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

РОЗРОБКА СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ ПРОВЕДЕННЯ ЗБОРІВ КОШТІВ В КРИПТОВАЛЮТІ

Текстова частина до кваліфікаційної роботи
за спеціальністю «Інженерія програмного забезпечення»

Керівник кваліфікаційної роботи
старший викладач Гороховський К. С.

_____
(*підпис*)
“____” _____ 2025 р.
Виконав студент
ІПЗ-4 Ігумнов О. О.
“____” _____ 2025 р.

Київ – 2025

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем

ЗАТВЕРДЖУЮ

Завідувач кафедри мультимедійних систем

доцент, кандидат наук

О. П. Жежерун

“____” _____ 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Ігумнову Олександру Олександровичу

факультету інформатики 4-го курсу бакалаврської програми

ТЕМА: Розробка соціальної мережі для проведення зборів коштів в криптовалюті

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1. Аналіз предметної області та загальні теоретичні відомості

2. Архітектура та технічні особливості застосунку

3. Розробка застосунку та оцінка отриманих результатів

Висновки

Список використаних джерел

Дата видачі “____” _____ 2025 р. Керівник _____

Завдання отримав _____

Тема: Розробка соціальної мережі для проведення зборів коштів в криптовалюті

Календарний план виконання роботи:

№ п/п	Назва етапу	Термін виконання	Примітка
1.	Вибір теми та визначення завдання курсової роботи	19.09.2024	
2.	Аналіз предметної області	01.10.2024	
3.	Знайомство з документацією засобів розробки веб-застосунків	Жовтень-Листопад 2025	
3.	Проєктування архітектури соціальної мережі	Грудень 2025	
4.	Розробка веб-застосунку	Січень-Квітень 2025	
5.	Написання текстової частини роботи	Травень 2025	
6.	Підготовка до презентації, написання доповіді	17.05.2025-18.05.2025	
7.	Захист кваліфікаційної роботи	Червень 2025	

Студент _____

Керівник _____ “_____” _____ 2025 р.

ЗМІСТ

АНОТАЦІЯ 6
ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЗАГАЛЬНІ ТЕОРЕТИЧНІ ВІДОМОСТІ 11
1.1. Еволюція веб-технологій від Web 1.0 до Web 3.0 11
1.2. Основні характеристики та принципи роботи Web 3.0 13
1.3. Проблематика волонтерських фандрайзингових платформ та доцільність використання web3-технологій для їх розробки 14
1.4. Аналіз існуючих соціальних мереж для зборів коштів у волонтерських цілях 16
1.5. Опис технології блокчейн та методологія її застосування при розробці web3-застосунків 18
1.6. Аналіз технологій для інтеграції криптоплатежів та авторизації через криптогаманці 20
1.7. Висновки до розділу 1 23
РОЗДІЛ 2. АРХІТЕКТУРА ТА ТЕХНІЧНІ ОСОБЛИВОСТІ ЗАСТОСУНКУ .25
2.1. Постановка задачі та технічні вимоги до системи 25
2.2. Технологічний стек та аргументація обраних інструментів розробки 28
2.3. Архітектурна модель веб-застосунку 33
2.4. Висновки до розділу 2 36
РОЗДІЛ 3. РОЗРОБКА ЗАСТОСУНКУ ТА ОЦІНКА ОТРИМАНИХ РЕЗУЛЬТАТІВ 37
3.1. Реалізація функціональних компонентів застосунку 37
3.1.1. Реалізація програми на Solana 37
3.1.2. Реалізація серверної частини 38
3.1.3. Реалізація клієнтської частини 41
3.2. Тестування системи 48
3.3. Підготовка до розгортання 49

3.4. Аналіз отриманих результатів та їх відповідності поставленим вимогам.....52
3.5. Порівняння з традиційними централізованими рішеннями.....53
3.6. Перспективи подальшого розвитку.....54
3.7. Висновки до розділу 3.....56
ВИСНОВКИ.....57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....58

АНОТАЦІЯ

Кваліфікаційна робота спрямована на вдосконалення веб-застосуноку для збору коштів на волонтерських засадах шляхом його інтеграції з технологіями Web3.

Додаток доповнений оновленим функціоналом, а саме: криптовалютними платежами, авторизацією через програмні криптовалютні гаманці такі як Phantom, Solflare, Backpack та інші, зберіганням інформації про користувачів на блокчейні у вигляді Soulbound NFT. Додано функціонал фільтрації користувацького контенту на основі ШІ. Значно підвищено продуктивність та рівень захищеності серверів, де розгорнуто цільовий веб-застосунок.

Front-end частина додатку реалізована з використанням мов програмування JavaScript/TypeScript, фреймворків TailwindCSS та Next.js, а також open-source бібліотеки іконок Lucide-React. Back-end написаний на фреймворку NestJS з використанням СКБД PostgreSQL та Prisma ORM. Додатково використовувалися Git, CI/CD-платформа GitHub Actions, система керування ізольованими Linux-контейнерами Docker, веб-сервер Nginx. Для зберігання та роботи з файлами обрано BaaS платформу Supabase. Інтегровано платіжну бібліотеку Stripe.js, API ШІ-платформи Gemini, а також Google та Discord APIs для авторизації через OAuth2. Застосунок працює з блокчейном Solana, що передбачає інтеграцію відповідних web3-бібліотек у веб-частині застосунку, а також використання мови програмування Rust для написання програми на блокчейні. Для розгортання веб-застосунку обрано хмарну платформу DigitalOcean.

Ключові слова: кваліфікаційна робота, веб-застосунок, соціальна мережа, волонтерські збори, блокчейн, криптовалюта, криптогаманець, Solana, Phantom, Solflare, Backpack, web3, Solana, смарт-контракт, Rust, TypeScript, Next.js, NestJS, Docker, Git, GitHub Actions, Supabase, Nginx, Docker, PostgreSQL, Prisma ORM, Visual Studio Code, Google API, Discord API, OAuth2, Stripe.js, Gemini, DigitalOcean.

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

API – application programming interface

BaaS – backend as a service

CI/CD – continuous integration/continuous delivery

CSR – client side rendering

CSS – cascading style sheets

CVC – card verification code

DeFi – decentralized finances

DEX – decentralized exchange

DTO – data transfer object

DVCS – distributed version control system

HTTP(S) – hypertext transfer protocol (secure)

IP – internet protocol

JSON – JavaScript object notation

NFT – non-fungible token

ORM – object-relational mapping

SEO – search engine optimization

SPL – Solana program library

SQL – structured query language

SSG – static site generation

SSH – secure shell

SSR – server-side rendering

TLS – transport layer security

URL – uniform resource locator

БД – база даних

СКБД – система керування базами даних

ШІ – штучний інтелект

ВСТУП

В сучасному світі технології розвиваються шаленими темпами, визначаючи значний прогрес, що суттєво впливає на різні аспекти життя людства. Веб-технології не є виключенням.

Ще не так давно сайти несли суто інформаційний характер, передбачалося, що взаємодія користувачів з Інтернет-ресурсами обмежуватиметься лише переглядом відображуваного контенту. Кількість інструментів розробки для створення веб-ресурсів була невелика, на той час, певно, не було потреби у їх широкому різноманітті, а залучення людського ресурсу, фінансові та часові затрати були мізерними, у порівнянні з сьогоденням. Прийнято вважати це першим етапом Інтернету, який ще часто називають Web 1.0.

Однак відсутність інтерактивності при активному збільшенні зацікавленості користувачів мережею, стрімкий розвиток хмарних та веб-технологій, зростання кількості мобільних пристроїв та прагнення бізнесу до чогось більшого сприяло появі нової ери – Web 2.0, в якій людство перебуває і досі. Сайти перетворилися у веб-застосунки, їх архітектурна складність зросла в рази, різноманітність інструментів розробки стала значно ширша. Активно почали розвиватися соціальні мережі, зросла залученість користувачів до веб-ресурсів. Проте даний етап сприяв централізації Інтернету. Великі корпорації отримали величезний контроль над інформацією, що отримують користувачі. В свою чергу, через це з’явилися і серйозні проблеми з конфіденційністю, оскільки згадані корпорації активно збирають персональні дані із різних джерел.

Тим часом, паралельно активно розвивається технологія блокчейн – децентралізований реєстр даних, які зберігаються у вигляді послідовності блоків, що з’єднані між собою за допомогою криптографії. Переваги цієї технології та зростання її популярності сприяли до формування нового етапу – Web 3.0, що нівелює усі недоліки Web 2.0, а також створює широке різноманіття нових

можливостей, таких як написання смарт-контрактів та взаємодія з ними, інтеграція криптогаманців у веб-застосунки та криптовалютні перекази між анонімними блокчейн-адресами.

На основі наведених вище відомостей можна зробити висновок, що наразі ми знаходимося на етапі переходу від Web 2.0 до Web 3.0, тому постає питання забезпечення відповідності існуючих веб-застосунків сучасним вимогам.

Метою роботи є удосконалення соціальної мережі для проведення зборів коштів на волонтерських засадах шляхом інтеграції технологій, що відповідають принципам Web 3.0.

З урахуванням поставленої мети роботи визначено наступні **завдання дослідження**:

- проаналізувати стан розвитку веб-технологій та особливості переходу від Web 2.0 до Web 3.0;
- дослідити основні принципи блокчейн-технологій та визначити способи їх застосування у сфері веб-розробки;
- визначити блокчейн або перелік блокчейнів, з якими працюватиме додаток;
- розробити архітектурне рішення для удосконалення вже існуючого веб-застосунку шляхом інтеграції web3-функціоналу, а саме: авторизації з використанням криптогаманця, криптовалютних платежів та зберігання інформації про користувачів у вигляді Soulbound NFT;
- реалізувати функціонал фільтрації користувацького контенту з використанням ШІ;
- налаштувати сервери веб-застосунку з урахуванням усіх потенційних безпекових проблем та здійснити його розгортання.

Об’єктом дослідження є соціальні мережі у форматі Web 2.0 веб-застосунків.

Предметом дослідження є удосконалення існуючого Web 2.0 веб-застосунку соціальної мережі для збору коштів на волонтерських засадах шляхом інтеграції web3-функціональності.

Актуальність обраної теми зумовлена стрімким розвитком веб-технологій та зростаючою популярністю децентралізованих рішень, що підвищують рівень безпеки додатків та забезпечують достатній рівень прозорості і конфіденційності для їх користувачів. Застосунки, побудовані з використанням web3-функціоналу стають дедалі популярнішими, а також демонструють стабільнішу роботу та надають більше інтерактивних можливостей цільовій аудиторії, ніж традиційні Web 2.0 додатки. Останнім часом можна спостерігати значний зріст зацікавленості цифровими активами пересічними людьми та навіть провідними державами. Стає очевидною необхідність інтеграції web3-функціоналу, такого як підключення та робота з криптогаманцями, в сучасних веб-застосунках, зокрема, якщо вони передбачають платіжні операції. Криптовалютні транзакції працюють значно швидше та є більш прозорими у порівнянні з загальновідомою централізованою обробкою фінансових платежів. Особливо актуальним це є у сфері волонтерської діяльності, де прозорість і доступність фінансових операцій є критично важливими. Саме тому інтеграція web3-функціоналу в соціальну мережу для проведення зборів коштів на волонтерських засадах є актуальною, як мінімум користувачі матимуть можливість альтернативного способу створення фінансових пожертв.

Робота складається з трьох розділів.

У першому розділі викладено інформацію про предметну область. Розглянуто основні принципи блокчейн-технологій та їх застосування у сфері децентралізованих веб-застосунків.

У другому розділі описано архітектурні та технічні особливості удосконаленого веб-застосунку, обґрунтовано вибір інструментів та технологій.

У третьому розділі надано опис процесу розробки, наведено досягнуті результати та визначено перспективи подальшого розвитку.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ЗАГАЛЬНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Еволюція веб-технологій від Web 1.0 до Web 3.0

Перша версія всесвітньої павутини була розроблена британським науковцем Тімом Бернерсом-Лі у 1989 році. Даний етап розвитку мережі Інтернет прийнято називати Web 1.0. Він передбачав взаємодію користувачів із веб-сторінками лише у вигляді перегляду статичного контенту, який був створений розробниками конкретного веб-ресурсу. Будь-яка інтерактивність між користувачами та сторінками була здебільшого відсутня. Незважаючи на вузький спектр можливостей для аудиторії Web 1.0, все ж варто відзначити переваги даної версії:

- швидкість доступу до інформації через статичність веб-сторінок;
- мінімальні технічні вимоги для розробки сайтів;
- високий рівень захищеності користувачів, зумовлений обмеженою інтерактивністю;
- простота розгортання.

Наступним етапом веб-еволюції є Web 2.0, що пропонує користувачам більш інтерактивний та колаборативний досвід, а також сприймається як мережа для читання та запису, оскільки учасники мають можливість взаємодіяти з контентом у різні способи: створення, редагування, видалення, коментування, оцінювання, обмін тощо. Для даного етапу характерні веб-застосунки, які передбачають перелічений вище функціонал і наочним прикладом таких додатків є соціальні мережі. Web 2.0 повністю централізований, адже значна кількість популярних ресурсів належить великим компаніям, таким як Google, Meta, Amazon, Apple, Microsoft та ін., що вважається однією з його негативних властивостей на ряду з низьким рівнем безпеки та захисту даних користувачів, а також проблемами з продуктивністю і доступністю веб-застосунків. Наразі можна констатувати, що

людство перебуває на поточному етапі розвитку веб-технологій, оскільки більшій кількості існуючих веб-застосунків притаманні Web 2.0 характеристики, однак можна зробити припущення, що це фінальна стадія даного етапу, оскільки останнім часом помітне стрімке зростання кількості Web 3.0 додатків.

Web 3.0 вважається новою ерою розвитку мережі Інтернет. Його основна суть полягає у децентралізації, де учасники матимуть можливість самостійно контролювати власні дані та ідентичність, взаємодіючи з блокчейн-технологіями та протоколами консенсусу. Таким чином підвищується безпека та конфіденційність користувачів, забезпечується нижча залежність від традиційних централізованих серверів, робота системи на основі смарт-контрактів, що, в свою чергу, підвищуватиме рівень прозорості. Крім того, створюються додаткові можливості для економічної взаємодії шляхом використання криптовалют, токенізації активів та використання NFT. Даний етап розвитку веб-технологій активно починає заміняти Web 2.0, даючи розуміння того, що невдовзі він стане основним, а отже постає необхідність удосконалення існуючих веб-застосунків з метою забезпечення відповідності стандартам Web 3.0.

Наведене нижче зображення чітко демонструє основні відмінності між різними етапами розвитку веб-технологій (рис. 1.1.1) [3].

