступним і отримує доменну адресу, що сформована на основі ідентифікатору каністри. Для прикладу, моя платформа після деплою отримала адресу «https://2dwgp-naaaa-aaaan-qzynq-cai.icp0.io». Для налаштування власного домену, ICP пропонує скористатись популярними реєстрами доменних адрес, наприклад GoDaddy чи NameCheap.

# РОЗДІЛ 4. Результати роботи

## 4.1 Робота застосунку

Початково користувач має авторизуватись, для цього він має обрати один з двох можливих методів: напряму через Internet Identity, або використовуючи Solana-гаманець. Обираючи Solana wallet, користувач отримуватиме NFT-знижку на обраний Solana-акаунт. При авторизації через Internet Identity відбувається редірект на нову сторінку, цей механізм подібний до OAuth2 авторизації з провайдерами Google, Facebook та іншими. На цій сторінці потрібно обрати або створити новий principal ID.

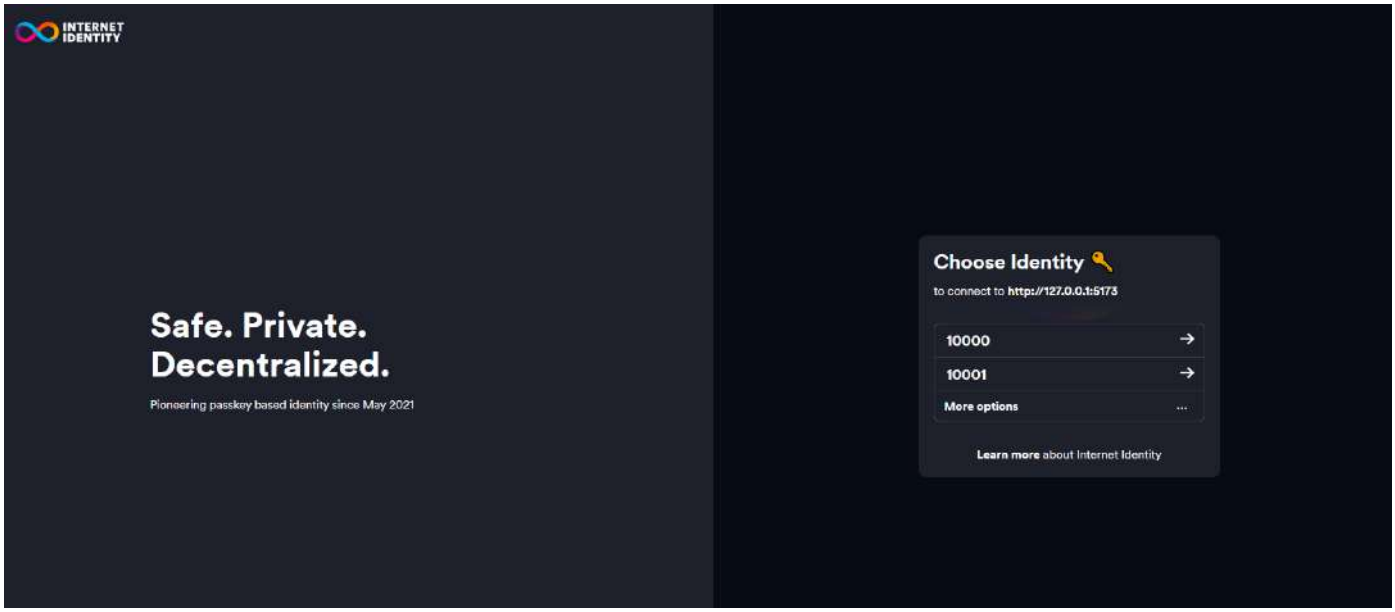



***Рисунок 4.1*** *Авторизація через Internet Identity*

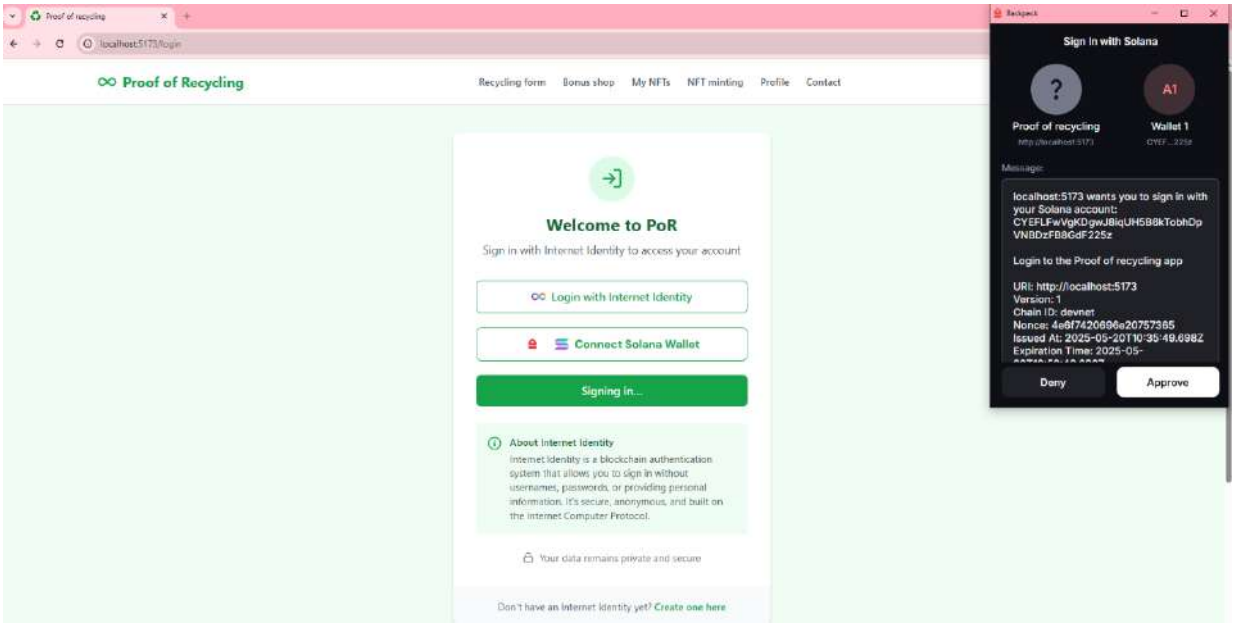



***Рисунок 4.2*** *Сторінка логіну на платформі і авторизації з Solana*

Далі користувач-партнер може створити шаблон NFT-знижки, вказавши назву, опис, фото (яке він може згенерувати також за допомогою штучного інтелекту), категорію і зберегти результати. Для нових купонів ціна визначається автоматично і становить 2500 POR.

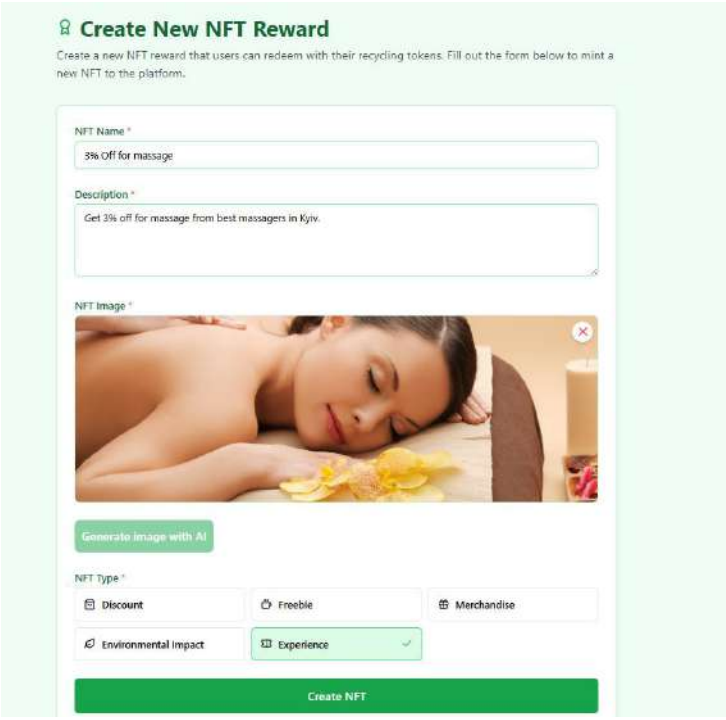



***Рисунок 4.3*** *Форма створення купона на знижку для партнера*

Зі свого боку, користувач, що переробляє сміття, має надати фото-доказ з локацією і описом своєї екологічно корисної активності. Після перевірки зображення, він відразу отримає 1000 POR-токенів.

