НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ПОГОДЖЕНО **Декан факультету**	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ** **Завідувач кафедри інформаційних систем і технологій**
_____Ігор БОЛБОТ (підпис)	**_____Михайло ШВИДЕНКО** (підпис)
«_____» _____2026 р.	«_____» _____2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка блокчейн-гаманця з функціоналом підтримки NFT»
Спеціальність «122 Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Керівник бакалаврської кваліфікаційної роботи	_____ (підпис)	**Костянтин РОГОЗА**
Виконав	_____ (підпис)	**Богдан ДРАБОВСЬКИЙ**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних систем і технологій

к.е.н., доцент_____Михайло ШВИДЕНКО
(підпис)

«___» _____2026 р.

ЗАВДАННЯ ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Драбовському Богдану Сергійовичу

(прізвище, ім'я, по батькові)

Спеціальність «122 Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»
Тема бакалаврської кваліфікаційної роботи
«Розробка блокчейн-гаманця з функціоналом підтримки NFT»
затверджена наказом від « 06 » січня 2026 р. № 46 «С»

Термін подання завершеної роботи на кафедру «22» травня 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи: аналіз предметної області, сформовані та підтверджені транзакції, інформація про баланс користувача, журнал дій користувача, аналітичні дані.

Перелік питань, що підлягають дослідженню:

1. Системний аналіз архітектур сучасних блокчейн-гаманців та стандартів криптографічного захисту.
2. Обґрунтування та опис п'ятирівневої багатошарової архітектури некастодіального Web3-додатка NexaVault.
3. Проєктування реляційної моделі бази даних на основі SQLite та LocalStorage із впровадженням чотирьох тригерів контролю цілісності фінансових транзакцій.
4. Реалізація клієнтської логіки на базі React 18, Vite та ethers.js v6.
5. Інтеграція модулів взаємодії з NFT (ERC-721/ERC-1155) та аналітики ринку.

Дата видачі завдання « 06 » січня 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Костянтин РОГОЗА**

Завдання взяв до виконання _____ **Богдан ДРАБОВСЬКИЙ**

(підпис)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

У записці використовуються такі скорочення та умовні позначення:

– СКБД – система керування базами даних;
– ППЗ – прикладне програмне забезпечення;
– ПЗ – програмне забезпечення;
– ОС – операційна система;
– БД – база даних;
– ABI (Application Binary Interface) – двійковий інтерфейс додатків для взаємодії з контрактами;
– AES (Advanced Encryption Standard) – стандарт симетричного блокового шифрування;
– API (Application Programming Interface) – програмний інтерфейс додатків;
– BIP-39 (Bitcoin Improvement Proposal 39) – стандарт мнемонічних фраз для генерації приватних ключів;
– CDN (Content Delivery Network) – мережа доставки вмісту;
– CSS (Cascading Style Sheets) – каскадні таблиці стилів;
– DEX (Decentralized Exchange) – децентралізована біржа;
– ECDSA (Elliptic Curve Digital Signature Algorithm) – алгоритм цифрового підпису на еліптичних кривих;
– EIP-1559 – стандарт ринку газу мережі Ethereum (Type-2 transactions);
– ERC-20 – стандарт взаємозамінних токенів у мережі Ethereum;
– ERC-721 – стандарт невзаємозамінних токенів (NFT);
– ERC-1155 – стандарт мультитокенів (поєднує властивості ERC-20 і ERC-721);
– ETH (Ether) – нативна криптовалюта мережі Ethereum;
– EVM (Ethereum Virtual Machine) – віртуальна машина Ethereum;

– HTML (HyperText Markup Language) – мова розмітки гіпертекстових документів;
– HTTPS (HyperText Transfer Protocol Secure) – захищений протокол передачі даних;
– ICO (Initial Coin Offering) – первинне розміщення монет;
– IPFS (InterPlanetary File System) – децентралізована файлова система для зберігання метаданих NFT;
– JSON (JavaScript Object Notation) – текстовий формат обміну даними;
– JSON-RPC – протокол віддалених викликів процедур у форматі JSON;
– Mainnet – основна (виробнича) мережа блокчейну;
– MATIC – нативна криптовалюта мережі Polygon;
– NFT (Non-Fungible Token) – невзаємозамінний токен;
– PoS (Proof-of-Stake) – алгоритм консенсусу «доказ частки»;
– PoW (Proof-of-Work) – алгоритм консенсусу «доказ роботи»;
– RPC (Remote Procedure Call) – віддалений виклик процедур;
– SDK (Software Development Kit) – набір засобів розробки;
– Sepolia – тестова мережа Ethereum для розробників;
– SHA-256 – криптографічна геш-функція;
– SPA (Single-Page Application) – односторінковий веб-застосунок;
– SVG (Scalable Vector Graphics) – масштабована векторна графіка;
– UI / UX (User Interface / User Experience) – інтерфейс / досвід користувача;
– URI (Uniform Resource Identifier) – уніфікований ідентифікатор ресурсу;
– USD (United States Dollar) – долар США;
– USDT (Tether USD) – стейблкоїн, прив’язаний до долара США;
– UTXO (Unspent Transaction Output) – невитрачений вихід транзакції;
– Web3 – концепція децентралізованого вебу на основі блокчейну.
– SPV (Simplified Payment Verification) – спрощена верифікація платежів.
– CAP – теорема про обмеження розподілених систем;

ВСТУП

Технологія блокчейн на сьогодні є однією з найдинамічніших галузей у сфері інформаційних технологій. З моменту запуску мережі Bitcoin у січні 2009 року вона еволюціонувала від простого розподіленого реєстру для електронної готівки до повноцінних обчислювальних платформ, здатних виконувати програмний код. Ключовою віхою стало розгортання у 2015 році мережі Ethereum, яка вперше дозволила створювати на блокчейні смарт-контракти – самовиконувані програми, що зберігаються в розподіленому реєстрі та виконуються віртуальною машиною EVM.

Поява смарт-контрактів відкрила можливості для створення цифрових активів нового типу – невзаємозамінних токенів (NFT) стандартів ERC-721 та ERC-1155. На відміну від взаємозамінних активів (ERC-20), кожен NFT є унікальним і може представляти права власності на цифрові або фізичні об’єкти: твори мистецтва, ігрові предмети, документи, нерухомість тощо. За даними аналітичних агенцій, ринок NFT досяг капіталізації у мільярди доларів США, що обумовлює потребу в зручних і безпечних інструментах для управління такими активами.

Центральним інструментом взаємодії користувача з блокчейном є криптовалютний гаманець. Гаманець дозволяє генерувати та зберігати приватні ключі, формувати й підписувати транзакції, переглядати баланси активів та історію операцій. Принципово гаманці поділяють на кастодіальні (приватні ключі зберігаються у третьої сторони – наприклад, біржі) та некастодіальні (ключі контролює виключно користувач). З погляду безпеки та принципу децентралізації.

Більшість сучасних гаманців реалізовані як розширення браузера або мобільні застосунки. Однак існують специфічні вимоги, які не завжди задовольняються готовими рішеннями: підтримка кількох мереж без перезавантаження, інтегрована робота з NFT-колекціями включно із завантаженням метаданих через IPFS, відображення ринкових цін, динамічний

розрахунок комісії за стандартом EIP-1559, повна відкритість вихідного коду. Тому розробка власного гаманця з гнучкою архітектурою є актуальним практичним завданням.

Актуальність роботи зумовлена такими чинниками: зростанням ринку децентралізованих фінансових інструментів і цифрових активів; необхідністю забезпечення некастодіального зберігання приватних ключів із сильним криптографічним захистом; відсутністю інтегрованих рішень, що поєднують управління криптовалютами, NFT та ринковою аналітикою в одному застосунку з відкритим вихідним кодом; потребою у формуванні практичних компетенцій студентів-фахівців з комп'ютерних наук у галузі розробки децентралізованих застосунків.