Type	**Web1.0**	**Web2.0**	**Web3.0**
Transfer of files	Read only	Read & Edit	Read & Edit & Execute
Type of networks	Simple network	Social network	Semantic network
User capacity	Million	Billion	Trillion
Websites	Static	Dynamic	Intelligent
AI	Non-applicable	Non-applicable	Applicable
Contents	Experts-designed	Social network-based	Personalized information flow
Search engines	Domain names	Search engine optimized	AI-based
Centralization	Centralized	Centralized	Distributed
Purpose	Connect information	Connect users	Connect knowledge

Рис. 1.1.1. Порівняння етапів розвитку веб-технологій

1.2. Основні характеристики та принципи роботи Web 3.0

Чітко визначених принципів Web 3.0 не існує, однак можна зробити логічне припущення, виходячи з наявної інформації, яка є у відкритому доступі, що дана екосистема формується на основі наступних принципів технології блокчейн:

1) Децентралізація – принцип, за яким відбувається розподіл контролю та керування системою між усіма учасниками певної мережі, уникаючи централізованого контролю даних та ресурсів конкретним користувачем або організацією. Зростаюча централізація Інтернету, за словами Тіма Бернерса-Лі, розробника World Wide Web, «призвела до того, що люди, які розробили цю платформу, без жодних свідомих дій, створили широкомасштабний емерджентний феномен, який є антилюдським». Дійсно, наразі в мережі Інтернет помітне значне домінування великих технологічних компаній таких як Google, Amazon, Meta та ін., які сконцентрували весь контроль над конфіденційністю та правом власності користувачів. Контент, який створюють люди в мережі Інтернет, наразі не є стовідсотково ними контрольований, оскільки згадані вище компанії мають можливість маніпулювати ним у своїх цілях. Звісно, правила обробки даних регулюються умовами надання послуг, які користувачі не зобов’язані, але часто змушені приймати задля користування певними сервісами, однак дані сервіси можуть у будь-який момент змінювати ці умови на свій розсуд. Децентралізація повністю вирішує дану проблему шляхом розподілу права власності між розробниками та користувачами, усуваючи присутність посередників між ними [4].
2) Довіра – принцип блокчейн-технологій, що дозволяє учасникам мережі довіряти програмам, які вони використовують. У централізованій мережі, як вже було згадано раніше, користувачі погоджуються на умови користування сервісом, однак ці умови можуть згодом бути змінені у

зовсім неприйнятний для особи спосіб, спричинюючи дискомфорт або навіть ряд небажаних проблем. Однак такі блокчейни як Solana або Ethereum мають функціонал смарт-контрактів, код яких після публікації неможливо змінити, і поки в мережі існує один вузол, вона продовжуватиме працювати вічно за будь-яких обставин.

3) Прозорість – принцип, що полягає у наданні повного доступу до інформації про транзакції та операції в децентралізованих мережах. Будь-який учасник мережі має можливість переглянути результати виконання транзакції без необхідності довіри посередникам, в тому числі у вигляді централізованих організацій. Усі дані та операції зберігаються в публічному та незмінному реєстрі. Кожен користувач може зробити аудит виконання умов смартконтракту, що підвищує рівень довіри до системи. Ще однією позитивною характеристикою є те, що більшість децентралізованих застосунків мають відкритий код, що дає можливість його перевірити та визначитись безпекою його використання.

Усі децентралізовані застосунки мають будуватися на основі описаних вище принципів, щоб створити якісну Web3 екосистему, в якій прозорість, безпека, конфіденційність та довіра користувача будуть мати найвищий пріоритет.

1.3. Проблематика волонтерських фандрайзингових платформ та доцільність використання web3-технологій для їх розробки

Наразі існує чимала кількість фандрейзингових платформ, які збирають кошти на волонтерських засадах з різною метою. Усі вони, принаймні найбільш відомі, переважно є централізованими web2-застосунками, а отже мають низку недоліків:

- *низький рівень прозорості*: користувачі зазвичай не мають можливості повністю перевірити інформацію про автора волонтерської кампанії,

результати усіх його попередніх кампаній, а якщо така можливість і є, то проблематично відслідкувати розподіл зібраних коштів навіть попри потенційно наявну звітність, оскільки її цілком можливо сфальсифікувати;

- *низький рівень довіри*: даний недолік можна вважати наслідком викладеної вище інформації. Користувачі, які, можливо, і хотіли б зробити фінансовий внесок, стикаються з ситуацією, коли безпека переказу коштів ставиться під сумнів. У випадку шахрайських дій зі сторони людини, яка проводить збір, кошти будуть використані для задоволення власних потреб, однак також існує ймовірність їх переказу для вчинення незаконних дій або підтримки терористичних організацій;
- *комісії та участь посередницьких організацій при переказах*: часто користувачі даних платформ зіштовхуються з проблемою високих комісій, які зумовлені різними чинниками, такими як: законодавства різних країн (зокрема податкові), посередницькі організації, що беруть відсоток від переказу (банки та інші фінансові установи), конвертації валют тощо;
- *високий рівень технічних неполадок*: оскільки дані сервіси є централізованими, то існує доволі висока ймовірність виникнення збоїв у роботі, які можуть бути не пов’язані безпосередньо з проблемами в програмному коді застосунку, а є наслідками технічних несправностей зі сторони хмарних платформ, які надають послуги з розгортання цих застосунків;
- *тотальний контроль*: адміністратори даних платформ у будь-який момент можуть маніпулювати контентом, розміщеним користувачами цих застосунків, на свій розсуд. Вони мають повний контроль над усіма життєвими процесами програми і їм ніщо не може завадити вносити зміни без попередження про це користувачів. У свою чергу, власники хмарних сервісів, на яких розміщені платформи для збору коштів, можуть їх контролювати, зокрема обмежувати до них доступ на невизначений термін

без пояснень. Можна зробити висновок, що звичайні користувачі в даному ланцюжку не є рівноцінними учасниками мережі та користуються наданими послугами на свій страх та ризик.

Web 3.0 технології нівелюють усі вищеописані недоліки, оскільки ґрунтуються на ідеї децентралізації. Це значно підвищує рівень прозорості та довіри користувачів до застосунків, якими вони користуються. В свою чергу, користувачі стають рівноправними учасниками мережі на рівні з розробниками продукту, а також зникає необхідність посередництва при виконанні транзакцій. Окрім цього, багато сучасних блокчейнів мають дуже низькі комісії, а також доволі високу кількість транзакцій на хвилину, що є значною перевагою над традиційними способами виконання фінансових операцій. Крім цього, відсутні будь-які обмеження для виконання грошового переказу, а особа, що надсилає залишається конфіденційною, що, в окремих випадках, є дуже важливою умовою.

Загалом, можна впевнено констатувати факт, що покращення існуючих веб-застосунків шляхом інтеграції web3-технологій стане правильним рішенням, як мінімум точно не зробить вже існуючий додаток гіршим. Таким чином вдасться підвищити рівень зацікавленості та задоволення користувачів, а без їх залученості описані сервіси не матимуть сенсу свого існування. Дані удосконалення дозволять значно переважати над конкурентами та розширять спектр можливостей для майбутнього розвитку.

1.4. Аналіз існуючих соціальних мереж для зборів коштів у волонтерських цілях

На етапі планування розробленого застосунку для волонтерських зборів постало питання визначити особливість, яка б вирізняла його від вже існуючих ресурсів з подібною тематикою. Його характерною рисою було визначено поєднання властивостей притаманних соціальним мережам та фандрайзинговим

платформам в одне ціле, з метою залучення більшої користувацької активності та концентрацією уваги на тематиці волонтерських зборів.

Проаналізувавши значну кількість мобільних та веб-продуктів, застосунків, що поєднують в собі вказані вище особливості, не було знайдено. Більша частина з них є або виключно фандрайзинговими платформами без функціоналу, що притаманний соціальним мережам (GoFundMe, Монобанка), або звичайними соціальними мережами, які, в своїй більшості, не передбачають платіжних та інших інструментів для ефективної організації волонтерських зборів коштів (Instagram, Facebook).

У зв’язку з виконаним аналізом існуючих платформ-аналогів, можна зробити висновок, що гібрид соціальної мережі та фандрайзингової платформи, об’єднаний в єдиний застосунок, виводить користувацький досвід на абсолютно новий рівень та є досить актуальним на сьогоднішній день, особливо, враховуючи геополітичні події сучасного світу та потреби нашої держави і її народу.

Враховуючи сучасну тенденцію активного розвитку web3-технологій та стрімкий ріст зацікавленості все більшої кількості людей до криптовалют та технології блокчейн в цілому, було вирішено додатково інтегрувати web3-функціонал для залучення більшої аудиторії та відкриття ширшого спектру можливостей для подальшого розвитку. Можливість криптовалютних платежів, авторизації через криптогаманці, взаємодія з блокчейном та інтеграція ШІ-технологій забезпечують відповідність застосунку сучасним стандартам веб-розробки, підвищують його інтерактивність та рівень зацікавленості користувачів, надають його цільовій аудиторії ширше різноманіття можливостей для взаємодії.

Отже, виходячи з наведених вище результатів аналізу потенційних платформ-аналогів та сучасних течій розвитку веб-технологій можна зробити висновок, що застосунок даного типу однозначно є актуальним та має надзвичайний потенціал для подальшого розвитку.

1.5. Опис технології блокчейн та методологія її застосування при розробці web3-застосунків

Блокчейн – це децентралізований цифровий незмінний реєстр даних, що дозволяє реєструвати транзакції та відстежувати користувацькі активи в межах цієї мережі та надає єдине джерело істини. Він працює як децентралізована база даних, дані в якій зберігаються одразу на декількох комп'ютерах (учасниках мережі блокчейн), що робить її стійкою до підробки. Усі транзакції на блокчейні підтверджуються за допомогою механізму консенсусу, який забезпечує згоду в мережі.

Структура блокчейну являє собою ланцюжок блоків, де кожен блок містить набір транзакцій та усі блоки зв'язані завдяки наявності в собі хешу попереднього блоку. В свою чергу, хеш блоку вираховується на основі декількох параметрів, таких як: хеш попереднього блоку, дані транзакцій, час створення блоку, нонс та інші метадані, проте ці параметри можуть різнитися для різних блокчейнів. Дані транзакцій також хешуються, їх хеші формують меркл корінь транзакцій, що також являє собою хеш вирахуваний за допомогою спеціальної структури – дерева Меркла. Меркл корінь транзакцій також включається в хеш блоку та відображає захешовані дані транзакцій. Така структура гарантує цілісність даних та захищає від несанкціонованого доступу, що робить блокчейн ідеальним для застосунків, які працюють з криптовалютами, документообігом, виборчими процесами тощо.

Варто окремо виділити одну з найважливіших деталей будь-якого блокчейну – механізм консенсусу. Вузли в мережі перевіряють і підтримують блокчейн, підтверджуючи дійсність кожної транзакції за допомогою алгоритмів консенсусу, забезпечуючи безпеку і незмінність системи. Proof of Work (PoW) і Proof of Stake (PoS) - одні з найпоширеніших алгоритмів консенсусу, кожен з яких допомагає захистити систему при перевірці транзакцій. Наприклад, в блокчейні Bitcoin використовується алгоритм Proof of Work (PoW), який працює за наступною

логікою: в мережі є майнери – учасники мережі, які змагаються за можливість додати новий блок до ланцюжку, мотивацією для цього є блокова винагорода (певна кількість біткоїнів). Право на додавання нового блоку отримує той майнер, який вирішить певну криптографічну задачу швидше за його конкурентів. У випадку, коли декілька майнерів отримують право додати блок в ланцюжок, найдовша гілка вважається єдиним джерелом істини. Однак такий механізм має низку недоліків, оскільки, у випадку малої кількості майнерів, влада концентрується у того, чиї обчислювальні потужності найбільші. В такому випадку, наприклад, є ймовірність, що шахраї зможуть «просунути» свою версію блокчейну, зробивши деякі з останніх блоків недійсними. Зловмисники також можуть об'єднати свої потужності і, у випадку, якщо їх частка займатиме більше 50% від загальних потужностей, блокчейн стає вразливий до атаки відомої як «атака 51%». Тому найсприятливішою умовою для Proof of Work блокчейнів є велика кількість незалежних майнерів. Задля вирішення подібних проблем було придумано низку інших механізмів консенсусу, зокрема Proof of Stake, який є основою для багатьох вже існуючих сучасних децентралізованих мереж. Його ідея полягає у відмові від енергозатратних обчислень, основна увага приділяється кількості заморожених монет учасників мережі. Той, хто має найбільшу частку, має вищу ймовірність отримати право на додавання нового блоку. При цьому, кожен учасник обмежений в шахрайських діях, оскільки ризикує втратити всю свою частку шляхом конфіскації. Також блокчейни, базовані на Proof of Stake, значно швидше обробляють транзакції за рахунок відмови відмови від довготривалих та енергозатратних обчислень. Тому Proof of Stake вважається найпопулярнішим механізмом консенсусу на даний час [1].

Основна перевага блокчейну полягає в його спроможності забезпечити прозорість, довіру та захист даних, не покладаючись на традиційних посередників у вигляді банків або інших третіх осіб. Його архітектура значно знижує рівень шахрайства та помилок, що є особливо цінним в фінансових та медичних галузях, де безпека транзакцій є критично важливою.

Технологія блокчейн ідеально підходить для інтеграції у веб-застосунки. Існує безліч ідей як можна використати дану систему при розробці веб-платформ на різну тематику. У вже існуючі додатки, які працюють з грошовими операціями, активно додається функціонал обробки криптоплатежів, також з'явилась можливість реалізовувати авторизацію через криптогаманці шляхом підпису спеціального повідомлення. Блокчейн чудово підходить для проєктів, пов'язаних з колаборативною розробкою документів, систем голосувань, ігрових платформ тощо.

Існує велика кількість вже готових рішень для інтеграції web3-авторизації, створення NFT, переказу криптовалюти між різними блокчейнами та багато іншого функціоналу. Доступне велике різноманіття бібліотек з відкритим кодом для різних мов програмування, які можна використовувати для створення власних web3-додатків. Зокрема, для веб-застосунків, які передбачають роботу з криптогаманцями існують зручні інструменти, такі як wallet adapters, що дозволяють зручно, швидко та безпечно працювати з користувацькими гаманцями без прямого доступу до них. За потреби реалізації власних рішень, для взаємодії з блокчейнами, їх власники, зазвичай, надають бібліотеки, що містять мінімальний необхідний функціонал для зручної роботи. Наприклад, розробники Solana пропонують TypeScript/JavaScript бібліотеки @solana/web3.js та @solana/spl-token, в той час як для Ethereum існує бібліотека ethers.

1.6. Аналіз технологій для інтеграції криптоплатежів та авторизації через криптогаманці

При розробці проєкту для взаємодії було обрано блокчейн Solana. Він має ряд переваг над іншими децентралізованими реєстрами, такі як висока пропускна здатність обробки транзакцій, дуже низькі та фіксовані комісії, хороша

масштабованість, велике ком'юніті та орієнтація на масову інтеграцію web3-технологій.