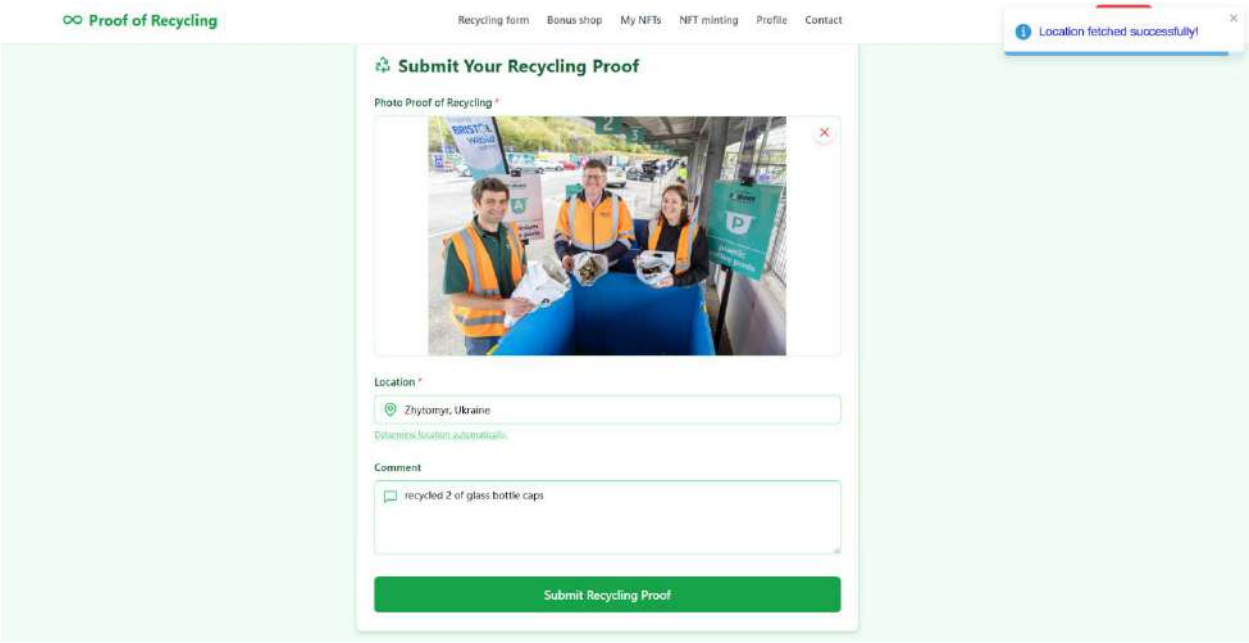



***Рисунок 4.4*** *Форма для переробки сміття*


***Рисунок 4.5*** *Повідомлення після успішного збереження інформації про переробку*

Коли користувач назбирає принаймні 2500 токенів, у нього з’явиться опція придбати знижку в магазині бонусів. Якщо він не має приєднаного Solana-гаманця, то NFT буде відображатись на сторінці NFT-колекції, інакше NFT можна буде побачити і на криптогаманці. Також, звертаємо увагу, що

попередньо створена партнером знижка для масажу з’явилась у списку бонусів. На прикладі здійснюється покупка саме цього купону. На сторінці бонусів наявна функція фільтрації за категоріями.