Мета бакалаврської кваліфікаційної роботи – розробка некастодіального браузерного блокчейн-гаманця з підтримкою кількох мереж стандарту EVM та функціоналом перегляду й передачі NFT, що забезпечує безпечне зберігання приватних ключів, виконання транзакцій та зручний інтерфейс користувача.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) провести системний аналіз предметної області, оглянути існуючі рішення (MetaMask, Trust Wallet, Rainbow, Phantom) та визначити переваги й недоліки;
2) побудувати UML-моделі предметної області – діаграми прецедентів, послідовності та діяльності;
3) спроєктувати інформаційне забезпечення: побудувати логічну та фізичну моделі даних, обрати систему управління базою даних;
4) спроєктувати архітектуру прикладного програмного забезпечення: визначити організаційну структуру за допомогою діаграми пакетів і розгортання;
5) обґрунтувати вибір технологічного стеку, мов програмування та середовища розробки;

6) реалізувати функціональні модулі гаманця: генерацію ключів, шифрування, відправку транзакцій, відображення NFT, ринкових даних та портфеля;
7) провести тестування системи, визначити вимоги до апаратно-програмного забезпечення та сформувати інсталяційний пакет.

Об’єктом дослідження є процеси взаємодії користувача з блокчейн-мережами стандарту EVM, що включають генерацію криптографічних ключів, формування транзакцій, перегляд і передачу цифрових активів.

Предмет дослідження – методи та програмні засоби побудови некастодіального браузерного гаманця з підтримкою стандартів NFT та мультимережевості.

Методи дослідження. У роботі використано системний аналіз для дослідження предметної області, методи об’єктно-орієнтованого моделювання UML (Unified Modeling Language) для побудови концептуальних моделей, методи проєктування реляційних баз даних, методи криптографічного захисту інформації (симетричне шифрування AES, асиметрична криптографія на еліптичних кривих ECDSA), методи компонентного програмування на основі бібліотеки React.

Практичне значення одержаних результатів полягає у створенні працездатного програмного продукту – гаманця NexaVault, який може бути використаний як кінцевими користувачами для управління криптоактивами, так і викладачами та студентами при вивченні дисциплін, пов’язаних із блокчейн-технологіями.

Структура та обсяг роботи. Пояснювальна записка складається зі вступу, чотирьох розділів основної частини, висновків, списку використаних джерел та додатків. Загальний обсяг роботи становить 68 сторінок (включно з додатками), у тому числі ілюстрацій –15, таблиць – 8, додатків – 2. Список використаних джерел містить близько 26 найменувань.

У першому розділі проведено системний аналіз предметної області, постановку завдання, огляд існуючих криптогаманців та побудовано UML-

моделі. У другому розділі описано інформаційне забезпечення системи: логічну і фізичну моделі даних, обґрунтування вибору СКБД SQLite та механізми збереження даних. Третій розділ присвячено прикладному програмному забезпеченню: організаційній структурі, обґрунтуванню вибору технологічного стеку та алгоритмізації ключових модулів. Четвертий розділ містить рекомендації щодо впровадження: процедуру тестування, апаратно-програмні вимоги та опис інсталяційного пакета.

Розроблюваний продукт NexaVault поєднує підхід maximum security з підходом maximum usability: усі криптографічні операції виконуються згідно з галузевими стандартами (BIP-39, Web3 Secret Storage), а інтерфейс залишається інтуїтивним навіть для користувачів-початківців.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

Метою розробки є створення програмного продукту – некастодіального браузерного блокчейн-гаманця з робочою назвою «NexaVault», який забезпечує користувачам можливість безпечно зберігати приватні ключі, відстежувати баланси нативної криптовалюти та токенів стандарту ERC-20, а також переглядати, відправляти та одержувати NFT стандартів ERC-721 і ERC-1155.

Система має задовольняти такі функціональні вимоги:

1) створення нового гаманця з автоматичною генерацією мнемонічної фрази за стандартом BIP-39 (12 або 24 слова);
2) імпорт існуючого гаманця трьома способами – за мнемонічною фразою, приватним ключем у форматі hex та зашифрованим Keystore-файлом JSON;
3) шифрування приватного ключа паролем користувача за алгоритмом AES відповідно до стандарту Web3 Secret Storage;
4) локальне збереження зашифрованого гаманця у браузерному сховищі LocalStorage із можливістю розблокування паролем;
5) перегляд балансу нативної криптовалюти (ETH або MATIC залежно від мережі);
6) перегляд балансів токенів стандарту ERC-20 із завантаженням метаданих (символ, назва, кількість десяткових розрядів, логотип);
7) відправка нативної криптовалюти з вибором рівня комісії (Low, Medium, High) за стандартом EIP-1559;
8) відправка токенів ERC-20 з автоматичним розрахунком газу;
9) перегляд NFT-галереї з підтримкою колекцій ERC-721 та ERC-1155 і завантаженням метаданих через IPFS;
10) відображення детальної інформації про NFT – атрибутів, опису, QR-коду токена;

11) передача NFT іншому користувачеві за публічною адресою;

12) перемикання між мережами Sepolia, Ethereum Mainnet та Polygon без перезавантаження застосунку;

13) відображення ринкових цін популярних криптовалют через CoinGecko API;

14) побудова графіка історії балансу за допомогою бібліотеки recharts;

15) формування QR-коду публічної адреси для отримання коштів;

16) збереження локальної історії транзакцій з можливістю переходу на блокчейн-експлорер (Etherscan, Polygonscan).

Класифікація розроблюваної системи. За характером використання інформації – це система обробки фінансових транзакцій з елементами інформаційно-пошукової системи (для роботи з NFT). За масштабом – настільний веб-застосунок, призначений для індивідуального використання. За підтримуваними стандартами комунікації – застосунок, що використовує протокол HTTPS поверх TCP/IP та віддалені виклики процедур у форматі JSON-RPC для взаємодії з блокчейн-нодами. За ступенем автоматизації – повністю автоматизована система, що не потребує втручання оператора, окрім вводу паролю та підтвердження транзакцій з боку власника гаманця.

Вхідна інформація системи:

– мнемонічна фраза для відновлення гаманця (під час імпорту);

– пароль для шифрування та розблокування приватного ключа;

– публічні адреси отримувачів транзакцій (40-символьні шістнадцяткові рядки);

– суми переказів криптовалюти або кількість токенів;

– адреси смарт-контрактів токенів ERC-20 та колекцій NFT;

– обрана блокчейн-мережа та рівень пріоритету газу;

– конфігураційні параметри (RPC-вузол Alchemy).

Вихідна інформація системи:

– зашифрований Keystore JSON, що зберігається у LocalStorage;

– хеш-сум транзакцій, опублікованих у блокчейн-мережі;

– візуальне представлення балансу та портфеля у вигляді карток і графіків;
– NFT-галерея з мініатюрами зображень, що завантажуються з IPFS;
– таблиця ринкових цін криптовалют, оновлювана у реальному часі;
– історія транзакцій із посиланнями на блокчейн-експлорер;
– QR-код адреси для зручного отримання коштів.

1.2 Огляд інформаційних джерел та існуючих рішень

Сучасний ринок криптогаманців включає десятки готових рішень, які можна класифікувати за способом зберігання ключів (гарячі/холодні), за способом доставки (браузерні розширення, мобільні застосунки, апаратні пристрої) та за кастодіальністю (некастодіальні і кастодіальні). Розглянемо ключових представників, що мають вплив на формування вимог до розробки.

1.2.1 MetaMask

MetaMask – найбільш відомий некомерційний гаманець у вигляді розширення для браузерів Chrome, Firefox, Edge та Brave. Розроблений компанією ConsenSys, MetaMask використовує бібліотеку ethers.js та має понад 30 мільйонів активних користувачів станом на 2024 рік. Підтримує всі EVM-мережі через додавання користувачем RPC-URL.

Переваги:
– широкий список попередньо налаштованих мереж;
– наявність мобільної версії з кросплатформовою синхронізацією;
– підтримка апаратних гаманців Ledger і Trezor.

Недоліки:
– громіздкий інтерфейс із надлишковими функціями (Swap, Bridge, Staking) – для базових операцій потрібно багато кліків;
– обмежена візуалізація NFT-колекцій (фактично відсутня окрема галерея);
– висока вартість Swap-операцій через комісію провайдера;
– конфіденційність – за замовчуванням використовується вузол Infura, що належить ConsenSys, який збирає телеметрію.

1.2.2 Trust Wallet

Trust Wallet – мобільний гаманець, придбаний криптобіржею Binance у 2018 році. Підтримує понад 70 блокчейнів.

Переваги: чудова мобільна підтримка; вбудована NFT-галерея; широкий вибір мереж.