Для реалізації авторизації через криптогаманці існує декілька вже існуючих рішень, а також це досить просто зробити самостійно. Зважаючи на те, що при розробці застосунку основними інструментами розробки серед стеку технологій обрано Next.js та NestJS, серед готових рішень можна розглянути такі бібліотеки як Web3Auth або Privy.io. Для інтеграції достатньо лише створити обліковий запис на відповідних платформах, налаштувати новий проєкт, скопіювати та зберегти відповідні API ключі у власному застосунку, завантажити TypeScript/JavaScript бібліотеки, надані розробниками цих платформ, та дописати частину фронтенд коду для відображення наданих бібліотеками форм авторизації. Проте можна зробити і власну реалізацію процесу авторизації через криптогаманці. Для цього потрібно інсталювати бібліотеки, які надають функціонал wallet-адаптерів, з метою забезпечення взаємодії користувачів із застосунком шляхом підключення гаманців. Також потрібно прописати логіку формування повідомлення для підпису на стороні бекенду, та окремо для перевірки підписаного користувачем повідомлення. У разі успішної валідації формувати нову пару JWT-токенів, session cookie або виконати інші дії, в залежності від обраного методу авторизації користувачів. Для реалізації описаної логіки достатньо буде використати наступні бібліотеки: @solana/wallet-adapter-base, @solana/wallet-adapter-react, @solana/wallet-adapter-react-ui, @solana/wallet-adapter-wallets, @solana/web3.js для фронтенд частини застосунку, а також @solana/web3.js, bs58, tweetnacl (для валідації підписаного повідомлення) зі сторони бекенду [9; 15].

Щоб інтегрувати базовий функціонал обробки криптоплатежів достатньо використати звичайну transfer інструкцію з System Program, однак цей підхід працюватиме лише для переказів native токенів, у випадку блокчейну Solana – SOL. Для того, щоб була можливість робити трансфери в інших токенах необхідно попередньо виконати ряд дій. Спочатку треба переконатись, чи існують associated

token accounts для особи, що надсилає токени, та для особи, що їх має отримати, оскільки, якщо в когось з них буде відсутній такий акаунт, то трансфер буде неможливий. Далі в транзакцію потрібно передати інструкцію transfer, але вже для Associated Token Account Program. Даний підхід дозволяє пересилати будь-який SPL Token, проте для кожного з них потрібно окремо прописувати логіку створення токен акаунтів та трансферу. Найкращим варіантом є інтеграція Jupiter – DEX агрегатора на основі Solana, який дозволяє обміняти один токен на будь-який інший. Jupiter будує шляхи (routes), які складаються з одного або декількох обмінів токенів на будь-яких доступних децентралізованих біржах. Це відкриває можливість для користувачів робити оплату будь-яким наявним у них токеном, оскільки він гарантовано буде обміняний на той, який очікує веб-застосунок. До того ж, Jupiter виконує обміни за максимально вигідною стратегією завдяки їх унікальному алгоритму Jupiter Smart Routing, тому навіть при наявній можливості прямого обміну токенів, якщо буде знайдено шлях, який пропонує обмін за кращою ціною, Jupiter вибере саме його (рис. 1.6.1) [11].

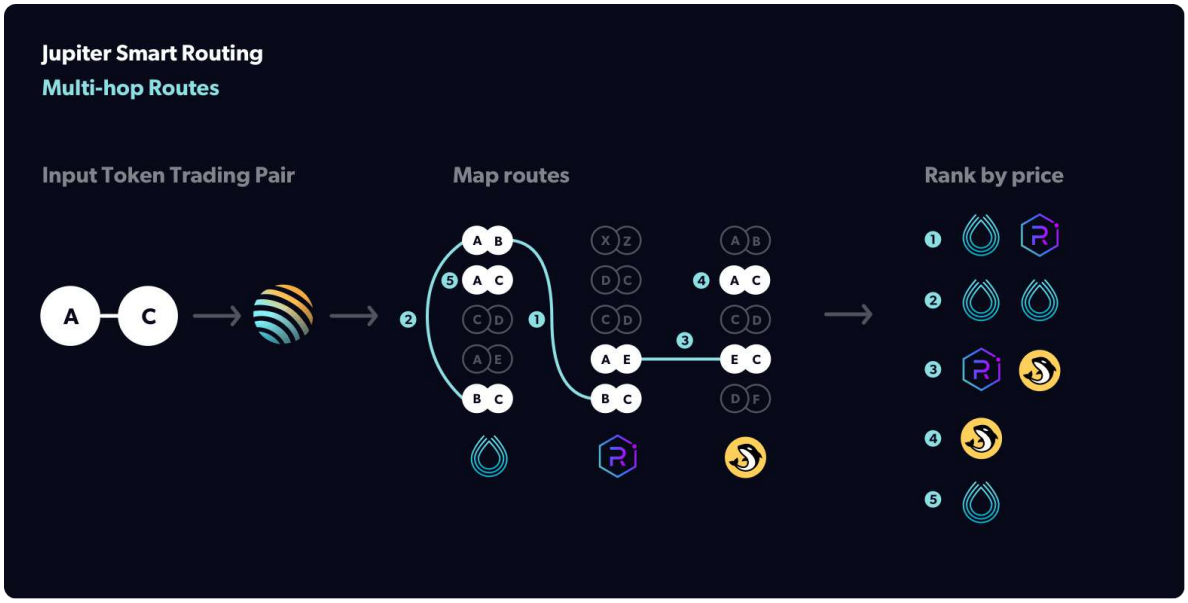

Рис. 1.6.1. Як працює Jupiter Smart Routing

Описані методи та технології розробки значно спрощують та пришвидшують написання коду, а також мають значну кількість переваг над іншими підходами до

розробки подібних застосунків. Авторизація через криптогаманець може бути реалізованою в будь-який з описаних способів, адже кожен з них має свої переваги та недоліки. Наприклад, використання готових рішень пришвидшує розробку, але обмежує розробника в додаванні своїх правок до процесу авторизації, в той час як власна реалізація може бути більш оптимізованою та надавати більше контролю над процесом. Інтеграцію криптоплатежів все ж краще реалізовувати через Jupiter, оскільки він надає найбільш гнучкий та оптимізований спосіб обміну токенів на блокчейні Solana.

1.7. Висновки до розділу 1

В даному розділі було надано основні теоретичні відомості про етапи розвитку веб-технологій, описана основна ідея та принципи функціонування сучасних web3-застосунків, їх переваги над традиційними веб-додатками. Розглянуто проблематику платформ для волонтерських зборів та доцільність використання web3-технологій при їх розробці чи вдосконаленні вже існуючих проєктів.

Проаналізовано існуючі соціальні мережі для фінансових зборів на волонтерських засадах, які потенційно могли б розглядатися як конкуренти цільовому застосунку. Визначено їх переваги та недоліки, обґрунтовано ідею інтеграції web3-технологій для вдосконалення проєкту, залучення більшої кількості аудиторії, підвищення зацікавленості користувачів до платформи та їх впевненості в безпеці, прозорості та стабільності роботи застосунку.

Розглянуто сутність поняття «блокчейн», описано основний принцип його функціонування, механізми консенсусу, їх різновиди, переваги, недоліки та які завдання вирішують. Аргументовано доцільність використання даної технології у сучасних веб-застосунках.

Виконано аналіз технологій для інтеграції обробки платежів у криптовалюті та авторизації через користувацькі гаманці. Розглянуто можливі рішення, як вже існуючі, так і ті, що можна втілити самостійно, для реалізації даного функціоналу. Визначено переваги та недоліки вказаних підходів.

РОЗДІЛ 2. АРХІТЕКТУРА ТА ТЕХНІЧНІ ОСОБЛИВОСТІ ЗАСТОСУНКУ

2.1. Постановка задачі та технічні вимоги до системи

Основним завданням є розробка соціальної мережі для проведення зборів коштів на волонтерських засадах з інтеграцією web3-технологій. Фінальний застосунок повинен мати базовий функціонал, притаманний соціальним мережам, декілька різних способів авторизації, обробку криптоплатежів, запис даних на блокчейн та фільтрацію контенту на основі.

Виходячи з поставленого завдання було визначено наступні функціональні вимоги:

- *авторизація та аутентифікація:* користувачам має бути доступно декілька способів авторизації на вибір. Доступні методи авторизації: через email та пароль, Google OAuth2, Discord OAuth2, з використанням Solana криптогаманця (Phantom, Backpack, Solflare та інші). Система також повинна мати декілька ролей користувачів з різними правами доступу. Права доступу визначаються на основі permission value, яке визначається через побітову операцію «&» для кожного дозволу. Значення 1 – є дозвіл на виконання певної дії, 0 – доступу немає. Основні 3 ролі користувачів: звичайний користувач, волонтер та адміністратор;
- *керування акаунтом:* будь-який користувач повинен мати можливість змінювати персональні дані, зображення або пароль на сторінці профілю. Окрім цього, повинен бути наявний функціонал для підвищення ролі користувача зі звичайного до волонтера;
- *керування кампаніями для збору коштів:* для користувачів з роллю «волонтер» має бути наявний функціонал для керування кампаніями для збору коштів, а саме можливість їх створення, редагування, видалення, зміна видимості, перегляд детальної інформації. В свою чергу, для усіх

користувачів платформи повинна бути доступна можливість взаємодії з даними кампаніями шляхом виставлення реакцій, написання коментарів та створення грошових внесків. Для кожного допису потрібно надати можливість додавання основного зображення, а також вкладених файлів;

- *керування коментарями:* будь-який користувач повинен мати змогу створювати, редагувати та видаляти коментарі під будь-яким дописом для збору грошових пожертв. Також має бути наявним функціонал взаємодії з іншими коментарями шляхом надання відповідей або виставлення реакцій. Для кожного коментаря потрібно надати можливість прикріплення вкладених файлів;
- *взаємодія між користувачами:* має відбуватися шляхом наявності функціоналу «слідкування» за іншими користувачами. Кожен учасник платформи повинен мати змогу підписатися на інших учасників, переглядати слідкувачів та тих, за ким слідкує сам;
- *обробка користувацьких платежів:* має бути доступно 2 варіанти створення грошового внеску: через сервіс Stripe для фіатних платежів, для криптоплатежів має бути використано DEX-агрегатор на блокчейні Solana – Jupiter, оскільки він будує найоптимальніші шляхи для обміну токенів, а також дає можливість користувачам робити внески будь-якими токенами;
- *взаємодія з блокчейном:* повинна відбуватись через @solana/web3.js та @solana/spl-token бібліотеки, можливо також використання бібліотек від Metaplex Foundation для зручної роботи з SPL токенами. Має бути окремо реалізована та розгорнута програма на блокчейні Solana з функціоналом для карбування NFT та створення дата-акаунтів для кожного користувача системи. Необхідно налаштувати взаємодію фронтенд/бекенд частин застосунку з Solana програмою;

- *фільтрування контенту:* реалізувати функціонал автоматичного виявлення та фільтрування на основі ШІ образливого, неприйнятного контенту або такого, що містить спам.

Також визначено наступні нефункціональні вимоги для застосунку:

- *швидкодія:* застосунок повинен швидко відповідати на запити користувачів. Проаналізувавши функціональні вимоги, можна визначити, що орієнтований час відгуку серверу повинен бути не довше за 2-3 секунди. За можливості необхідно додати кешування даних з метою пришвидшення роботи застосунку;
- *масштабованість:* застосунок має бути спроєктований таким чином, щоб додавання нового функціоналу в майбутньому було швидким та безперешкодним, а також не могло б зашкодити вже працюючій версії платформи та не повпливало б на її загальну продуктивність, рівень безпеки тощо;
- *безпека:* застосунок повинен бути розгорнутий на серверах з добре налаштованим рівнем безпеки, зокрема, має бути налаштоване HTTPS-з'єднання, TLS шифрування, сервіс fail2ban для змоги фільтрації та блокування підозрілих IP-адрес, доступ до серверів має бути наявним лише за наявності SSH ключа;
- *цілісність даних:* в бекенд частині застосунку має бути реалізований функціонал автоматичного та періодичного збереження стану бази даних шляхом створення бекапів;
- *якісний UI/UX та адаптація під різні пристрої:* користувацький інтерфейс повинен бути привабливим, простим та інтуїтивно зрозумілим. Він не повинен навантажувати відвідувачів застосунку занадто широким різноманіттям кольорів або великою кількістю інформації. Додаток повинен бути адаптований під пристрої з різною розмірністю дисплеїв.

Розподіл ролей застосунку:

1) *адміністратор* – може виконувати будь-які дії в застосунку, зокрема, здійснювати блокування інших користувачів, а також керувати будь-якими дописами, коментарями, слідкуваннями, особистою інформацією інших користувачів;
2) *волонтер* – може виконувати будь-які дії в застосунку, окрім тих, які доступні лише адміністратору, а саме: здійснення блокувань інших користувачів та керування їх даними;
3) *звичайний користувач* – може виконувати будь-які дії в застосунку, окрім тих, що доступні для ролей «адміністратор» та «волонтер», а також не має можливості керувати кампаніями для збору коштів на волонтерських засадах.

2.2. Технологічний стек та аргументація обраних інструментів розробки

Для реалізації обраного веб-застосунку було обрано наступний набір інструментів розробки:

Solana – Proof of Stake (PoS) блокчейн з механізмом Proof of History (PoH). Виріняється високою пропускною здатністю виконання транзакцій, дуже низькими комісіями та великим ком’юніті, що робить даний блокчейн чудовим вибором для інтеграції у web3-додатки.

Мови програмування JavaScript/TypeScript – більша частина коду була написана саме на TypeScript, оскільки це типізована версія JavaScript, яка дозволяє писати код значно надійніше. TypeScript має статичну типізацію, тому дозволяє знаходити помилки ще на етапі компіляції коду, а не під час його виконання. Також він надає велику кількість utility-типів, які надзвичайно зручні та корисні для використання в різних частинах застосунку. JavaScript використовувався дуже рідко,

лише в окремих випадках, в той час як TypeScript застосовувався як в бекенд частині, та і на фронтенді.

Мова програмування Rust – використовувалася для написання програми на блокчейні Solana. Безальтернативний вибір, оскільки усі програми на Solana пишуться виключно на Rust. Попри свою складність, ця мова програмування має лише переваги над іншими мовами з точки зору швидкості виконання, безпечної роботи з пам’яттю та надійності роботи. Rust також має відмінну підтримку паралелізму, зручний пакетний менеджер cargo та якісну документацію. Ідеальний вибір для написання надійних програм на блокчейні [13].

NestJS – фреймворк на основі Node.js, який було використано для написання бекенд частини застосунку. Він має модульну архітектуру, використання якої полегшує розбиття проєкту на логічні частини. Це забезпечує значно легше керування, масштабування та ізолювання функцій або їх наборів. Ще однією особливістю даного фреймворку є вбудований DI-контейнер, який сприяє зручній інтеграції одних сервісів в інші, а також гарантує перевикористання вже існуючих об’єктів цих сервісів без повторного створення. NestJS добре підходить для швидкого та комфортного написання коду [6].

Next.js – фронтенд фреймворк, надбудова над бібліотекою React, який нівелює більшу частину недоліків React. Особливістю фреймворку є підтримка різних стратегій відмальовки: SSR (Server Side Rendering), CSR (Client Side Rendering), SSG (Static Site Generation), кожна з яких використовується для різних цілей. Також Next.js підтримує streaming механізм, який забезпечує рендеринг даних частинами для оптимізації роботи застосунку. Окрім цього, наявний інший функціонал у вигляді особливої системи навігації, можливість пре-фетчингу даних, middlewares тощо. Також Next.js має хорошу підтримку SEO, що теж відіграє значну роль при виборі даного фреймфорку.