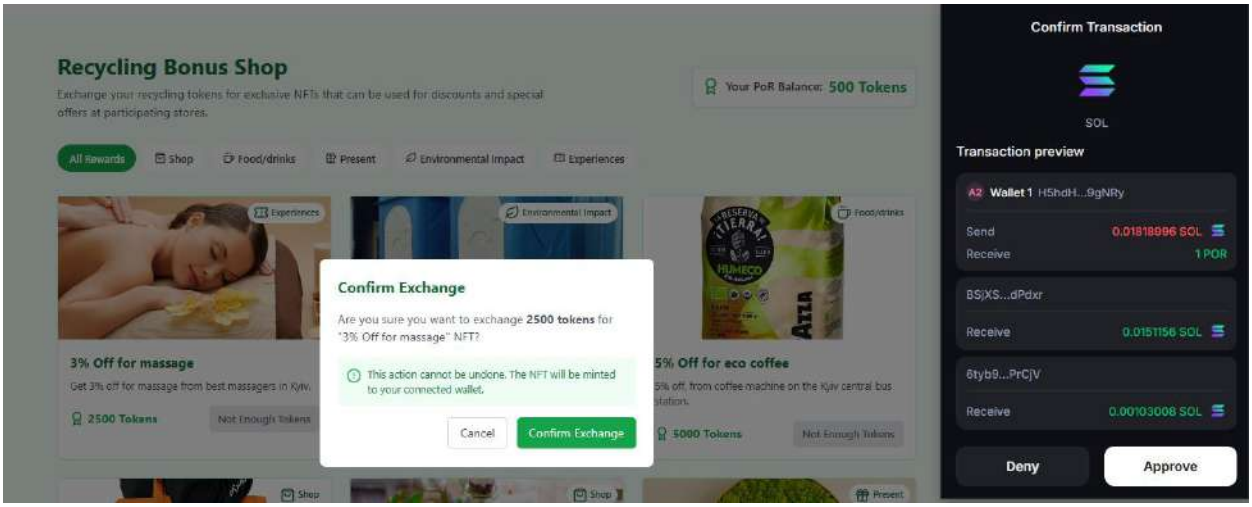



***Рисунок 4.6*** *Покупка знижки за POR-токени*

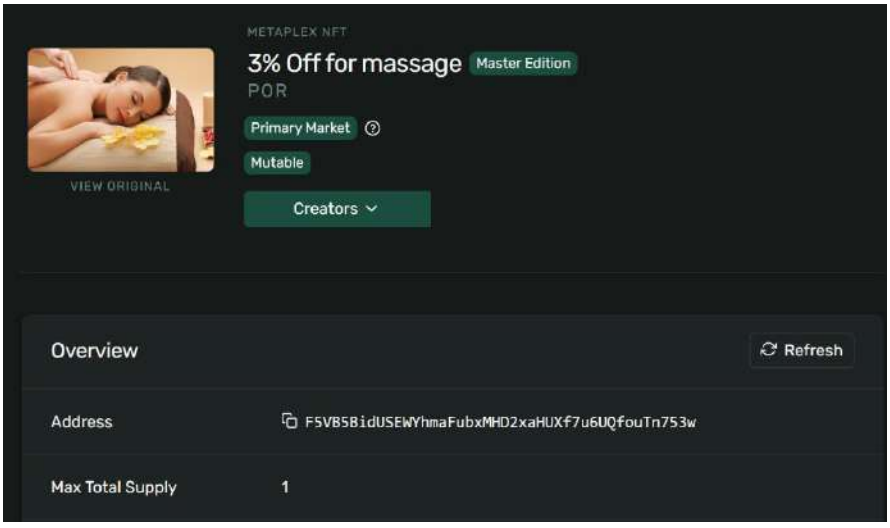



***Рисунок 4.7*** *Вигляд NFT в Solana explorer*

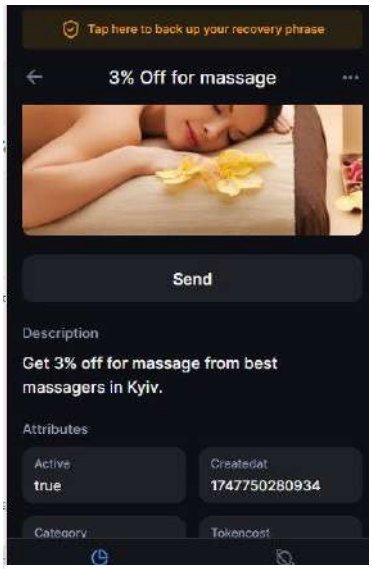



***Рисунок 4.8*** *Вигляд NFT в криптогаманці*

Учасник платформи, що купує знижку з Solana-гаманцем має сплатити комісію за проведення транзакції, однак за аналогічну дію, що здійснюється на ICP, оплата за транзакцію не береться.  Це можливо через «reverse gas model».

Потім, щоб користувач мав змогу скористатися своєю знижкою, він може перейти на сторінку з власною NFT-колекцією і показати QR-код партнеру, який створив шаблон цього купона. Лише він здатний списати знижку. Для синхронного оновлення змін, була використана технологія WebSocket. QR-код генерується за допомогою бібліотеки «react-qr-code».