Недоліки: відсутність повноцінної десктоп- або веб-версії; залежність від екосистеми Binance; обмежений контроль над RPC-вузлами.

1.2.3 Rainbow Wallet

Rainbow – мобільний гаманець з акцентом на користувацький досвід і дизайн. Зорієнтований переважно на Ethereum-екосистему та NFT.

Переваги: найкращий у класі дизайн; чудова NFT-галерея з підтримкою OpenSea API; відображення Floor Price колекцій.

Недоліки: тільки мобільна версія; обмежена кількість підтримуваних мереж (переважно Ethereum L2).

1.2.4 Phantom

Phantom – гаманець для мережі Solana, у 2024 році розширив підтримку до Ethereum і Polygon.

Переваги: блискавична швидкість роботи; інтуїтивний UI; швидкий перегляд NFT.

Недоліки: відсутність відкритого коду; обмежений функціонал для EVM-мереж порівняно з MetaMask.

1.2.5 Аналіз і висновки за оглядом

У таблиці 1.1 наведено порівняльний аналіз розглянутих рішень за ключовими критеріями.

Таблиця 1.1

Порівняльна характеристика існуючих рішень

Гаманець	**Платформа**	**Мережі**	**NFT-галерея**	**Відкритий код**
MetaMask	Браузер, мобільний	EVM-сумісні	Базова	Частково
Trust Wallet	Мобільний	70+	Так	Ні
Rainbow	Мобільний	Ethereum L2	Так, преміум	Так
Phantom	Браузер, мобільний	Solana, EVM	Так	Ні
NexaVault (розробка)	Веб (браузер)	Sepolia, ETH, Polygon	Так, повна	Так

За результатами аналізу сформульовано позиціонування розроблюваного гаманця NexaVault. Він орієнтований на користувача-розробника або просунутого користувача, який потребує:

– відкритого вихідного коду і можливості повного аудиту безпеки;

– браузерної (а не розширенням) версії, що працює на будь-якому пристрої без встановлення;

– повноцінної NFT-галереї з мінімальною кількістю кліків;

– швидкого перемикання між тестовою і виробничою мережами для розробки;

– виразного дизайну в естетиці cyberpunk;

– відсутності залежності від однієї криптобіржі чи централізованого провайдера.

1.2.6 Теоретичні засади блокчейн-технології

Для глибшого розуміння функціонування розроблюваної системи розглянемо ключові поняття та механізми технології блокчейн, на яких ґрунтується робота будь-якого криптогаманця.

Блокчейн (англ. blockchain) – це розподілена база даних, у якій зберігається інформація про кожну транзакцію, створену в системі. Особливістю є те, що вона одночасно зберігається на безлічі комп'ютерів, з'єднаних через мережу Інтернет. Дані зберігаються у вигляді ланцюжка блоків із записами про транзакції. Кожен новий запис здійснює підтвердження в уже існуючих ланцюжках, тому підробити дані неможливо: для цього потрібно змінити інформацію у всіх копіях блоків у всіх вузлах мережі одночасно.

Структура блоку складається з заголовка (Header), у якому зберігається службова інформація, та корисного навантаження (Payload) – записів транзакцій. У заголовку блоку міститься: версія блоку; дата і час створення; хеш заголовка поточного блоку; хеш заголовка попереднього блоку; корінь дерева Меркла транзакцій; параметри nonce і bits (для PoW-мереж). Саме хеш попереднього блоку, записаний у заголовку наступного, утворює ланцюжок, що дав назву технології.

Дерево Меркла (Merkle Tree), розроблене Р. Мерклом у 1979 р., є структурою даних, що дозволяє перетворити будь-який масив даних на рядок фіксованого розміру (корінь дерева). У блокчейні воно використовується для верифікації цілісності транзакцій усередині блоку та для роботи «легких» (SPV) клієнтів, що зберігають лише заголовки блоків, а не повний їх вміст. Власне, мобільні криптогаманці (включно з розроблюваним) є фактично SPV-клієнтами – вони не зберігають увесь блокчейн, а покладаються на RPC-вузли для верифікації.

Транзакція у блокчейні Bitcoin має модель UTXO (Unspent Transaction Output): нова транзакція через входи (Inputs) посилається на виходи (Outputs) попередніх транзакцій і формує нові виходи для майбутнього використання. Натомість Ethereum використовує модель Account-based: у системі

підтримується глобальний стан з балансами всіх адрес, і транзакція просто змінює баланс рахунків. Це принципова різниця, яку важливо розуміти при розробці EVM-гаманця.

Платформа Ethereum, запропонована В. Бутеріним у 2014 році та запущена в липні 2015 року. Внутрішня валюта платформи – Ether (ETH) – застосовується не лише як розрахункова одиниця, а й як «пальне» (gas) для виконання смарт-контрактів. На відміну від Bitcoin, який забезпечує лише фінансові транзакції, Ethereum пропонує тьюрінг-повну середовище виконання – Ethereum Virtual Machine (EVM).

Облікові записи в Ethereum поділяються на два типи: зовнішні (Externally Owned Accounts – EOA), контрольовані приватними ключами; та контрактні (Contract Accounts), контрольовані власним кодом. Стан кожного з облікових записів характеризується чотирма значеннями: nonce (кількість надісланих транзакцій), balance (баланс у Wei), storageRoot (хеш дерева збереження), codeHash (хеш EVM-коду – для EOA він дорівнює хешу порожнього рядка).

Газ (gas) – одиниця виміру обчислювальної роботи в Ethereum. Ціна газу (gasPrice) – кількість Wei, що сплачується за одну одиницю. Ліміт газу (gasLimit) – максимальна кількість, яку відправник готовий витратити. У 2021 році з оновленням London Hard Fork було запроваджено стандарт EIP-1559, що ввів динамічне ціноутворення з двома компонентами: baseFee (мінімальна обов’язкова ціна, що спалюється) та priorityFee (чайові майнерам/валідаторам). Саме цей стандарт використано в розроблюваному гаманці для формування транзакцій типу 2.

Виокремлюють публічні та приватні блокчейни. Публічні (як Ethereum) не мають центрального органу контролю – кожен може приєднатися до мережі, переглянути дані, створити транзакцію. Приватні (як R3 Corda, Hyperledger Fabric) накладають обмеження на участь і використовуються переважно в корпоративних рішеннях. Розроблюваний гаманець взаємодіє виключно з публічними блокчейнами.

Альтернативні мережі (альткоїни) також можуть бути EVM-сумісними, що дозволяє переносити смарт-контракти Ethereum в інші середовища без зміни коду. Polygon (раніше Matic Network), Binance Smart Chain, Avalanche C-Chain, Arbitrum, Optimism – усі EVM-сумісні і підтримуються розробленим гаманцем шляхом простої зміни RPC-вузла. Це є одним із ключових архітектурних рішень – використання EVM-сумісності для забезпечення мультимережевості без принципових змін у коді.

1.2.7 Огляд літературних джерел

Теоретичною основою роботи стали наукові праці та технічні документи в галузі блокчейн-технологій. Базовим джерелом є «біла книга» Bitcoin за авторством С. Накамото [1], яка ввела поняття однорангової електронної готівки на основі ланцюжка блоків. Технічні аспекти мережі Ethereum розкриті у Yellow Paper Г. Вуда [2] та документації віртуальної машини EVM.

Стандарти токенів детально описано в матеріалах Ethereum Foundation: ERC-20 (EIP-20), ERC-721 (EIP-721) та ERC-1155 (EIP-1155). Стандарт мнемонічних фраз BIP-39 та ієрархічно детермінованих гаманців BIP-32 / BIP-44 описано у відповідних BIP-документах Bitcoin Core.

Питання криптографії на еліптичних кривих (secp256k1, ECDSA), що використовується для підписання транзакцій в Ethereum, розглянуто у працях Н. Коблиця та В. Міллера [1] Алгоритм AES, який використано для шифрування Keystore-файлів, специфіковано в стандарті FIPS PUB 197.

1.3 Моделювання предметної області засобами UML

Для формального опису предметної області та поведінки системи побудовано три UML-діаграми: діаграму прецедентів, діаграму послідовності та діаграму діяльності. Унікальна мова моделювання UML (Unified Modeling Language) є галузевим стандартом для проєктування об’єктно-орієнтованих програмних систем.