TailwindCSS – CSS-фреймворк з відкритим вихідним кодом. Він не надає ряд попередньо визначених компонентів та стилів для них, як, наприклад, Boostrap.

Замість цього він створює список спеціальних класів, які можна використовувати для стилізації кожного елемента, шляхом їх компонування. Даний фреймворк значно прискорює процес розробки, оскільки повністю прибирає потребу в написанні CSS-стилів.

PostgreSQL – потужна об'єктно-реляційна система керування базами даних з відкритим вихідним кодом, яка вже впродовж довгого часу активно розвивається і завоювала міцну репутацію завдяки надійності, функціональності та продуктивності. Розробники можуть використовувати PostgreSQL для своїх додатків без витрат на ліцензування та з повною гнучкістю у розгортанні, оскільки ця база даних працює на широкому спектрі хмарних і нехмарних платформ. Спільнота розробників робить значний внесок у вдосконалення та підтримку даної СКБД, підвищуючи її безпеку та надійність. Postgres також пропонує широкий спектр вбудованого функціоналу, який зазвичай відсутній в інших СКБД, для полегшення роботи з даними.

Prisma ORM – це інструмент нового покоління, призначений для спрощення взаємодії з базами даних для додатків на Node.js і TypeScript. Він пропонує сучасний підхід до управління базами даних, що робить його привабливим вибором для розробників, які прагнуть ефективності та надійності. Даний інструмент було обрано через нативну підтримку TypeScript, а також досить зручний, у порівнянні з конкурентами (Sequelize, TypeORM, Drizzle), набір функцій для роботи з БД [16].

Supabase – альтернатива Firebase, backend-as-a-service (BaaS) платформа з відкритим вихідним кодом, яка надає розробникам потужну, масштабовану базу даних PostgreSQL, можливості роботи в реальному часі, автентифікацію, зберігання файлів та безсерверні функції. Дозволяє розробникам швидко створювати повностекові додатки без управління інфраструктурою, при цьому зберігаючи гнучкість. При розробці соціальної мережі використовувалась лише як сховище для зберігання файлів.

Jupiter – це платформа децентралізованих фінансів (DeFi) на блокчейні Solana, яка функціонує як агрегатор ліквідності, впорядковуючи та оптимізуючи торговельну діяльність у просторі DeFi. Вона об'єднує ресурси з різних джерел, щоб сприяти більш плавному та ефективному обміну токенів, безстрокових ф'ючерсів та децентралізованих стейблкоінів. Токен JUP слугує токеном управління, що дозволяє власникам брати участь у процесах прийняття рішень, які спрямовують розвиток і роботу платформи. Jupiter використано як інструмент для інтеграції обробки криптоплатежів, оскільки надає для цього потужний функціонал, зокрема, використовує унікальний Jupiter Smart Routing алгоритм.

Git – на сьогоднішній день найпоширеніша у світі сучасна система контролю версій. Це активно підтримуваний проєкт з відкритим вихідним кодом, розроблений у 2005 році Лінусом Торвальдсом, відомим творцем ядра операційної системи Linux. Маючи розподілену архітектуру, Git є прикладом розподіленої системи контролю версій (DVCS). Замість того, щоб мати лише одне єдине місце для повної історії версій програмного забезпечення, робоча копія коду кожного розробника також є сховищем, яке може містити повну історію всіх змін. Значна кількість програмних проєктів покладається на Git для контролю версій, включаючи комерційні проєкти та ті, що мають відкритий вихідний код [20].

Docker – це відкрита платформа для розробки, доставки та запуску застосунків. Він надає можливість пакувати та запускати програми у вільно ізольованому середовищі, яке називається контейнером. Ізоляція та безпека дозволяють запускати багато контейнерів одночасно на одному хості. Контейнери легкі і містять все необхідне для запуску програми, тому немає потреби покладатися на те, що встановлено на хості. Можна надавати спільний доступ до контейнерів під час роботи і бути впевненими, що будь-хто інший, хто має доступ, отримає такий самий контейнер, який працює так само [19].

GitHub Actions – CI/CD платформа, розроблена GitHub, яка дозволяє автоматизувати робочі процеси в межах GitHub репозиторію. За допомогою неї

можна прописувати воркфлоу для запуску тестів, розгортання додатків на серверах або хмарних платформах, а також автоматизувати будь-які завдання життєвого циклу розробки за допомогою конфігураційних файлів YAML. Завдяки підтримці матричних збірок, подій (пул реквестів, пушів у визначені гілки тощо) та багаторазових дій з GitHub Marketplace, GitHub Actions допомагає впорядкувати розробку та забезпечити узгодженість автоматизованих процесів. У розробленому застосунку GitHub Actions використовувався для виконання тестових кейсів, а також підключення та автоматичного розгортання застосунку на віддаленому сервері.

Nginx – це веб-сервер з відкритим вихідним кодом і reverse proxy сервер, який зазвичай використовується для обслуговування статичного контенту, маршрутизації запитів до бекенд-додатків та балансування трафіку. Серед його переваг можна виділити високу продуктивність, масштабованість та ефективність роботи. Nginx може виступати в якості шлюзу між клієнтами і внутрішніми серверами, підвищуючи продуктивність і безпеку, при цьому пропонує гнучку конфігурацію для маршрутизації та кешування.

Gemini API – надає доступ до набору ШІ-моделей та функціоналу для роботи з цими моделями. Використовується в розробленому застосунку як інструмент для фільтрації образливого контенту або того, що може містити спам. Ідеально підходить для описаної задачі, оскільки має спеціально розроблені під це безкоштовні моделі та демонструє кращі результати за аналогічні сервіси (OpenAI, Microsoft Copilot тощо) [8].

DigitalOcean – постачальник хмарної інфраструктури, який пропонує масштабовані обчислювальні ресурси та послуги. Надає віртуальні машини, які часто називають дроплетами (droplets), керовані бази даних, кластери kubernetes, рішення для зберігання даних та мережеві інструменти, що дозволяють користувачам швидко та зручно розгортати застосунки та здійснювати ефективне керування ними. Було обрано саме цю платформу, оскільки Digital Ocean пропонує кращі плани, якщо порівнювати з Google Cloud Platform або Amazon Web Services.

2.3. Архітектурна модель веб-застосунку

З метою безперешкодної, швидкої та якісної розробки було вирішено перед початком роботи виконати проєктування архітектури бази даних, а також усієї системи загалом. В результаті отримано наступну візуалізацію схеми БД (рис. 2.3.1, 2.3.2):

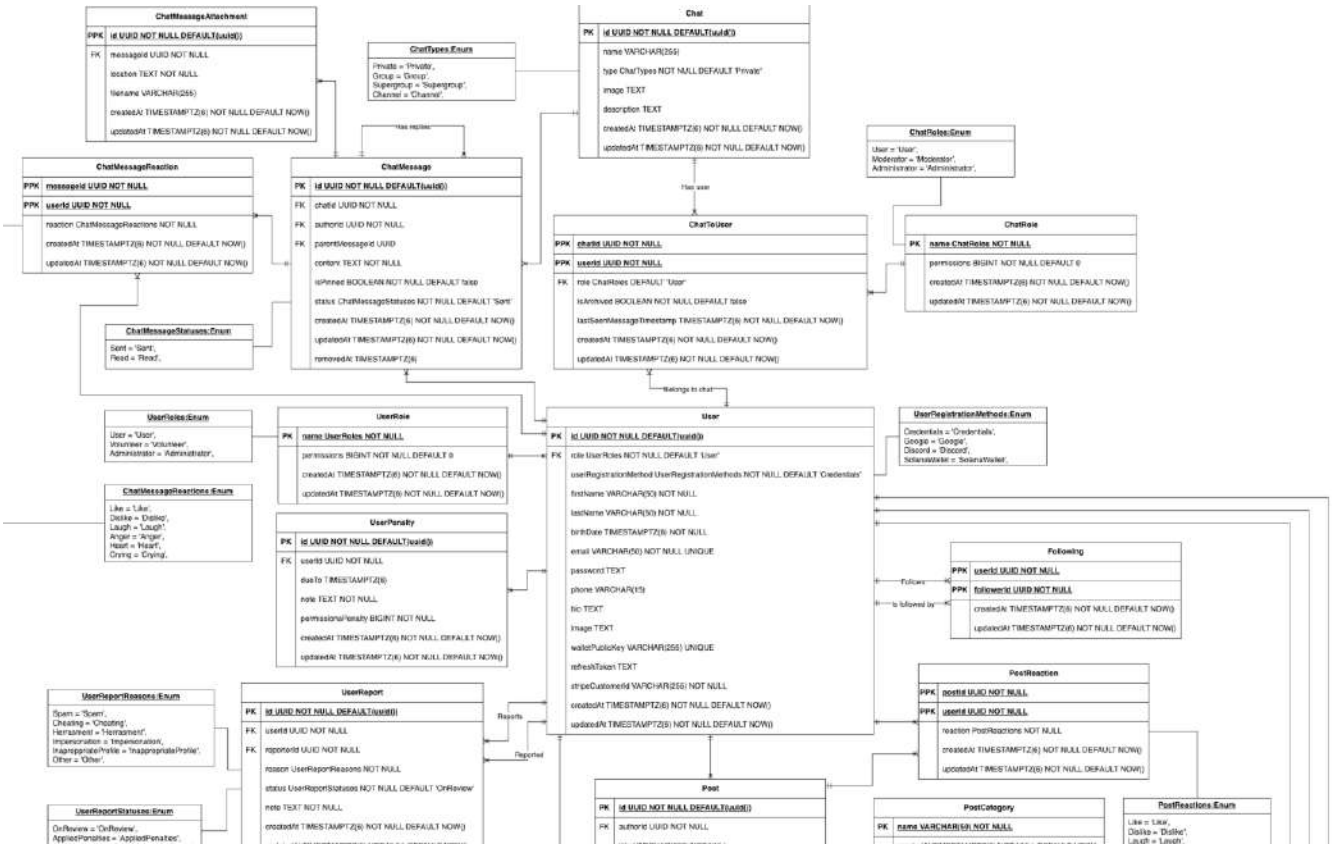

Рис. 2.3.1. Архітектура бази даних застосунку (1 частина)

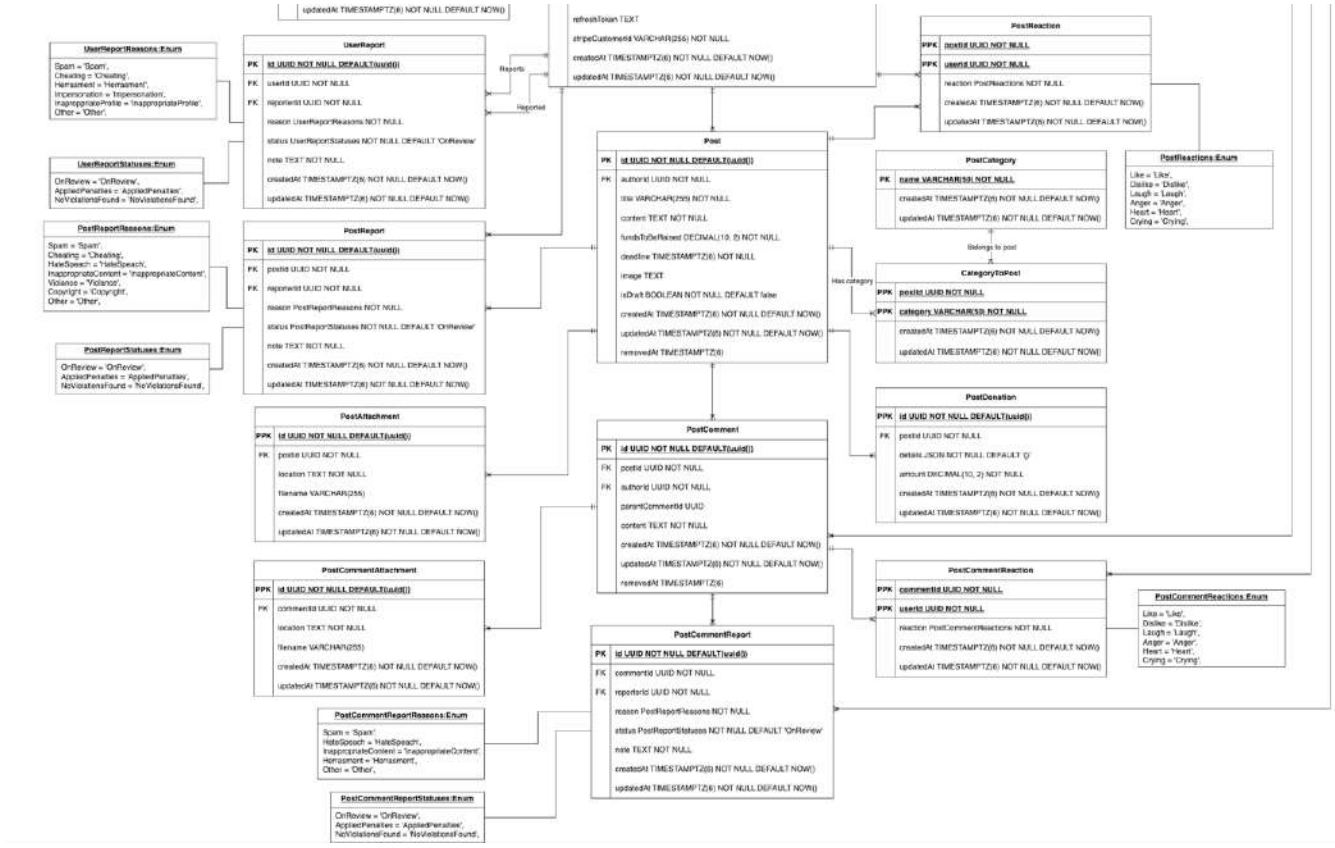

Рис. 2.3.2. Архітектура бази даних застосунку (2 частина)

Аналізуючи надані вище схеми, можна побачити, що база даних застосунку налічує 22 таблиці та 14 перелічуваних типів (enums). Дану систему можна умовно поділити на 3 основні логічні частини: користувачі, дописи та чати. Можливо також окремо виділити модуль коментарів до дописів.