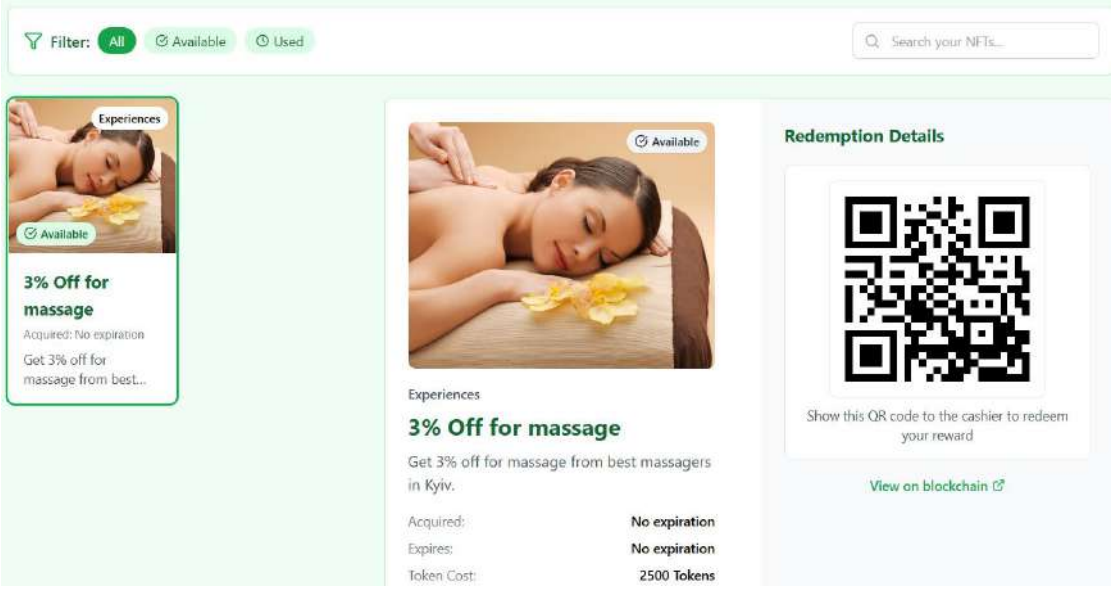



***Рисунок 4.9*** *NFT-колекція знижок перед списанням*

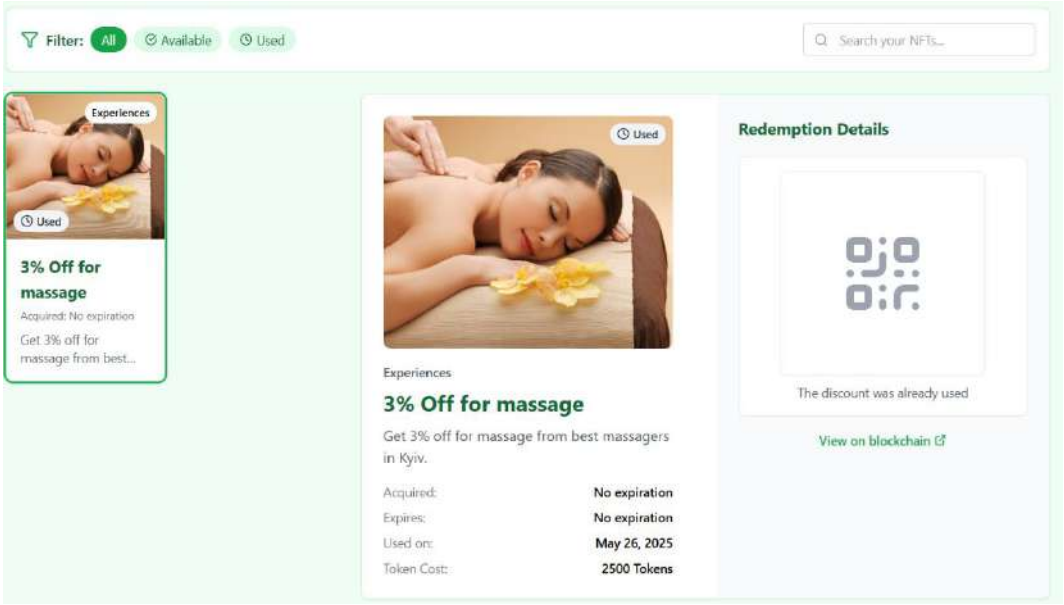



***Рисунок 4.10*** *NFT-колекція знижок після списання*

Крім того, на платформі існує можливість заробляти POR-токени за певну активність на платформі. Наприклад, за те що користувач 5 днів поспіль переробляє відходи, він отримує додаткову винагороду. Валюту також можна отримати за щоденне відвідування застосунку, а також за невелике опитування, що складається з 3 екологічних питань. Чим більше правильних відповідей буде набрано, тим більше токенів учасник може отримати. До того ж, розроблена функціональність рейтингу спільноти, що може бути додатковим способом заохочення учасників вкладати більше зусиль в переробку за рахунок фактору змагальності.