1.3.1 Діаграма прецедентів

Діаграма прецедентів (Use Case Diagram) відображає функціональність системи з погляду користувача. Вона показує, які саме дії може виконувати актор (Actor) та як система взаємодіє із зовнішнім середовищем – у даному випадку з блокчейн-мережею.

На діаграмі (рис. 1.1) показано основного актора – Користувача гаманця, а також зовнішню систему – Блокчейн-мережа, яка валідує всі дії, пов’язані з фінансами. Виділено такі прецеденти:

1) Створення та імпорт гаманця – початковий етап генерації або відновлення приватних ключів;
2) Перегляд балансу та NFT – функції читання даних із блокчейну;
3) Передача NFT – сценарій, що включає обов’язкову підфункцію «Підпис транзакції» (зв’язок include);
4) Перемикання мережі – користувач може змінювати активну блокчейн-мережу;
5) Перегляд ринкових цін – взаємодія із зовнішнім сервісом CoinGecko.

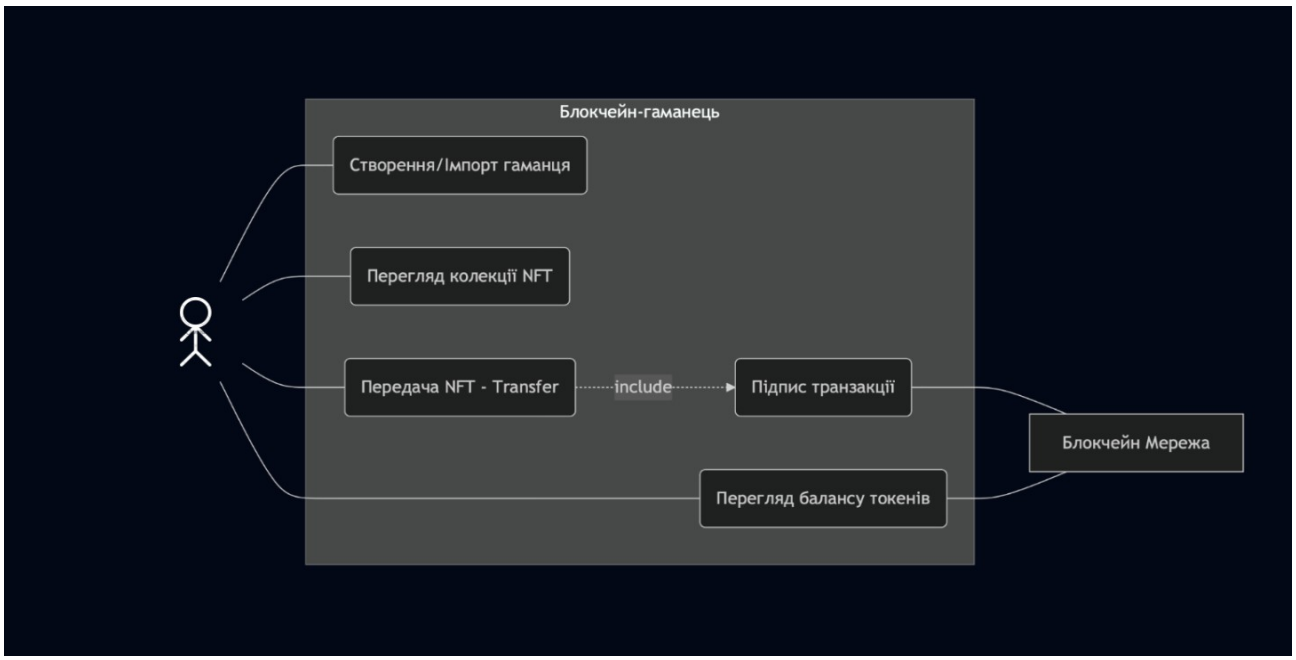

Рис. 1.1 – Діаграма прецедентів системи NexaVault

Зв’язок типу include використовується там, де один прецедент обов’язково включає виконання іншого: так, «Передача NFT» обов’язково включає «Підпис транзакції» приватним ключем. Зв’язок extend (розширення) застосовано до прецеденту «Перегляд балансу», який може бути розширений до «Перегляду історії транзакцій» за бажанням користувача.

1.3.2 Діаграма послідовності

Діаграма послідовності (Sequence Diagram) ілюструє взаємодію об’єктів у часі. У ролі такого сценарію обрано «Передачу NFT від одного власника до іншого», оскільки цей процес є найбільш репрезентативним і охоплює всі рівні архітектури – від інтерфейсу до блокчейн-вузла.

Послідовність дій (рис. 1.2):

1) Користувач ініціює дію через інтерфейс (натискає кнопку «Send NFT»);
2) Менеджер транзакцій формує пакет даних для смарт-контракту (виклик функції safeTransferFrom);
3) Користувач підписує цей пакет приватним ключем (виклик ethers.Wallet.signTransaction);
4) Програма транслює підписаний хеш у блокчейн-вузол через JSON-RPC (метод eth_sendRawTransaction);

5) Блокчейн-вузол повертає хеш транзакції;

6) Система очікує підтвердження від мережі (включення в блок) та оновлює стан інтерфейсу – показує статус «Confirmed» з посиланням на Etherscan.

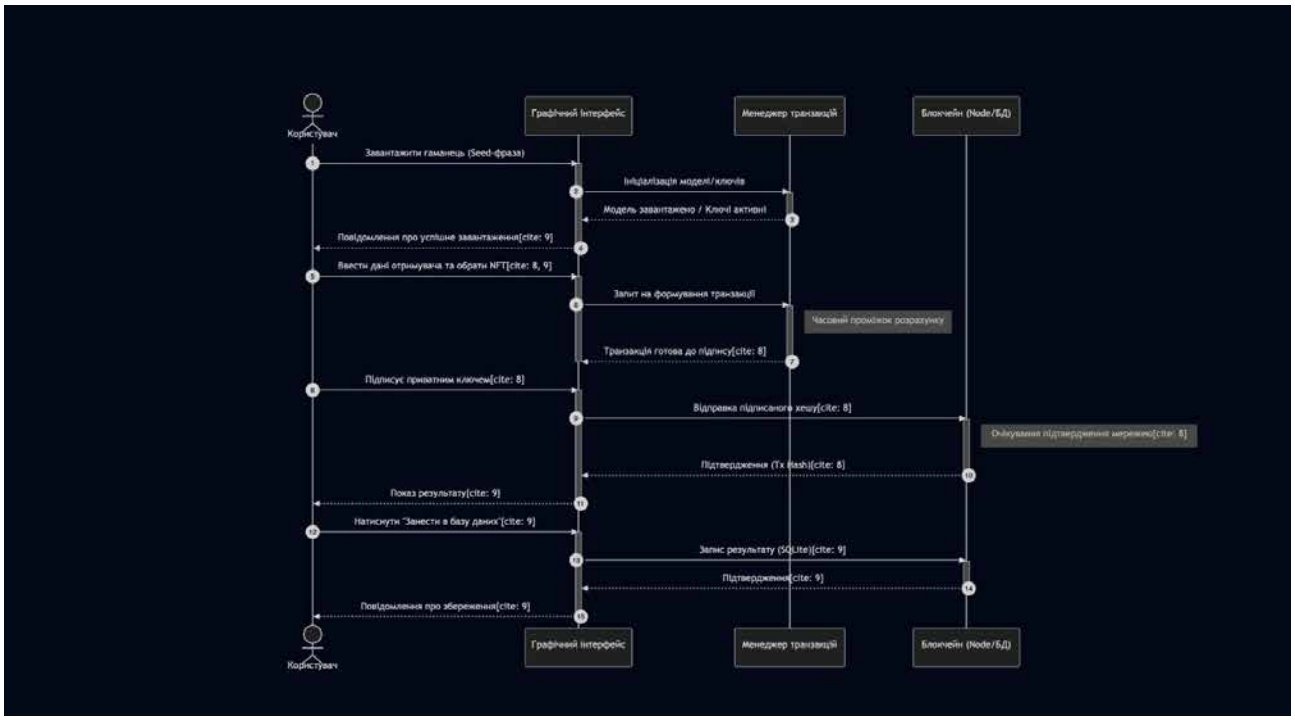

Рис. 1.2 – Діаграма послідовності передачі NFT

1.3.3 Діаграма діяльності

Діаграма діяльності (Activity Diagram) відображає алгоритмічну поведінку системи. У роботі побудовано діаграму процесу авторизації та оновлення даних, який запускається при кожному відкритті застосунку (рис. 1.3).