Загальна архітектура застосунку виглядає наступним чином (рис. 2.3.3):

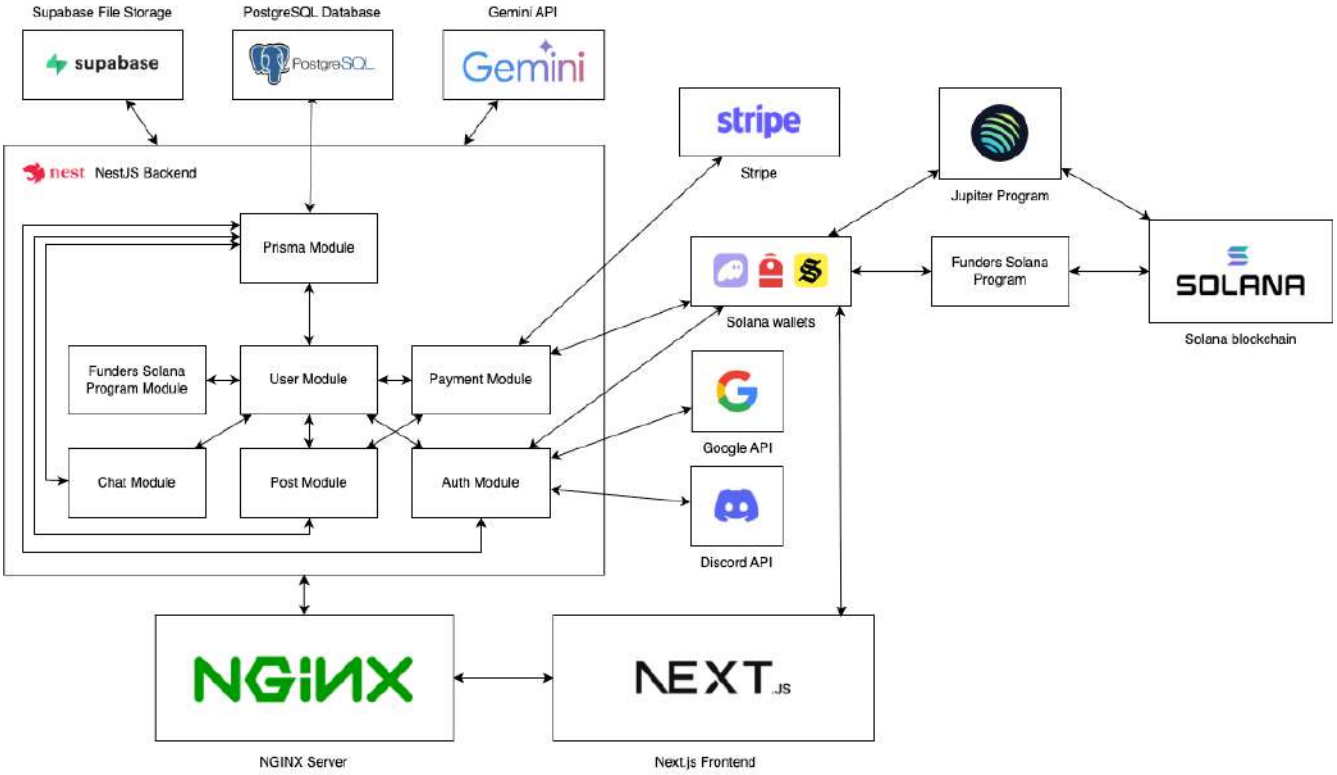

Рис. 2.3.3. Загальна архітектура застосунку

Бекенд частина соціальної мережі розділена на декілька модулів, кожен з яких має свою сферу відповідальності. Окремо варто виділити модуль для роботи з Prisma ORM, оскільки він є проміжною ланкою між бекенд частиною та базою даних PostgreSQL. Також на наданій вище схемі можна побачити активну взаємодію зі сторонніми сервісами, зокрема з Supabase для зберігання файлів та керування ними, Gemini API для реалізації функціоналу фільтрування користувацького контенту, Google та Discord API для авторизації, а також взаємодія з різними криптогаманцями на блокчейні Solana, які в свою чергу комунікують з Jupiter та Funders програмами на блокчейні та є ще одним способом авторизації для користувачів.

Фронтенд та бекенд частина взаємодіють через Nginx, який виконує роль reverse-proxy сервера та переспрямовує запити з фронтенду, виконуючи їх попередню обробку.

2.4. Висновки до розділу 2

В даному розділі було проаналізовано постановку задачі, визначено функціональні та нефункціональні вимоги до системи, а також викладено систему користувацьких ролей та доступні їм можливості в залежності від ролі.

Детально описано стек технологій, які використовувалися для розробки застосунку. Для кожного інструменту розробки було надано короткий опис та обґрунтовано причину вибору саме цієї технології, в окремих випадках роз’яснено переваги та недоліки в порівнянні з конкурентами.

Надано схеми та ілюстрації, які демонструють загальну архітектуру бази даних, а також архітектуру усього застосунку в цілому. Для кожної схеми додано детальний опис, а також роз’яснення щодо тих чи інших архітектурних рішень.

РОЗДІЛ 3. РОЗРОБКА ЗАСТОСУНКУ ТА ОЦІНКА ОТРИМАНИХ РЕЗУЛЬТАТІВ

3.1. Реалізація функціональних компонентів застосунку

Процес розробки соціальної мережі було вирішено розділити на декілька частин, а саме: написання програми на блокчейні Solana з використанням мови програмування Rust, реалізація бекенд та фронтенд частин з використанням фреймворків NestJS та Next.js відповідно.

3.1.1. Реалізація програми на Solana

Для взаємодії соціальної мережі з блокчейном Solana було вирішено розробити окрему програму з використанням мови програмування Rust та фреймворком Anchor, що значно прискорює та полегшує розробку програм на вищезгаданому блокчейні.

Розроблена програма надає наступний функціонал для взаємодії:

- *Ініціалізація конфігурації програми:* встановлює початкові параметри, на даний момент – публічний ключ адміністратора, однак надалі перелік параметрів може збільшуватися;
- *Оновлення конфігурації програми:* інструкція, що дозволяє оновити налаштування, наразі лише публічний ключ адміністратора;
- *Ініціалізація акаунту користувача:* інструкція для створення дата акаунту, що міститиме інформацію про користувача прямо на блокчейні. Зберігає таку інформацію як публічний ключ користувача, публічний ключ NFT, який користувач отримуватиме при реєстрації, та дата створення акаунту;
- *Карбування NFT:* інструкція, яка дозволяє карбувати новий NFT для кожного користувача, що реєструється в розробленій соціальній мережі.

Комісії за виконання будь-якої з описаних вище інструкцій оплачує адміністратор, щоб користувачі не були змушені витрачати власний кошти за даний функціонал.

Загальна файлова структура програми розділена на декілька модулів. Модуль constants передбачає зберігання усіх констант програми, наприклад, список публічних ключів, які мають доступ до початкової ініціалізації програми. Модуль errors містить файли, що у вигляді перелічень описують усі помилки, які виникають під час виконання інструкцій програми. Модуль utils призначений для зберігання допоміжних функцій, які використовуються в інструкціях. Модуль state містить файли з детальним описом структур акаунтів програми, вони відображають дані, що зберігатимуться на блокчейні. Модуль instructions призначений для зберігання основної логіки, а також списків акаунтів, що передаються в кожну інструкцію [14].

Також були написані тести, що покривають увесь написаний функціонал. Вони реалізовані на мові TypeScript та взаємодіють з програмою через AnchorProvider.

Розроблену програму розгорнуто в кластері devnet, вона має адресу 7HCeHcJGhxTH9jx6G9uB5Fn63gnBxFNuAvdL5vE563Q6. У мережі mainnet її немає у зв’язку з значними затратами на розгортання Solana-програм, однак в майбутньому це все ж планується реалізувати.

3.1.2. Реалізація серверної частини

Серверна частина написана на фреймворку NestJS, що побудований на основі Node.js-бібліотеки express. Даний фреймворк передбачає TypeScript як основну мову програмування під час розробки. Це хороший вибір для швидкої розробки та створення надійних веб-застосунків.

Для зберігання даних було обрано СКБД PostgreSQL, взаємодія з БД відбувається через ORM Prisma. Налаштовано міграції, а також прописано логіку

сідерів для швидкого заповнення таблиць наперед визначеними даними, можуть бути викликані за потреби. Надано можливість запускати сідери в різні способи: усі разом або за шаблоном назви, для цього потрібно при запуску команди вказати параметр --matches=<шаблон назви файлу>.

Архітектура бекенду розбита по модулям, де кожен модуль має свою сферу відповідальності. AppModule – основний модуль, в який імпортуються усі інші компоненти програми. В нього додано та налаштовано глобальні компоненти застосунку: ConfigModule, ServeStaticModule, JwtModule, SupabaseModule, PasswordModule, WinstonModule, StripeModule, RedisModule, FundersCoreProgramModule, GeminiModule, PrismaModule, LoggerModule, SolanaModule, UtilsModule та PaymentModule. В них реалізовано важливу логіку для функціонування застосунку, таку як налаштування змінних середовища, файлове сховище, логування, обробка платежів, вза'ємодія з блокчейном та базою даних, робота з мовними моделями тощо. Окрім вищезгаданих компонентів до AppModule також включено 4 основних модулі, які містять сервіси, контролери, сутності та DTO, тобто формують каркас розробленого API. Це AuthModule, UserModule, PostModule та ChatModule. Кожен з них має папку submodules, в якій зберігаються імпортовані підмодулі, які логічно залежні або відносяться до конкретного базового компоненту. Наприклад, до PostModule відносяться CategoryToPostModule, PostAttachmentModule, PostCategoryModule, PostCommentModule, PostDonationModule, PostReactionModule та PostReportModule. Вони, в свою чергу, також можуть містити підмодулі за необхідності. Загалом можна вважати, що кожен з цих модулів відповідає конкретній таблиці в базі даних.

Розроблене API повністю задокументоване завдяки використанню бібліотеки Swagger. Повністю описаний кожен ендпойнт застосунку, а також структура DTO та сутностей, які передаються в тілі запиту або відповіді сервера. Окрім цього, прописана низка unit, integration та end-to-end тестів, що покривають компоненти створеної соціальної мережі.

Авторизація та аутентифікація працюють на основі JWT-токенів. Дана логіка повністю знаходиться в AuthModule. Реалізовані різні стратегії авторизації (jwt strategy та jwt refresh strategy), а також гарди, які з цими стратегіями взаємодіють та інші функціональні компоненти для полегшення роботи з цим модулем. При авторизації користувачу надається пара JWT-токенів (access та refresh токени). Access token розглядається як ключ доступу до захищених ендпойнтів застосунку, постійно перевіряється в jwt стратегії для того, що перевірити, чи користувач має право взаємодіяти з сервером через конкретну кінцеву точку. В свою чергу, refresh token використовується для періодичного оновлення пари JWT-токенів з метою підвищення безпеки застосунку, оскільки access token користувача може бути викрадений зловмисниками. Пара цих токенів віддається клієнту лише при реєстрації, авторизації або під час «рефрешу» шляхом визначення хедеру Set-Cookie у відповіді сервера. Вони позначені прапорцем HttpOnly, щоб унеможливити доступ до них зі сторони клієнта, це значно підвищує безпеку застосунку.

Окрему увагу варто приділити SolanaModule та FundersCoreProgramModule, які відповідають за взаємодію API з блокчейном. FundersCoreProgramModule вважається обгорткою над AnchorProvider класом, туди передається IDL розробленої Solana програми, а також низку інших допоміжних параметрів. SolanaModule, в свою чергу, містить сервіс для взаємодії користувачів з блокчейном, де API виступає в ролі проміжної ланки. В ньому виконуються на відправляються блокчейн-транзакції, зокрема створення користувацького дата-акаунту або карбування NFT при реєстрації [5; 14].

Окрім цього, розроблено функціонал для автоматичного фільтрування спаму або неприйнятного контенту, який користувачі додають на сайт. Перевірка відбувається для контенту дописів та коментарів під ними, оскільки це є потенційно найбільш вразливі частини, де користувачі можуть писати будь-яку інформацію. Для реалізації цього функціоналу було вирішено використати мовні моделі від Gemini. При створенні чи редагуванні нового допису або коментаря їх контент

передається на перевірку в GeminiService та у разі знаходження неприйнятної інформації такі сутності одразу видаляються з соціальної мережі.

Задля підвищення рівня безпеки API було також додано валідацію тіла запитів, які шлють користувачі. Для кожного тіла запиту створено DTO-клас з параметрами валідації усіх його полів. Це вдалося реалізувати завдяки використанню бібліотек class-validator та class-transformer.

Файловим сховищем було обрано сервіс Supabase, оскільки він надзвичайно простий, швидкий та надає досить зручний функціонал для розробників, що значно полегшує інтеграцію цього сервісу в веб-застосунок. Окрім цього, Supabase надає досить хороші умови навіть при використанні безкоштовного плану, тому було обрано саме цей сервіс [17].

Додатково було прописано файли для розгортання застосунку на віддаленому сервері. Створено базову конфігурацію для сервера Nginx, а також налаштовано файли Docker та docker-compose для development та production середовищ.

3.1.3. Реалізація клієнтської частини

Клієнтська частина застосунку розроблена з використанням мови програмування TypeScript, фреймворку Next.js, який є надбудовою над JavaScript-бібліотекою React та надає значно ширший набір функціоналу, його особливістю є підтримка SSR, що дає можливість перенести частину процесу рендерингу на сервер, зменшуючи навантаження на клієнтську частину. Для стилізації фронтенду активно використовувався фреймворк TailwindCSS, що надає набір передвизначених класів, які можна комбінувати між собою для досягнення бажаного результату. Він значно прискорив процес розробки frontend-частини застосунку.

Валідація даних на стороні клієнта відбувається з використанням бібліотеки valibot. Даний процес передбачає використання спеціально створених для валідації структур – схем, де кожне поле містить параметри для перевірки вхідних даних.

Авторизація та аутентифікація реалізована на основі JWT-токенів. В цілому, дана логіка повністю реалізована на серверній частині, де пара access та refresh JWT-токенів записується в cookies з HttpOnly прапорцем для підвищення безпеки. На фронтенді до запиту через бібліотеку axios передається параметр withCredentials: true, що гарантує включення cookies до запиту, що надсилається на сервер. Для користувачів всього доступно 4 способи авторизації: через email та пароль, за допомогою Google OAuth2, Discord OAuth2, або Solana Wallet.

Навігація клієнтської частини застосунку працює на основі файлової структури проєкту – логіка за замовчуванням для програм, написаних на Next.js. Загалом, максимально було використано усі переваги Next.js, а саме server side rendering, server actions, streaming, функціонал для покращення SEO та інші. Загальна структура проєкту розбита на 2 логічні частини: routes та core, де routes призначений для зберігання шаблонів та сторінок, а також відповідає за навігацію, в той час як core зберігає весь інший функціонал: UI-компоненти, server actions, утиліти, константи, провайдери, хуки, клієнтське глобальне сховище даних тощо [12].

Взаємодія з гаманцем користувача відбувається з використанням npm-бібліотек @solana/wallet-adapter-base, @solana/wallet-adapter-react, @solana/wallet-adapter-react-ui, @solana/wallet-adapter-wallets та @solana/web3.js. Оплата в криптовалюті реалізована з використанням Jupiter, а саме їх бібліотеки @jup-ag/api. При підключенні гаманця, відбувається автоматичний пошук його асоційованих токен акаунтів, дістається їх token mint адреса, symbol (назва токену) та баланс. Акаунти з нульовим балансом фільтруються та не відображаються у списку. Для кожного балансу акаунту відображається ціна в USDC для розуміння максимально можливої суми фінансового внеску користувача цим токеном. Під час створення

донату надсилається 2 запити: обрахунок кількості токенів для обміну та запит на формування інструкцій обміну на основі попереднього запиту. До отриманих інструкцій додається ще інструкція для трансферу, яка надсилає отримані в результаті обміну USDC-токени на загальний гаманець застосунку. На основі цих інструкцій формується versioned transaction, яка підписується гаманцем користувача та надсилається на блокчейн, паралельно дублюючи інформацію про фінансовий внесок в БД [9; 18].