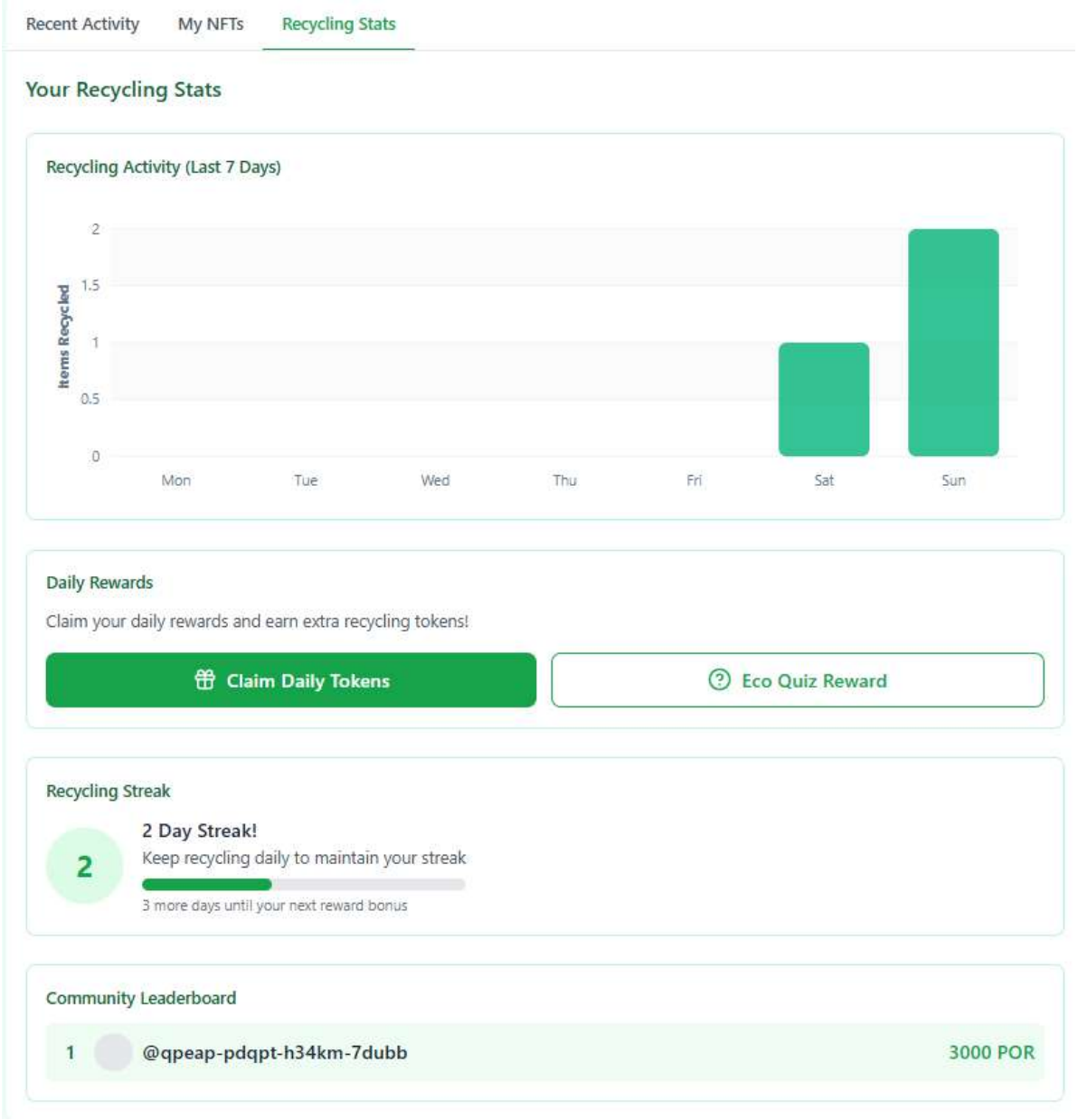



***Рисунок 4.11*** *Статистика користувача і leaderboard*

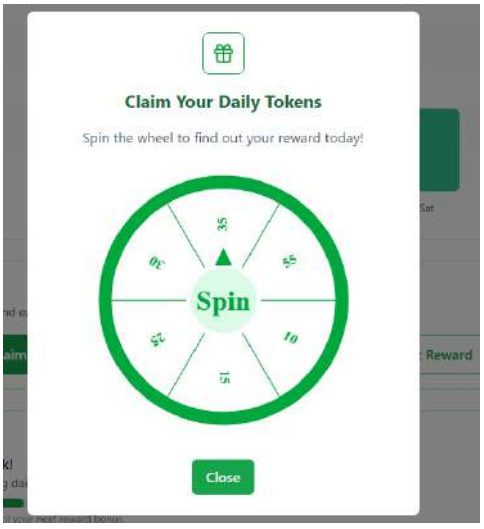



***Рисунок 4.12*** *Колесо щоденного бонусу*

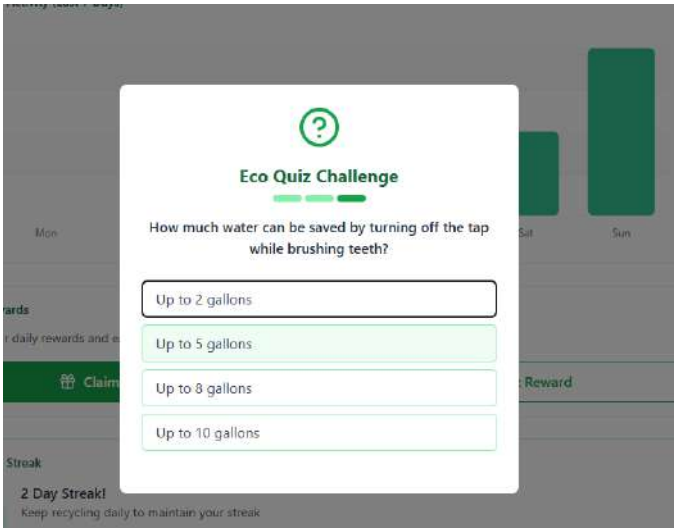



***Рисунок 4.12*** *Проходження щоденного опитування*

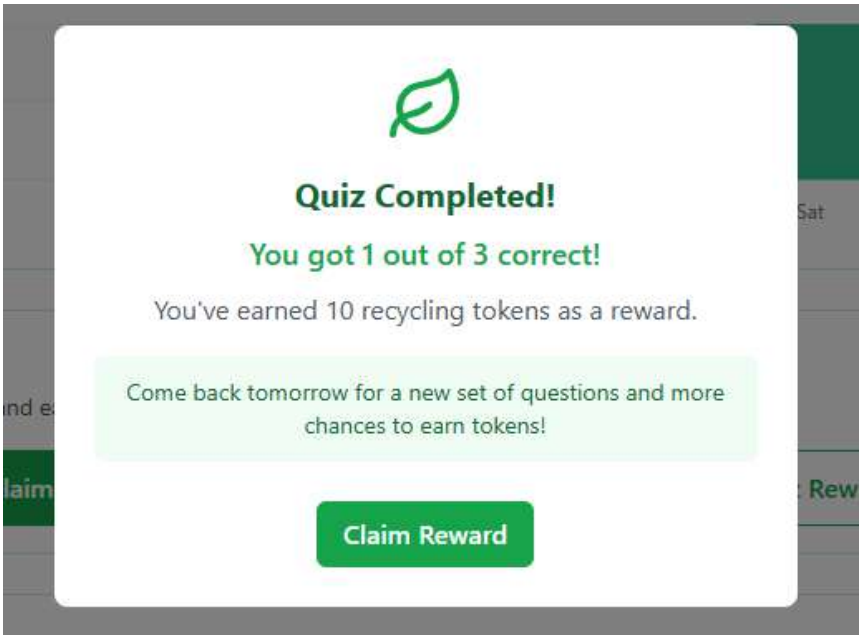



***Рисунок 4.14*** *Результат проходження опитування*

## Висновки

У процесі виконання кваліфікаційної роботи вдалось дослідити серйозність і масштаби екологічних проблем в Україні та світі. Крім того, з'ясовано головні переваги ICP для побудови децентралізованих застосунків, а також проаналізовано відмінності між блокчейнами з різними моделями газу. Був проведений детальний аналіз та надана аргументація використання Web3-архітектури у проєкті.

Як результат, було реалізувано гнучку платформу для мотивації користувачів переробляти відходів. Веб-додаток має адаптивний інтерфейс та розгорнутий і повноцінно працює у блокчейні-мережі ICP. Також, вдалось додати кросчейнову функціональність, побудувавши Solana-програми на Anchor і впровадивши їх у платформу. Було додано декілька способів авторизації, в залежності від вибору та потреб користувача. Логіка отримання винагород написана і працює на обох блокчейнах. Разом з тим, розроблені інтеграції з API зовнішніх сервісів, що значно автоматизують деякі процеси під час роботи в додатку, а також демонструють потужну можливість взаємодії ICP-програм з Web2-сервісами без оракулів. Елементи гейміфікації надають змогу зацікавити користувача у щоденному відвідуванні платформи, а також, допомагають дізнатись більше про проблематику переробки, що існує в нашій державі.

Ідея застосунку вже отримала ствердні відгуки від експертів. Проєкту «Proof of Recycling» вдалось перемогти в хакатоні «Let's try ICP» і зайняти друге місце на «Mohyla startup accelerator». Це означає, що платформа має високий потенціал за рахунок своєї важливої соціальної місії і інноваційному поєднанню децентралізованих технологій.

**Список використаної літератури**

1. Waste recycling market size, share, growth & trend by 2032. *Allied Market Research*. 2024. URL: https://www.alliedmarketresearch.com/waste-recycling-market-A144607.