Алгоритм:

1) Процес починається з перевірки локального пароля (розшифрування Keystore JSON);
2) У разі успіху запускається паралельний процес синхронізації: запит балансу нативної криптовалюти, балансів ERC-20 токенів та завантаження метаданих NFT через IPFS;
3) Паралельно ініціюється запит ринкових цін до CoinGecko;
4) У разі помилки розшифрування – повернення до екрану розблокування з повідомленням про невірний пароль;

5) Завершення – перехід у стан повної готовності головного екрана (Dashboard).

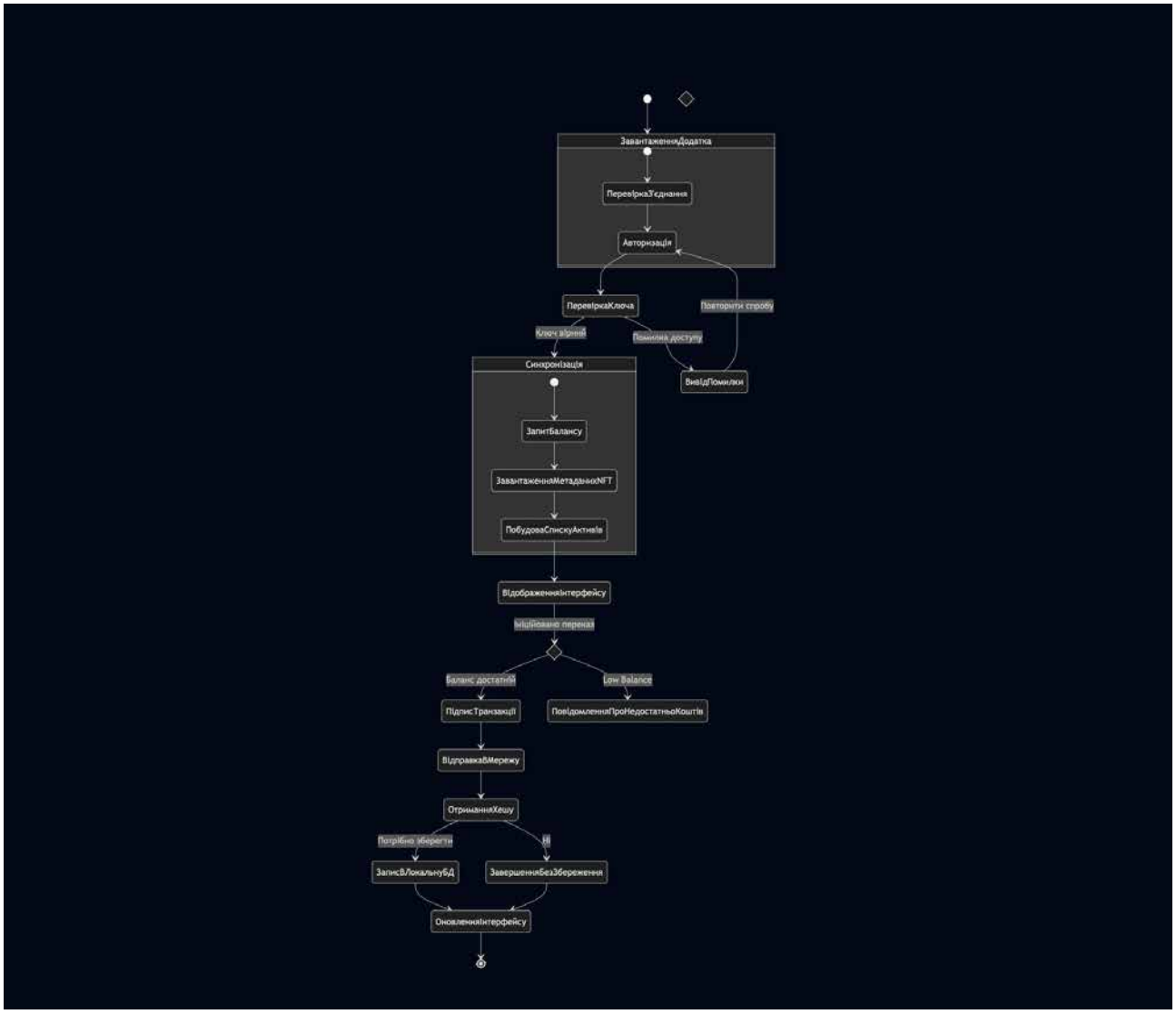

Рис. 1.3 – Діаграма діяльності процесу авторизації

Побудовані UML-моделі дозволяють здійснити перехід від абстрактного опису функціональних вимог до конкретного проєктування інформаційного і прикладного програмного забезпечення, чому присвячено наступні розділи записки.

Висновки до розділу 1. У першому розділі проведено системний аналіз предметної області, у результаті якого: сформульовано постановку завдання з переліком функціональних вимог; здійснено огляд існуючих рішень (MetaMask, Trust Wallet, Rainbow, Phantom) і встановлено нішу для розроблюваного

продукту – браузерний гаманець з відкритим кодом і повноцінною NFT-галереєю; розглянуто теоретичні засади блокчейн-технології, ключові поняття та стандарти (BIP-39, EIP-1559, ERC-20/721/1155), що формують основу для розробки; побудовано три ключові UML-діаграми (прецедентів, послідовності, діяльності), які формалізують функціональність, поведінку та алгоритм роботи системи. Сформовані моделі є основою для подальшого проєктування інформаційного та прикладного забезпечення, чому присвячені наступні розділи записки.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Інформаційне забезпечення системи становить сукупність даних, організованих певним чином, і програмних засобів роботи з ними. У випадку блокчейн-гаманця інформаційна база має складну структуру: частина даних є глобальними і зберігаються у блокчейн-реєстрі (баланси, транзакції, NFT), а частина – локальна, специфічна для конкретного користувача (зашифрований ключ, історія активності, кеш метаданих). Тому проєктування інформаційного забезпечення починається з виокремлення сутностей, які мають зберігатися саме у локальній базі даних.

Логічна модель даних описує об’єкти предметної області (сутності) та зв’язки між ними у незалежний від конкретної СКБД спосіб. Для побудови моделі використано нотацію ER-діаграм (Entity-Relationship), оскільки вона є найбільш поширеною для проєктування реляційних баз даних.

На основі аналізу функціональних вимог виділено п’ять ключових сутностей, які наведені в таблиці 2.1.

Таблиця 2.1

Сутності інформаційної бази та їх атрибути

Сутність	**Тип**	**Ключові атрибути**
Wallet	Сильна	Public_Address (PK), Encrypted_Key, Mnemonic_Hash
Transaction	Залежна	Tx_Hash (PK), Amount, Timestamp, Wallet_ID (FK)
Asset	Сильна	Contract_Address (PK),

		Symbol, Decimals
NFT_Collection	Сильна	Collection_Addr (PK), Name, Floor_Price
NFT_Item	Залежна	Token_ID (PK), Metadata_URI, Collection_ID (FK), Owner_ID (FK)

Семантика виділених сутностей є такою:

– Wallet – гаманець користувача, ідентифікований публічною адресою. Сильна сутність, оскільки існує самостійно і має унікальний ідентифікатор. Атрибут Encrypted_Key зберігає зашифрований приватний ключ у форматі Keystore JSON, Mnemonic_Hash – хеш мнемонічної фрази для верифікації автентичності;

– Transaction – транзакція, ініційована конкретним гаманцем. Залежна сутність: транзакція не має сенсу без посилання на ініціатора, тому атрибут Wallet_ID є зовнішнім ключем. Tx_Hash – унікальний хеш транзакції в блокчейні;

– Asset – крипто-актив (нативна валюта або токен ERC-20). Сильна сутність, ідентифікована адресою смарт-контракту. Атрибути включають символ (USDT, USDC, BNB), назву та кількість десяткових розрядів;

– NFT_Collection – колекція невзаємозамінних токенів. Сильна сутність із базовою інформацією про колекцію та її поточну Floor Price (мінімальну ціну продажу);

– NFT_Item – конкретний NFT всередині колекції. Залежна сутність із двома зовнішніми ключами: на колекцію (Collection_ID) та на гаманець-власник (Owner_ID).