Загалом було розроблено досить зручний та привабливий інтерфейс для користувачів (рис. 3.1.3.1-3.1.3.16).

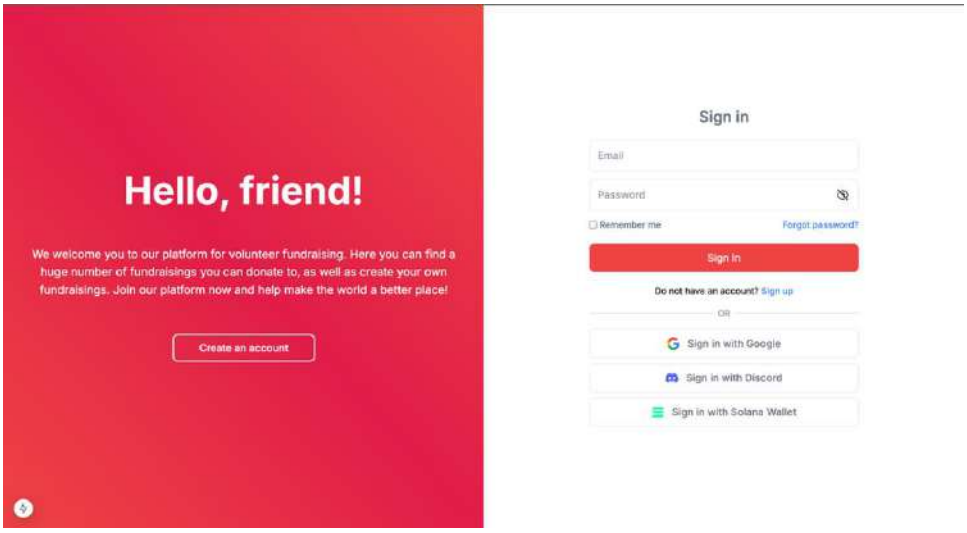

Рис. 3.1.3.1. Сторінка авторизації

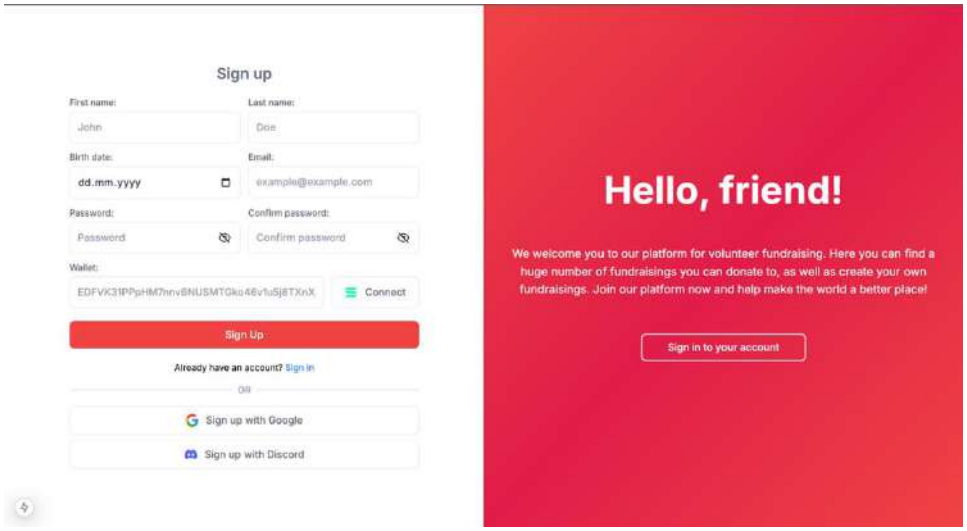

Рис. 3.1.3.2. Сторінка реєстрації

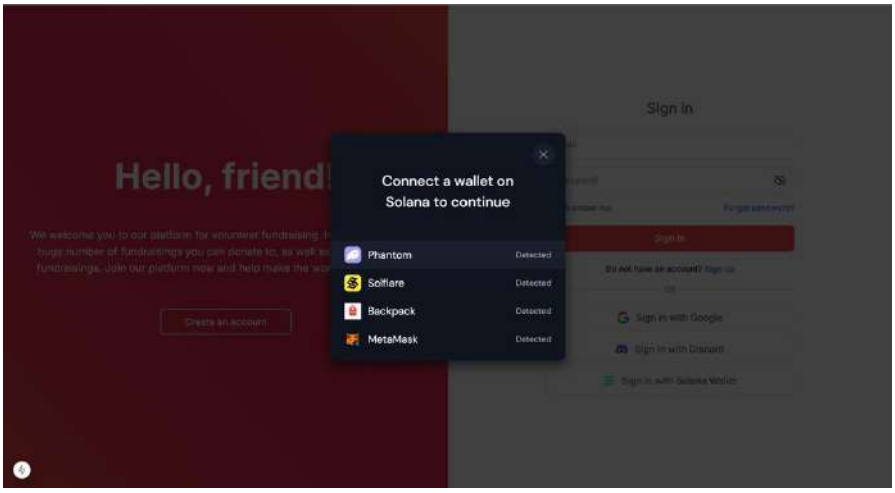

Рис. 3.1.3.3. Підключення гаманця для авторизації

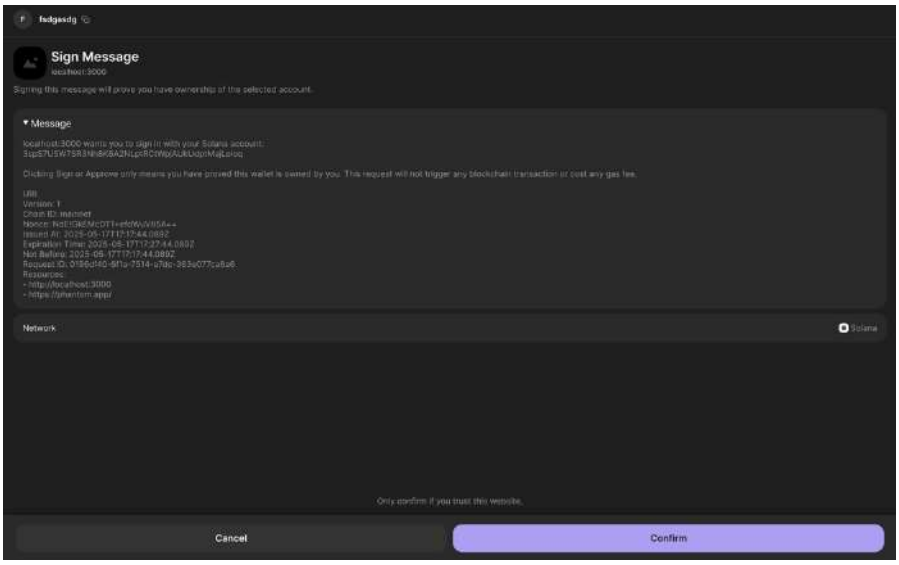

Рис. 3.1.3.4. Повідомлення для підпису, щоб авторизуватися через гаманець

Рис. 3.1.3.5. Отриманий користувачем Soulbound NFT після реєстрації з гаманцем

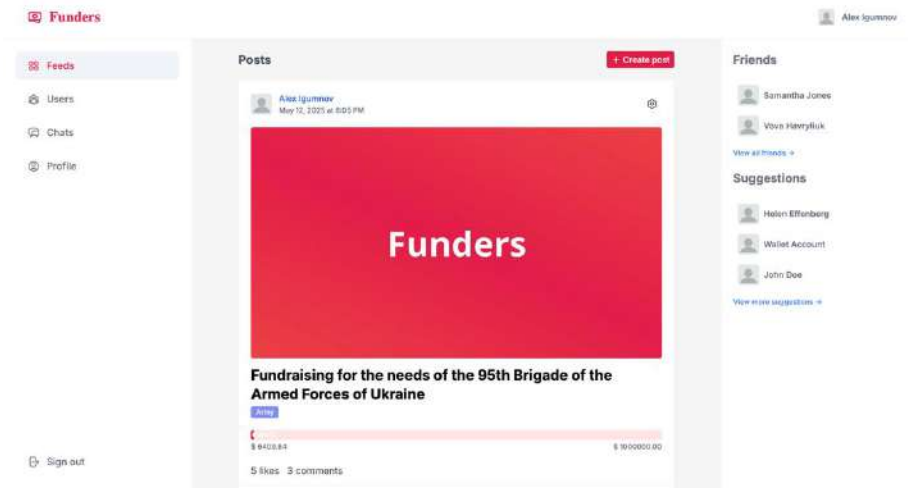

Рис. 3.1.3.6. Головна сторінка з дописами

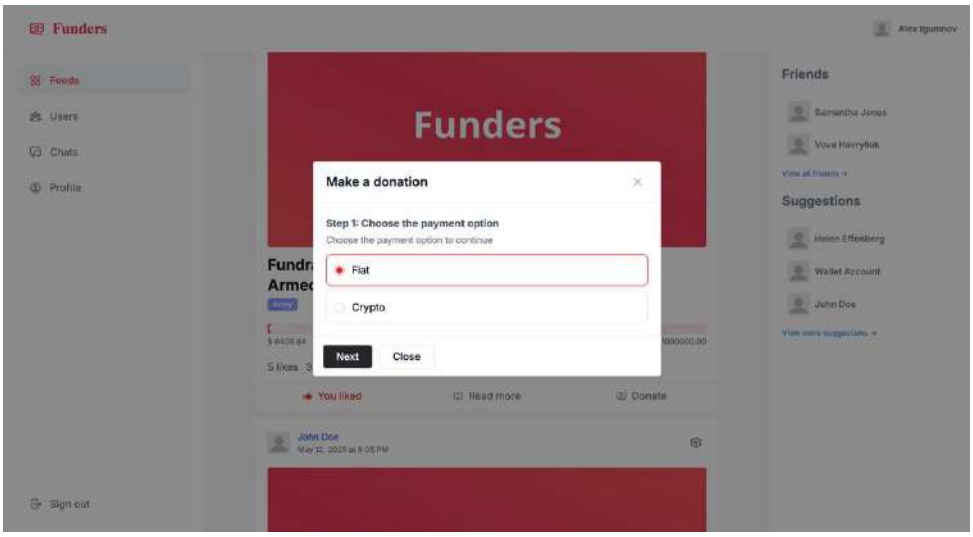

Рис. 3.1.3.7. Створення грошового внеску (вибір між фіатом та криптовалютою)

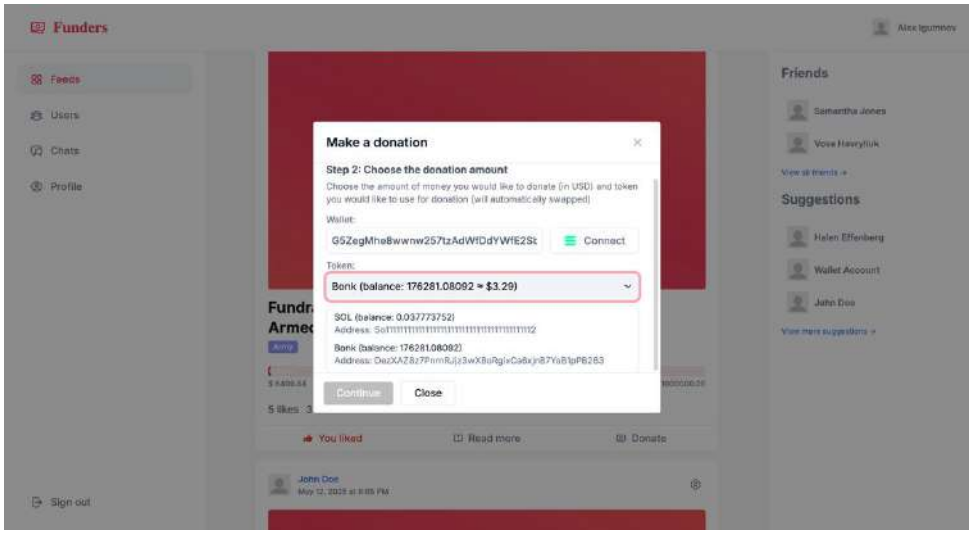

Рис. 3.1.3.8. Створення грошового внеску (підключення гаманця та вибір токену для оплати)

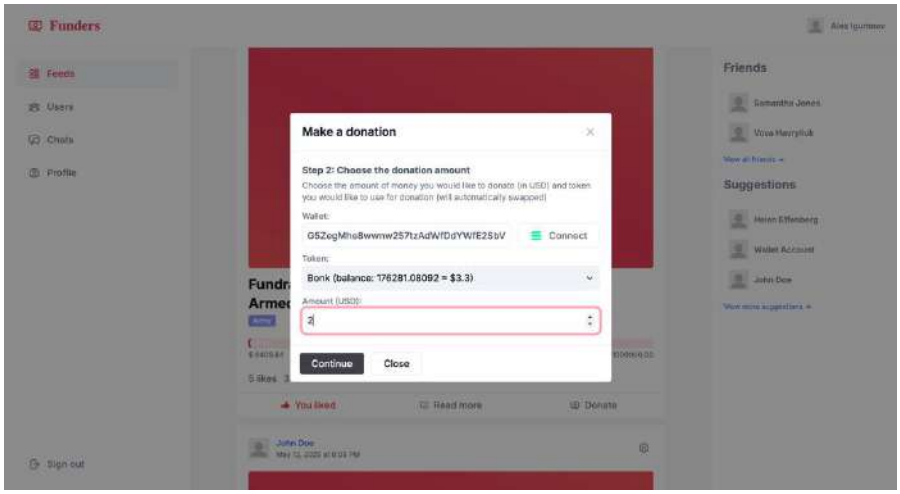

Рис. 3.1.3.9. Створення грошового внеску (вибір розміру пожертви)