2. How to get a grant for the development of a recycling enterprise. *Ministry of Economy of Ukraine*. 2023. URL: https://me.gov.ua/news/detail?lang=en-gb&id=0ef5b3cb-692d-4902-a6f1-778ebf89a86e&title=howtogetagrantforthedevelopmentofarecyclingenterprise-&showmenutree=true.

3. Greencubator запускає гранти для зеленних стартапів. *greencubator - Ukrainian energy innovations network*. 2024. URL: https://greencubator.info/mini_grants_for_startups_from_greencubator.

4. Byrnes H. Canada produces the most waste in the world. The US ranks third. *Usa today*. *Ecology*. 12.07.2019. URL: https://www.usatoday.com/story/money/2019/07/12/canada-united-states-worlds-biggest-producers-of-waste/39534923.

5. Vaughan A. China tops WHO list for deadly outdoor air pollution. *the Guardian*. URL: https://www.theguardian.com/environment/2016/sep/27/more-than-million-died-due-air-pollution-china-one-year.

6. Скільки українців готові сортувати сміття. *Active Group - Маркетингові та соціологічні дослідження*. 2021. URL: https://activegroup.com.ua/2021/11/15/skilki-ukraynciv-gotovi-sortuvati-smittya.

7. Сімончук О. Як мобільний додаток «Кліматичні краплі» вчить дітей екoвідповідальності. *«Освіторія» Медіа*. 2019. URL: https://osvitoria.media/experience/yak-mobilnyj-dodatok-klimatychni-krapli-vchyt-ditej-ekovidpovidalnosti.

8. Blockchain and Applications, 4th International Congress (Modelling of the Internet Computer Protocol Architecture: The Next Generation Blockchain; pp. 3-

12) / ed. by J. Prieto et al. Cham : Springer International Publishing, 2023. URL: https://doi.org/10.1007/978-3-031-21229-1.

9. Stable memory | Internet Computer. *Internet Computer docs*. URL: https://internetcomputer.org/docs/motoko/main/stable-memory/stablememory.

10. The oracle problem in smart contracts: data privacy, security, and solutions. *MediaLaws*. URL: https://www.medialaws.eu/rivista/the-oracle-problem-in-smart-contracts-data-privacy-security-and-solutions.

11. Guide to Web Authentication. *WebAuth docs*. URL: https://webauthn.guide/#about-webauthn.

12. Solana Quick Start Guide. Web3 Infrastructure for Everyone. *Solana docs*. URL: https://solana.com/docs/intro/quick-start.

13. Albus J. Solana deep dive: Unpacking proof-of-history. *Blockworks*. 17.03.2025. URL: https://blockworks.co/news/solana-proof-of-history.

14. Application frontends. *Internet Computer docs*. URL: https://internetcomputer.org/docs/building-apps/frontends/using-an-asset-canister.

15. The concept behind canister smart contracts. *DFINITY Foundation*. *Terminology*. 01.03.2025. URL: https://support.dfinity.org/hc/en-us/articles/360057605571-What-are-canister-smart-contracts.

16. Stable structures. *Internet Computer docs*. URL: https://internetcomputer.org/docs/building-apps/developer-tools/cdks/rust/stable-structures.

17. Introduction to Anchor basics with Rust. *Anchor docs*. URL: https://www.anchor-lang.com/docs.

18. Kristofer L. SIWS, Sign in with Solana for ICP, the Internet Computer. Build cross chain Solana apps on ICP. *GitHub*. URL: https://github.com/kristoferlund/ic-siws.

19. Responsive design. Core concepts. *Tailwind CSS docs*. URL: https://tailwindcss.com/docs/responsive-design.

20. Gemini API reference. *Google AI for Developers*. URL: https://ai.google.dev/api?lang=js.

21. Open versus closed data. OpenCage - Easy, Open, Worldwide, Affordable Geocoding and Geosearch. *OpenCage API docs*. URL: https://opencagedata.com/why-use-open-data.

22. Whiles G. Deep AI Review - What Would You Like Help With. *Originality AI Plagiarism and Fact Checker - Publish With Integrity*. 17.09.2024 URL: https://originality.ai/blog/deep-ai-review.

23. Quickstart. *Pinata Docs*. URL: https://docs.pinata.cloud/quickstart.

24. Cycles in terms of the Internet Computer. *DFINITY Foundation*. *Terminology*. URL: https://support.dfinity.org/hc/en-us/articles/360057132632-What-are-cycles-in-terms-of-the-Internet-Computer.