Зв’язки між сутностями описані нижче (рис. 2.1).

– Wallet – Transaction: тип «один-до-багатьох». Один гаманець може ініціювати безліч транзакцій. Зв’язок ідентифікувальний, оскільки транзакція не має сенсу без гаманця відправника;

– NFT_Collection – NFT_Item: тип «один-до-багатьох». Кожна колекція містить багато окремих токенів;

– Wallet – NFT_Item: тип «один-до-багатьох». Один гаманець може володіти багатьма NFT. Атрибут Owner_ID мігрував з сутності Wallet;

– Wallet – Asset: тип «багато-до-багатьох». Гаманець може тримати багато різних токенів, а один тип токена може бути в багатьох гаманцях. Реалізовано через проміжну сутність Asset_Balance.

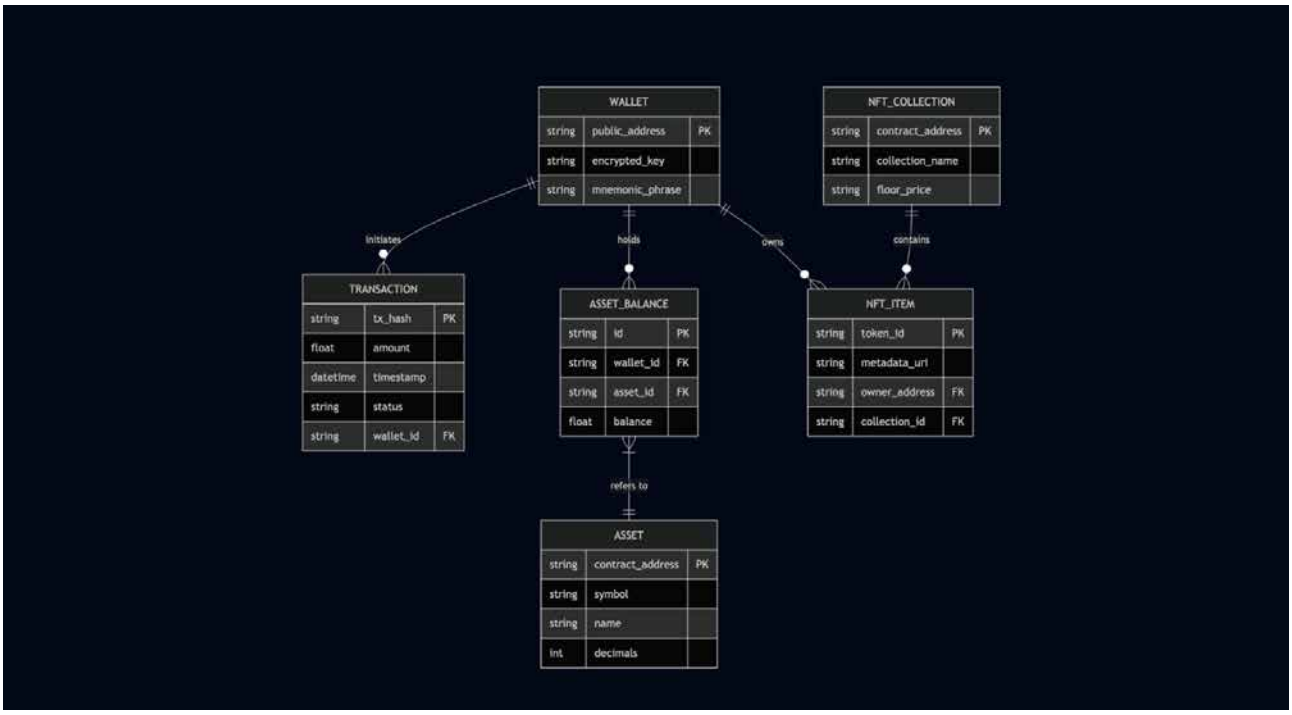

Рис. 2.1 – Логічна ER-діаграма інформаційної бази

Перевірка моделі на відповідність нормальним формам. Розроблена модель відповідає вимогам першої нормальної форми (1НФ): усі атрибути є атомарними. Друга нормальна форма (2НФ) також виконується, оскільки кожна неключова атрибутивна залежить від усього первинного ключа. Третя нормальна форма (3НФ): між неключовими атрибутами немає транзитивних залежностей. Таким чином, модель є реляційною і може бути реалізована за допомогою реляційної СКБД.

Альтернативні моделі. У ході проєктування розглядалися також документоорієнтовані рішення (MongoDB, IndexedDB) для зберігання гнучких метаданих NFT, які мають різну структуру у різних колекцій. Однак для основної бізнес-логіки гаманця реляційна модель є оптимальною завдяки:

– гарантованій підтримці ACID-властивостей фінансових транзакцій;

– декларативній мові SQL для побудови складних запитів;

– потужним механізмам цілісності (FOREIGN KEY, CHECK, TRIGGER);

– мінімальним накладним витратам на зберігання.

Для гнучких метаданих NFT використано гібридний підхід: основні поля (token_id, owner, collection) зберігаються у реляційних таблицях, тоді як повний JSON-документ метаданих кешується як TEXT-поле або як файл у локальному кеші. Це поєднує переваги обох підходів.

Додаткові сутності. Окрім п’яти основних сутностей, у фізичній моделі присутні допоміжні таблиці:

– Asset_Balances – проміжна таблиця багато-до-багатьох між Wallets і Assets, зберігає поточний баланс кожного активу для кожного гаманця;

– Networks – довідник підтримуваних мереж із RPC-URL та chainId;

– Settings – таблиця ключ-значення для збереження користувацьких налаштувань (тема, мова, поточна активна мережа).

2.2 Вибір системи управління інформаційною базою

Серед сучасних СКБД, що підтримують реляційну модель, до розгляду включено такі: MS SQL Server, Oracle Database, PostgreSQL, MySQL, MariaDB, MS Access, SQLite. Окремо проаналізовано документоорієнтовані (MongoDB) та вбудовані браузерні рішення (IndexedDB, WebSQL).

Критерії вибору СКБД для проєкту:

1) локальна робота – гаманець є клієнтським застосунком, тому СКБД повинна функціонувати на пристрої користувача без розгортання сервера;
2) мінімальні системні вимоги – не накладати додаткового навантаження на пристрій;

3) портативність – уся база даних має зберігатися в одному файлі для зручної міграції;

4) підтримка ACID-транзакцій – для гарантії цілісності фінансових даних;

5) безкоштовна ліцензія – бажано Public Domain або MIT;

6) наявність драйверів для JavaScript / Python – для можливої міграції на десктопну версію.

Порівняльний аналіз СКБД наведено в таблиці 2.2.

Таблиця 2.2

Порівняльний аналіз СКБД

СКБД	**Сервер**	**Файл**	**Ліцензія**	**Підходить**
MS SQL Server	Так	Багато	Комерційна	Ні
Oracle DB	Так	Багато	Комерційна	Ні
PostgreSQL	Так	Багато	PostgreSQL	Ні
MySQL	Так	Багато	GPL	Ні
MS Access	Ні	Один	Комерційна	Частково
SQLite	Ні	Один	Public Domain	Так

На підставі проведеного аналізу для реалізації локального сховища даних обрано СКБД SQLite. Основні переваги SQLite для проєкту:

– відсутність необхідності розгортання сервера – уся СКБД є вбудованою бібліотекою;

– уся база даних зберігається в одному файлі (wallet_history.db), що дозволяє користувачеві легко переносити свої локальні дані між пристроями;

– висока швидкість виконання SQL-запитів до локальних таблиць балансів та активів;
– підтримка ACID-транзакцій (Atomicity, Consistency, Isolation, Durability);
– підтримка динамічних типів даних (Type Affinity), що зручно для зберігання гексадецимальних рядків блокчейн-адрес;
– Public Domain ліцензія – без жодних обмежень на комерційне використання;
– розмір рушія – менше 1 МБ, не впливає на загальний розмір застосунку.

Для безпосередньо браузерної версії застосунку як аналог SQLite використано LocalStorage браузера з аналогічною логікою «ключ – значення». При переході на десктопну версію (наприклад, через Electron або Tauri) перехід на SQLite не потребує переписування бізнес-логіки – змінюється лише шар Persistence.