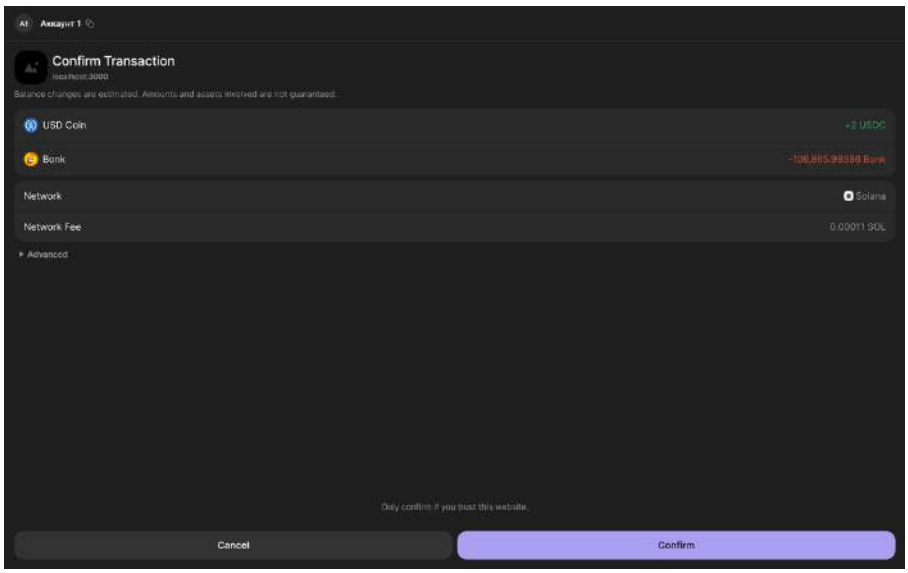

Рис. 3.1.3.10. Підтвердження транзакції для фінансової пожертви

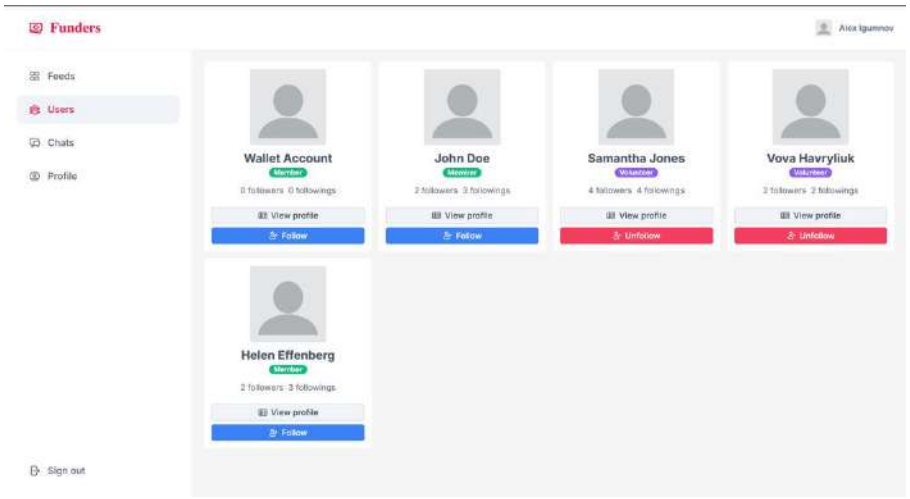

Рис. 3.1.3.11. Сторінка з користувачами

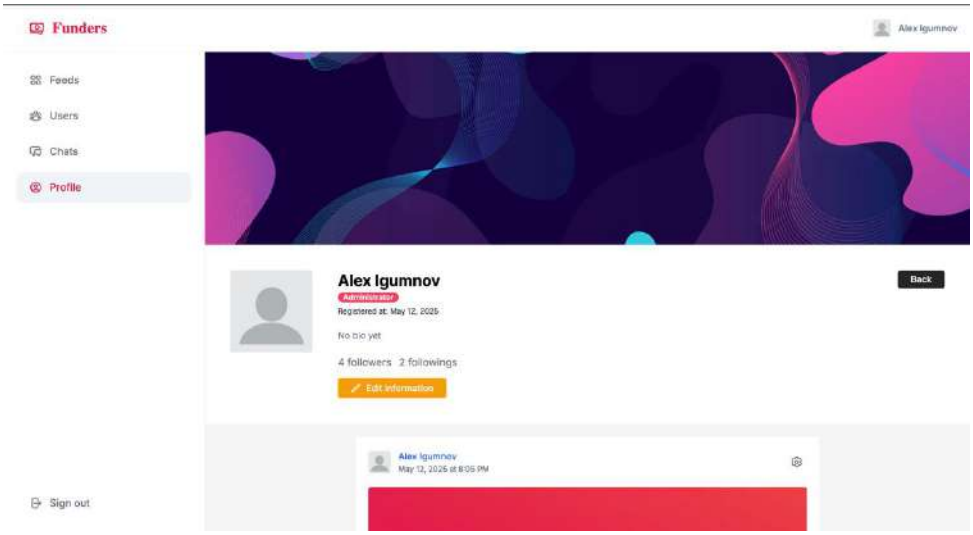

Рис. 3.1.3.12. Профіль користувача

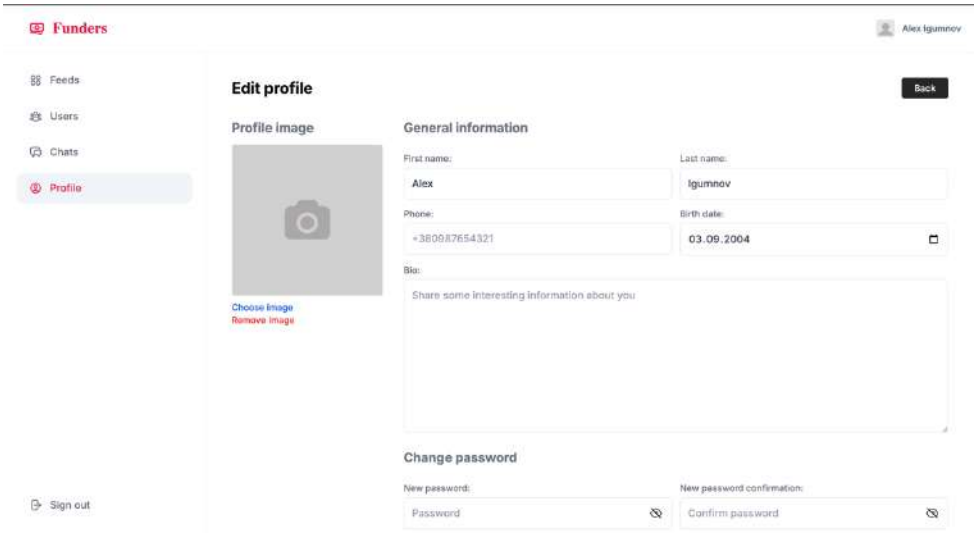

Рис. 3.1.3.13. Редагування профілю

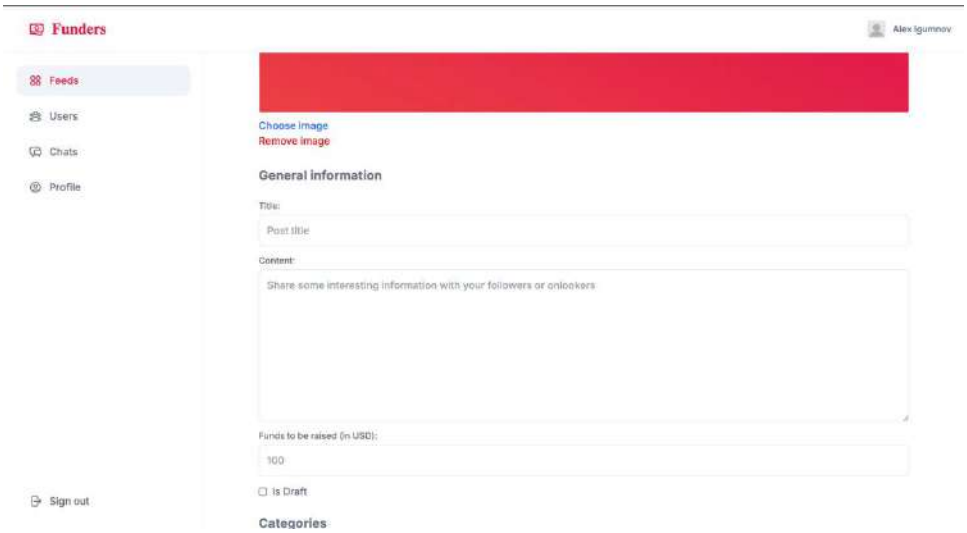

Рис. 3.1.3.14. Створення допису

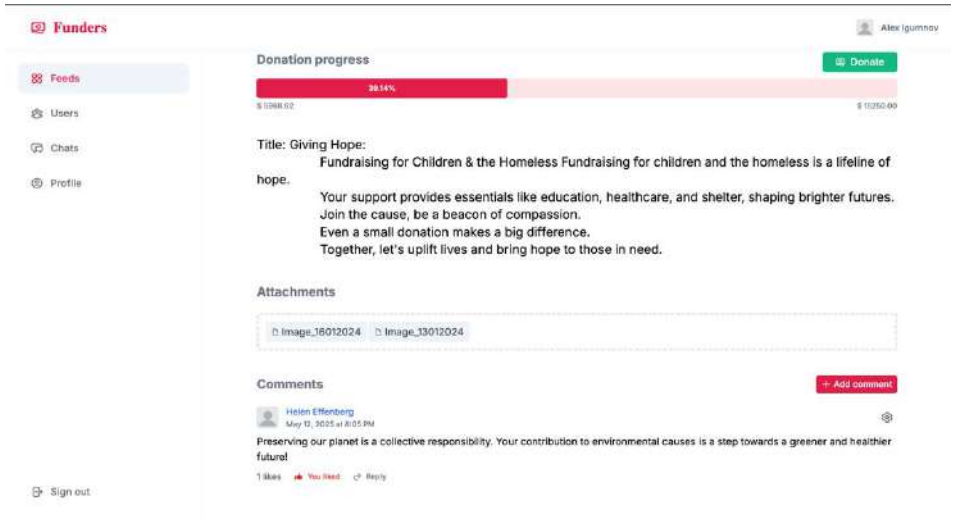

Рис. 3.1.3.15. Сторінка з деталями про допис

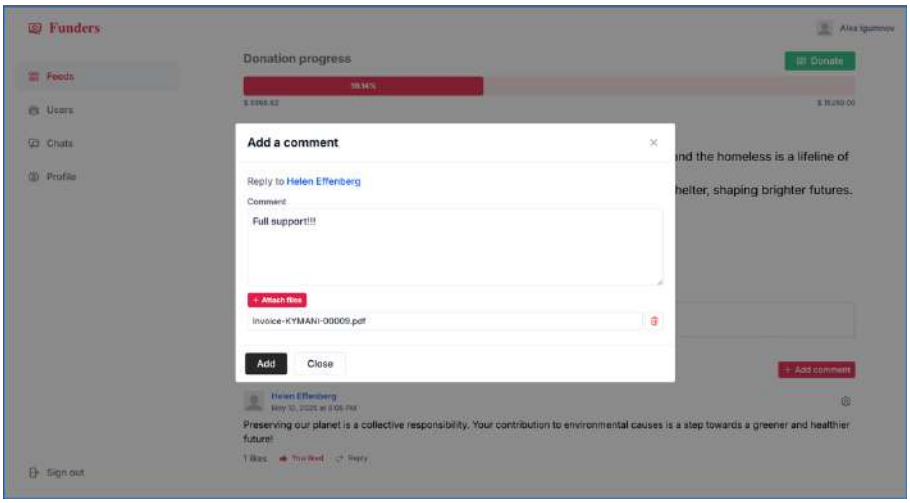

Рис. 3.1.3.16. Створення коментаря до допису

3.2. Тестування системи

В процесі розробки проєкту для кожного модуля було здійснено покриття автоматизованими тестами за допомогою бібліотеки Jest. До їх переліку входили unit, integration та end-to-end тести, які забезпечили можливість вичерпного тестування усієї системи.

Використовувався підхід розробки, що передбачав покриття різними видами тестів кожного окремого логічного компоненту застосунку одразу після його реалізації. Це дозволило поетапно та безперешкодно прописувати окремі частини

коду при цьому бути певним, що новий функціонал не змінює поведінку вже існуючого, та давало можливість швидко брати до роботи наступні частини застосунку.

Також було прописано набір скриптів, які дозволяють запускати тести у різні способи, наприклад, запустивши повне тестування системи, модуля чи окремого файлу.

Додатково реалізовано функціонал автоматичного запуску тестів при оновленні змін коду на віддаленому репозиторії GitHub. Для цього було прописано CI/CD workflow, в окремому .yml файлі, який запускається при створенні пул реквестів, а також при додаванні коду у гілки develop та main. Даний підхід гарантує, що код, розміщений на віддаленому репозиторії GitHub повністю робочий і не містить некоректно працюючих компонентів застосунку. Для реалізації даної логіки було використано інструмент розробки GitHub Actions.

3.3. Підготовка до розгортання

Після написання основного коду застосунку було розпочато процес підготовки до його розгортання на серверах DigitalOcean. Для даної соціальної мережі передбачено два окремих віддалених сервери для фронтенд та бекенд частини відповідно. Також було придбано доменне ім’я для зручного пошуку застосунку в мережі Інтернет та приховування IP-адреси серверів [10].

Далі було розпочато налаштування серверів з метою їх відповідності основним стандартам безпеки. На кожному з них створено окремого non-root юзера, для того, щоб у разі проникнення до сервера зловмисник не мав доступу до функціоналу адміністратора та не міг виконати ряд зловмисницьких дій. Після цього, виконано налаштування ssh_config файлу, змінено SSH-порт зі стандартного значення 22 на випадкове значення в межах від 50000 до 65535. Це зроблено для того, щоб зменшити кількість потенційних атак ботів на загальновідомий порт та

ускладнення ручних атак зловмисників. Також змінено ряд параметрів, що відповідають за способи авторизації на сервер, зокрема було залишено можливість авторизації лише з використанням SSH-ключа, що також підвищує рівень безпеки сервера. В свою чергу, новий SSH-ключ створено на локальному комп'ютері, а його публічну версію додано до файлу authorized_keys на віддаленому сервері для того, щоб працювала авторизація через SSH [2].

Наступним кроком було оновлення конфігурації для ssh.socket сервісу, оскільки там також треба було замінити порт з 22 на власне значення. Якщо цього не зробити, то з високою ймовірністю сервер буде заблоковано через неправильну конфігурацію і єдиним варіантом буде скидання усіх його параметрів до початкових. Після виконання даного кроку треба перезапустити ssh сервіс, щоб активувати зміни.

Далі було налаштовано Uncomplicated Firewall, відкрито доступ до SSH-порту, та Nginx Full (тобто 80 порт для HTTP та 443 порт для HTTPS з'єднання). Після цього було встановлено та налаштовано такі важливі інструменти розробки як Node.js з сучасним пакетним менеджером pnpm, Git, Docker та Nginx [2].