2.3 Створення інформаційної бази

У фізичній моделі сутності логічної моделі перетворюються на таблиці, а атрибути – на колонки з конкретними типами даних, що відповідають синтаксису обраної СКБД. SQLite має п’ять основних типів афініту даних: TEXT, INTEGER, REAL, BLOB, NUMERIC.

2.3.1 Фізична модель даних

Фізична схема, побудована за результатами перетворення логічної моделі, наведена на рис. 2.2. Конкретні типи колонок та обмеження наведено в таблиці 2.3.

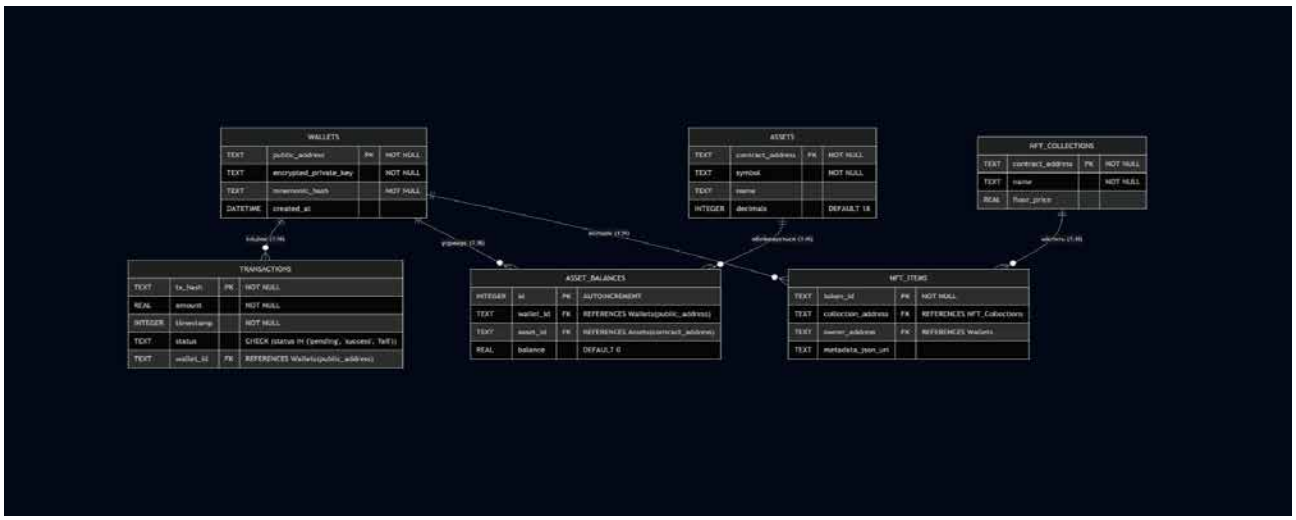

Рис. 2.2 – Фізична ER-діаграма бази даних SQLite

Таблиця 2.3

Фізична структура таблиць

Таблиця	**Колонка**	**Тип**	**Опис**
Wallets	public_address	TEXT PK	Публічна адреса гаманця (42 симв.)
Wallets	encrypted_private_key	TEXT NOT NULL	Зашифрований приватний ключ (Keystore JSON)
Wallets	mnemonic_hash	TEXT NOT NULL	Хеш мнемонічної фрази
Wallets	created_at	DATETIME	Час створення гаманця
Transactions	tx_hash	TEXT PK	Хеш транзакції (66 симв.)
Transactions	amount	REAL NOT	Сума переказу

		NULL	
Transactions	timestamp	INTEGER	Unix-час транзакції
Transactions	status	TEXT CHECK	pending / success / failed
Transactions	wallet_id	TEXT FK	Зовнішній ключ на Wallets
Assets	contract_address	TEXT PK	Адреса смарт-контракту
Assets	symbol	TEXT NOT NULL	Тікер активу (ETH, USDT)
Assets	name	TEXT	Повна назва токена
Assets	decimals	INTEGER DEFAULT 18	Кількість десяткових розрядів
NFT_Items	token_id	TEXT PK	ID токена в межах контракту
NFT_Items	metadata_json_uri	TEXT	URI метаданих NFT
NFT_Items	owner_address	TEXT FK	Адреса власника NFT
NFT_Items	collection_id	TEXT FK	Зовнішній ключ на колекцію

2.3.2 Правила валідації та значення за замовчуванням

Для забезпечення цілісності даних застосовано такі правила:

– значення за замовчуванням (DEFAULT): для поля balance у проміжних таблицях встановлено 0.0, для decimals у таблиці Assets – значення 18 (стандартна кількість десяткових для ERC-20);

– обмеження CHECK: для поля status у таблиці Transactions встановлено правило CHECK (status IN ('pending', 'success', 'failed'));

– NOT NULL: усі ключові поля та адреси мають статус NOT NULL для запобігання некоректним записам;

– UNIQUE: для пари (wallet_id, asset_id) у проміжній таблиці Asset_Balances встановлено унікальний індекс.

2.3.3 Індексація

Для оптимізації роботи гаманця створено такі індекси:

– idx_wallet_transactions на колонці wallet_id таблиці Transactions – прискорює відображення історії конкретного гаманця при відкритті вкладки «Історія»;

– idx_nft_collection на колонці collection_address – прискорює групування предметів за колекціями в NFT-галереї;

– idx_tx_timestamp на колонці timestamp – для сортування транзакцій за часом у порядку спадання.

2.3.4 SQL-скрипт створення бази даних

Нижче наведено фрагмент SQL-коду, що використовується для розгортання структури бази даних:

```
CREATE TABLE Wallets (
   public_address         TEXT PRIMARY KEY NOT NULL,
   encrypted_private_key   TEXT NOT NULL,
   mnemonic_hash          TEXT NOT NULL,
   created_at             DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE Assets (
   contract_address  TEXT PRIMARY KEY NOT NULL,
   symbol            TEXT NOT NULL,
   name              TEXT,
   decimals          INTEGER DEFAULT 18
);

CREATE TABLE Transactions (
   tx_hash      TEXT PRIMARY KEY NOT NULL,
   amount       REAL NOT NULL,
   timestamp    INTEGER NOT NULL,
   status       TEXT CHECK (status IN ('pending','success','failed')),
   wallet_id    TEXT,
   FOREIGN KEY (wallet_id) REFERENCES Wallets(public_address)
);

CREATE INDEX idx_wallet_transactions ON Transactions(wallet_id);
CREATE INDEX idx_tx_timestamp        ON Transactions(timestamp);
```

2.3.5 Тригери для контролю цілісності

Оскільки SQLite не підтримує збережені процедури в класичному розумінні (як MS SQL Server), логіка валідації та підтримки посилальної цілісності винесена в тригери (TRIGGER). Розроблено чотири ключові тригери.

1) Тригер перевірки невід’ємності балансу. Запобігає встановленню від’ємного балансу, що є критичним для фінансової системи:

```
CREATE TRIGGER trg_check_positive_balance
BEFORE UPDATE ON Asset_Balances
FOR EACH ROW
BEGIN
   SELECT CASE
      WHEN NEW.balance < 0 THEN
         RAISE(ABORT, 'Помилка: Баланс не може бути від’ємним')
   END;
```

```
END;
```

2) Тригер каскадного видалення гаманця. Якщо видаляється запис із таблиці Wallets, автоматично видаляються всі пов’язані баланси та NFT:

```
CREATE TRIGGER trg_cleanup_wallet_data
AFTER DELETE ON Wallets
FOR EACH ROW
BEGIN
  DELETE FROM Asset_Balances WHERE wallet_id = OLD.public_address;
  DELETE FROM NFT_Items      WHERE owner_address = OLD.public_address;
END;
```

3) Тригер валідації суми транзакції. Перевіряє, що сума переказу є додатною:

```
CREATE TRIGGER trg_validate_transaction_amount
BEFORE INSERT ON Transactions
FOR EACH ROW
BEGIN
  SELECT CASE
    WHEN NEW.amount <= 0 THEN
      RAISE(ABORT, 'Сума транзакції має бути більшою за нуль')
  END;
END;
```