До Nginx приділено особливу увагу, його конфігураційний файл є дуже важливим, оскільки дозволяє встановити ряд критичних параметрів, які підвищують продуктивність та безпеку усього застосунку. У даному файлі прописано налаштування перенаправлень на внутрішні порти сервера, HTTPS з'єднання, доменне ім'я, додаткові хедери для запитів, зберігання та керування різними сертифікатами, налаштування TLS протоколу, різні шифрування тощо (рис 3.3.1, 3.3.2).

```
upstream backend {
  server localhost:8000;
}

client_max_body_size 100M;

server {
  listen 80;
  listen [::]:80;

  server_name api.funders.com www.api.funders.com;
  server_tokens off;

  location / {
    return 301 https://$host$request_uri;
  }
}

server {
  listen 443 ssl;
  listen [::]:443 ssl;
  server_name api.funders.com www.api.funders.com;
  server_tokens off;

  include snippets/well-known-location.conf;
  ssl_certificate /etc/letsencrypt/live/api.funders.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/api.funders.com/privkey.pem;
  ssl_trusted_certificate /etc/letsencrypt/live/api.funders.com/fullchain.pem;
  ssl_dhparam /etc/ssl/certs/dhparam.pem;

  ssl_session_timeout 1d;
  ssl_session_cache shared:SSL:10m;
  ssl_session_tickets off;

  ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;

  ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-
AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-
AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-
SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-
SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-
SHA:AES256-SHA:DES-CBC3-SHA:!DSS:!aNULL:!eNULL:!EXPORT:!DES:!MD5:!PSK:!RC4';
  ssl_prefer_server_ciphers on;
```

Рис. 3.3.1. Конфігураційний файл Nginx для серверу з бекендом (1 частина)

```
  ssl_stapling on;
  ssl_stapling_verify on;

  resolver 8.8.8.8 8.8.4.4 valid=300s;
  resolver_timeout 5s;

  add_header Strict-Transport-Security max-age=15768000;
  add_header Strict-Transport-Security "max-age=63072000; includeSubDomains; preload";
  add_header X-Frame-Options DENY;
  add_header X-Content-Type-Options nosniff;

  location /api/v1 {
    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_redirect off;
    proxy_ssl_session_reuse off;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_cache_bypass $http_upgrade;
  }

  location /api/docs {
    proxy_pass http://backend;
    proxy_http_version 1.1;
    proxy_set_header X-NginX-Proxy true;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_redirect off;
    proxy_ssl_session_reuse off;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_cache_bypass $http_upgrade;
  }
}
```

Рис. 3.3.2. Конфігураційний файл Nginx для серверу з бекендом (2 частина)

Одразу після налаштування Nginx було інстальовано сервіс certbot для автоматичного виписування безкоштовних ssl-сертифікатів. У зв'язці з Nginx вдалося налаштувати HTTPS-з'єднання для розробленої соціальної мережі.

Фінальним етапом стало завантаження та конфігурація сервісу fail2ban, який дозволяє фільтрувати та, за потреби, блокувати підозрілі IP-адреси або ті, що здійснюють різні типи атак на застосунок.

В результаті було отримано повністю сконфігуровані сервери, на яких можна розгортати фінальну версію платформи. Все, що для цього потрібно – клонувати віддалений GitHub репозиторій на сервер (під non-root користувачем) та запустити код застосунку.

3.4. Аналіз отриманих результатів та їх відповідності поставленим вимогам

Результатом виконаної роботи є повністю функціонуючий веб-застосунок у форматі соціальної мережі для волонтерських зборів коштів. Створена платформа також містить інтегрований web3-функціонал, а саме запис інформації про користувачів на блокчейні Solana, нагородження користувачів пам'ятною Soulbound NFT, функціонал авторизації через криптогаманці та обробка криптовалюних платежів з використанням DEX-агрегатора Jupiter.

Усі функціональні та нефункціональні вимоги, що були викладені в пункті 2.1, були виконані у повному обсязі, розподіл ролей у готовому застосунку цілком відповідає тому, який було визначено на етапі постановки задачі.

Система повністю протестована з використанням різних видів тестувань. Усі знайдені функціональні проблеми були оперативно вирішені, застосунок працює стабільно та має досить високу продуктивність.

Соціальна мережа була розгорнута на серверах, наданих DigitalOcean. Кожен сервер налаштований з дотриманням усіх основних стандартів безпеки, додатково

було налаштовано інші інструменти для підвищення рівня безпеки серверів, а саме задано конфігурацію для Uncomplicated Firewall та сервісу fail2ban, який проводить фільтрацію підозрілих IP-адрес або тих, що намагаються виконати заборонені дії у відношенні до сервера, а також надає окремий функціонал для адміністраторів з метою блокування таких IP-адрес.

Застосунок наразі перебуває у відкритому доступі та готовий до активного використання.

3.5. Порівняння з традиційними централізованими рішеннями

Створена соціальна мережа все ж вважається переважно централізованим рішенням, оскільки розгорнута на серверах, які є під контролем платформи DigitalOcean. Тому значна кількість функціоналу та підходів до розробки не сильно відрізняються від тих, що притаманні традиційним веб-застосункам. Це зумовлено високим рівнем складності в імплементації повної децентралізації компонентів застосунку, а також передбачає значні фінансові витрати, оскільки за виділення місця на блокчейні навіть для невеликої програми потрібно зробити чималий фінансовий внесок.

Однак розробленій платформі притаманний певний набір відмінностей від звичайних застосунків подібного типу. Наприклад, новий метод авторизації через криптогаманець, який дозволяє реєструватися та авторизуватися навіть швидше ніж з використанням традиційних Google, Discord або інших OAuth2 рішень. До того ж, даний підхід не перенаправляє користувача на окрему сторінку, а лише просить користувача підписати повідомлення для входу, що значно економить час, та робить сам процес в рази простішим. Однак попри переваги варто також виділити і один недолік – даним способом можуть скористатися лише ті користувачі, які мають криптогаманець.

Ще однією значущою перевагою можна виділити можливість створення фінансових внесків з використанням криптовалюти. Наразі цифрові гроші набирають надзвичайно великої популярності і у великої кількості людей вже є окремі фінансові відкладення у криптовалюті. Тому додатковий спосіб оплати у розробленому застосунку лише надає користувачам більше можливостей для донатів. Тепер у них є вибір в який спосіб вони бажають здійснити грошовий переказ. Криптовалютні платежі на блокчейні Solana дуже швидкі та займають секунди на відміну від оплату через банк, що в деяких випадках може займати навіть декілька днів. Для кампаній по збору коштів час відіграє важливу роль, тому швидкість та зручність переказу можна вважати вагомими критеріями для застосунків подібного типу.

Окрім вищезазначених відмінностей також варто відзначити базову взаємодію застосунку з блокчейном, а саме надання пам’ятного Soulbound NFT користувачам, що вважається індикатором наявності акаунту в розробленій соціальній мережі, а також збереження інформації про користувачів на блокчейні. Це було зроблено з метою подальшої інтеграції web3-функціоналу в застосунок, оскільки децентралізовані технології наразі набувають активної популярності. В майбутньому планується розширювати цю логіку. Проте варто відзначити, що у аналогічних застосунків даний функціонал взагалі відсутній [5; 14].

Отже на даному етапі можна зазначити, що відмінності від традиційних централізованих рішень все ж є, однак застосунок не є повністю децентралізованим і знаходиться у перехідній стадії між web2 та web3.

3.6. Перспективи подальшого розвитку

Розроблений застосунок має досить великі перспективи для розвитку у майбутньому. Активний розвиток веб-технологій та тенденція до поступової інтеграції web3-функціоналу у вже існуючі застосунки зумовлює актуальність

удосконалення розробленої соціальної мережі шляхом встановлення її взаємодії з децентралізованими технологіями, зокрема блокчейном.

Варіантами для вдосконалення можуть стати розробка користувацької системи рівнів, а також реалізація функціоналу квестів та досягнень. Окрім цього, хорошою концепцією може також стати розробка реферальної системи. Додавання даних механізмів передбачає в тому числі дублювання їх логіки в системі блокчейну задля підвищення прозорості та надійності роботи. Очікується, що запропоновані ідеї дадуть можливість для залучення більшої кількості нових користувачів, а також підвищать рівень їх зацікавленості та активності у створеній соціальній мережі.

Ще однією ідеєю для вдосконалення є додавання гейміфікації до застосунку. Цей задум тісно переплітається з описаними вище системою користувацьких рівнів та досягнень, однак, в свою чергу, передбачає отримання користувачами різних винагород за їх активність, також можливий інший вид взаємодії в ігровій формі. Підставою для цієї пропозиції є те, що наразі відбувається значний приріст зацікавленості web3-ком'юніті до ігрових проєктів на різних блокчейнах. Водночас, помітно виросла активність розробки подібних застосунків, що може бути зумовленим значним приростом інвестицій в даний сектор. Станом на 27 листопада 2024 року, за даними звіту криптобіржі Gate.io, капіталізація ігрового ринку (GameFi) досягла позначки 19,584 мільярда доларів США і, як очікується, збереже середньорічні темпи зростання на рівні 29,5% у найближчі роки. Тож інтеграція запропонованого функціоналу може не лише залучити більшу аудиторію, але й отримати значні інвестиції [7].

Отже, враховуючи унікальність розробленого застосунку та відсутність аналогів в даній предметній області, а також беручи до уваги світові тренди, що стосуються розвитку веб-технологій та криптоіндустрії, можна констатувати факт, що створена соціальна мережа є надзвичайно перспективною в найближчому майбутньому та має великий потенціал для подальшого розвитку. Вона вже встигла взяти участь у Solana Radar Global Hackathon, де звернула на себе увагу окремих

розробників з інших країн, а також отримала фінансову винагороду від українського web3-ком'юніті Kumeka Team, як студентський проєкт, що має потенціал для майбутнього розвитку.

3.7. Висновки до розділу 3

У даному розділі викладено повний процес розробки застосунку. Надано детальний опис написання коду для програми на блокчейні Solana, бекенд та фронтенд частин платформи.

Надано повну інформацію щодо процесу тестування, які види тестувань використовувалися та результати їх виконання.

Представлено процес підготовки застосунку до розгортання. Викладено детальні інструкції щодо налаштування віддалених серверів, на яких планувалося розгортання бекенд та фронтенд частин додатку. Розглянуто налаштування Nginx для кожного зі згаданих серверів.

Виконано аналіз отриманих результатів та перевірено їх відповідність поставленій задачі, функціональним та нефункціональним вимогам до застосунку, які були викладені у пункті 2.1. Здійснено порівняння розробленої соціальної мережі з традиційними централізованими рішеннями, визначено переваги та недоліки, а також перспективи подальшого розвитку даного проєкту.

ВИСНОВКИ

В ході виконання роботи було визначено мету та актуальність цільового застосунку, проаналізовано платформи зі схожою тематикою. Виконане порівняння з конкурентами із зазначенням переваг і недоліків.

З огляду на проведений аналіз було сформовано постановку задачі, визначено функціональні та нефункціональні вимоги застосунку. Спроєктовано базу даних та загальну архітектуру застосунку, які надані в тексті роботи у вигляді схем та роз’яснень до них.

Результатом виконання роботи є розроблена соціальна мережа для проведення кампаній по збору коштів на волонтерських засадах з вбудованою web3-функціональністю. Інтегровано авторизацію користувачів різними способами, зокрема, надано можливість авторизації через криптогаманці в мережі Solana. Також впроваджено платіжну функціональність з використанням криптовалюти за допомогою DEX-агрегатора Jupiter. Окрім вищезазначених можливостей було також реалізовано керування дописами і коментарями під ними, фільтрування на основі ШІ некоректного контенту або того, що містить спам, та функціональність для взаємодії користувачів між собою.

Виконане ретельне тестування розробленого застосунку та його розгортання на віддаленому сервері з дотриманням багатьох стандартів безпеки.

Розроблена соціальна мережа має значний потенціал для подальшого розвитку, кодова база застосунку спроєктована таким чином, що дає можливість швидко та безперешкодно масштабуватися без впливу на загальний рівень безпеки та продуктивності системи.

Даний проєкт був учасником Solana Radar Global Hackathon та в студентському треку потрапив в 5 кращих, що підтверджує факт актуальності створеного застосунку та зацікавленості в ньому як інвесторів, так і пересічних користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aaron. Proof of Stake (POS) [Електронний ресурс] / Aaron // Binance Academy. – Режим доступу: https://academy.binance.com/en/glossary/proof-of-stake. – Назва з екрана.
2. Agrawal A. A runbook for setting up a linux based secure, production web server, for serving static web content as well as deploying node.js based web applications. [Електронний ресурс] / Avinash Agrawal // GitHub Gist. – Режим доступу: https://gist.github.com/journeymanavi/dc7df599dd7618a12528c553ea47b1b1. – Назва з екрана.
3. A study of blockchain technology development and military application prospects [Електронний ресурс] / Y. Zhu [та ін.] // Journal of physics: conference series. – 2020. – Т. 1507, № 5. – С. 2. – Режим доступу: https://iopscience.iop.org/article/10.1088/1742-6596/1507/5/052018/pdf. – Назва з екрана.
4. Brooker K. “I was devastated”: the man who created the world wide web has some regrets [Електронний ресурс] / Katrina Brooker // Vanity Fair. – Режим доступу: https://www.vanityfair.com/news/2018/07/the-man-who-created-the-world-wide-web-has-some-regrets. – Назва з екрана.
5. Creating assets [Електронний ресурс] // Metaplex Developer Hub. – Режим доступу: https://developers.metaplex.com/core/create-asset. – Назва з екрана.
6. Documentation [Електронний ресурс] // NestJS Docs. – Режим доступу: https://docs.nestjs.com/. – Назва з екрана.
7. Gate research: analyzing the 2024 GameFi market: insights from data, trends, and future outlook [Електронний ресурс] // Gate.io. – Режим доступу: https://www.gate.io/learn/articles/gate-research-analyzing-the-2024-game-fi-market-insights-from-data-trends-and-future-outlook/4871. – Назва з екрана.

8. Gemini developer API [Електронний ресурс] // Google AI for Developers. – Режим доступу: https://ai.google.dev/. – Назва з екрана.
9. How to connect a wallet with React [Електронний ресурс] // Solana. – Режим доступу: https://solana.com/uk/developers/cookbook/wallets/connect-wallet-react. – Назва з екрана.
10. How to deploy from container images [Електронний ресурс] // DigitalOcean Docs. – Режим доступу: https://docs.digitalocean.com/products/app-platform/how-to/deploy-from-container-images/. – Назва з екрана.
11. In-depth analysis of Jupiter Aggregator [Електронний ресурс] // Mirror. – Режим доступу: https://mirror.xyz/kingson.eth/LUmns90hfAXd0GVYV2APGmhwOygSG3xtuUY8gnf0tlg. – Назва з екрана.
12. Introduction [Електронний ресурс] // Next.js Docs. – Режим доступу: https://nextjs.org/docs. – Назва з екрана.
13. Introduction [Електронний ресурс] // RustBook. – Режим доступу: https://rust-book.cs.brown.edu/ch00-00-introduction.html. – Назва з екрана.
14. Local development [Електронний ресурс] // Anchor Docs. – Режим доступу: https://www.anchor-lang.com/docs/quickstart/local. – Назва з екрана.
15. Phantom sign in with Solana (SIWS) [Електронний ресурс] // Phantom. – Режим доступу: https://phantom.com/learn/developers/sign-in-with-solana. – Назва з екрана.
16. Prisma documentation [Електронний ресурс] // Prisma Docs. – Режим доступу: https://www.prisma.io/docs. – Назва з екрана.
17. Resumable uploads [Електронний ресурс] // Supabase. – Режим доступу: https://supabase.com/docs/guides/storage/uploads/resumable-uploads. – Назва з екрана.
18. V6 swap API [Електронний ресурс] // Jupiter Develop. – Режим доступу: https://dev.jup.ag/docs/old/apis/swap-api. – Назва з екрана.

19. What is Docker? [Електронний ресурс] // Docker Docs. – Режим доступу: https://docs.docker.com/get-started/docker-overview/. – Назва з екрана.
20. What is Git? [Електронний ресурс] // Atlassian. – Режим доступу: https://www.atlassian.com/git/tutorials/what-is-git. – Назва з екрана.