4) Тригер перевірки володіння NFT. Перед записом транзакції типу NFT_TRANSFER перевіряє, чи дійсно гаманець-відправник є власником NFT:

```
CREATE TRIGGER trg_check_nft_ownership
BEFORE INSERT ON Transactions
FOR EACH ROW WHEN NEW.tx_type = 'NFT_TRANSFER'
BEGIN
  SELECT CASE
    WHEN (SELECT owner_address FROM NFT_Items
          WHERE token_id = NEW.token_id) != NEW.wallet_id
    THEN RAISE(ABORT, 'Ви не є власником цього NFT')
  END;
END;
```

2.3.6 Заповнення базовими даними

Умовно-постійна інформація (підтримувані активи, відомі NFT-колекції) завантажується в базу один раз при інсталяції. Це дозволяє гаманцю одразу відображати релевантні дані без додаткових мережевих запитів:

```
INSERT INTO Assets (contract_address, symbol, name, decimals) VALUES
('native_eth',                    'ETH',  'Ethereum', 18),
('0xdAC17F958D2ee523a2206206994597C13D831ec7', 'USDT', 'Tether USD', 6),
('0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48', 'USDC', 'USD Coin',  6);

INSERT INTO NFT_Collections (contract_address, name, floor_price) VALUES
('0xbc4ca0eda7647a8ab7c2061c2e118a18a936f13d', 'Bored Ape YC', 12.5),
('0xb47e3cd837ddf8e4c57f05d70ab865de6e193bbb', 'CryptoPunks',  25.0);
```

2.3.7 Подання (Views)

Для зручного отримання агрегованих даних створено віртуальне подання View_TotalPortfolioValue, яке обчислює сумарну вартість усіх активів кожного гаманця в USD:

```
CREATE VIEW View_TotalPortfolioValue AS
SELECT  w.public_address,
      SUM(ab.balance * a.price_usd) AS total_usd_value
FROM    Wallets w
JOIN    Asset_Balances ab ON w.public_address = ab.wallet_id
JOIN    Assets a          ON ab.asset_id = a.contract_address
GROUP BY w.public_address;
```

Висновки до розділу 2. У другому розділі спроєктовано інформаційне забезпечення системи: побудовано логічну ER-модель з п’ятьма ключовими сутностями, виконано аналіз нормальних форм і доведено реляційність моделі. На підставі порівняння СКБД (MS SQL Server, Oracle, PostgreSQL, MySQL, MS Access, SQLite) обрано вбудовану СКБД SQLite як таку, що найкраще відповідає вимогам локального некастодіального гаманця. Розроблено фізичну схему даних із застосуванням обмежень NOT NULL, CHECK, DEFAULT, створено індекси

для оптимізації пошуку, а також тригери для контролю цілісності фінансових даних і запобігання некоректним операціям.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Прикладне програмне забезпечення (ППЗ) гаманця NexaVault побудоване за багатошаровою архітектурою. Кожен шар має чітко визначену зону відповідальності, що забезпечує модульність, тестування та можливість незалежної заміни компонентів. Архітектурне рішення представлено у вигляді UML-діаграми пакетів (рис. 3.1).

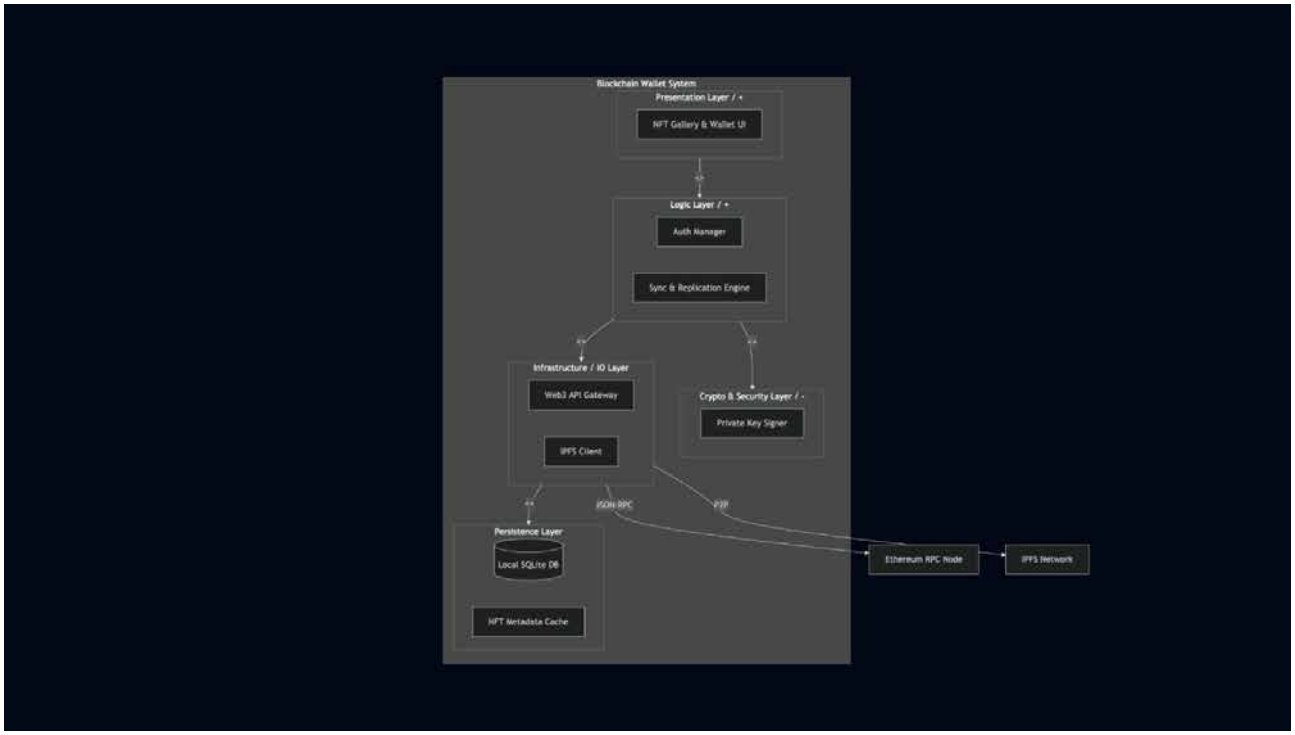

Рис. 3.1 – Діаграма пакетів програмного забезпечення NexaVault

Призначення шарів:

3.1.1 Шар представлення (Presentation Layer)

Видимість пакета – публічна (+). Відповідає за графічний інтерфейс користувача. Містить React-компоненти візуалізації – NFT-галерею, портфоліо, форми створення транзакцій, налаштування мережі. Цей шар не містить бізнес-логіки і не звертається напряму до блокчейну – він лише викликає сервіси нижніх шарів і відображає отримані дані. Реалізація цього шару базується на бібліотеці React 18 та фреймворку Tailwind CSS.

3.1.2 Шар логіки (Logic Layer)

Видимість пакета – публічна (+). Реалізує бізнес-логіку гаманця: перевірку формату блокчейн-адрес (валідація 42-символьної геш-стрічки), розрахунок комісії за стандартом EIP-1559, обробку вхідних даних від користувача, форматування сум з урахуванням decimals токена, конвертацію валют. Сюди ж входить пакет Sync & Replication Engine, що координує оновлення локального кешу з даних блокчейну.

3.1.3 Шар безпеки (Crypto & Security Layer)

Видимість пакета – приватна (–). Закритий пакет, що відповідає за криптографічні операції: генерацію мнемонічних фраз за BIP-39, отримання приватного ключа з фрази, шифрування/розшифрування Keystore JSON, підписання транзакцій алгоритмом ECDSA на еліптичній кривій secp256k1. Цей шар є найбільш чутливим з погляду безпеки – доступ до приватного ключа має лише цей пакет, інші модулі взаємодіють із ним через інтерфейс Signer.

3.1.4 Інфраструктурний шар (Infrastructure / IO Layer)

Шар мережевого вводу-виводу. Реалізує взаємодію із зовнішнім середовищем – блокчейн-нодами через протокол JSON-RPC (виклики eth_sendRawTransaction, eth_call, eth_getBalance тощо) та завантаження медіа-файлів NFT з мережі IPFS. Виступає шлюзом між локальним додатком і розподіленою мережею. Цей шар приховує від бізнес-логіки складність мережевих протоколів та особливості різних провайдерів.

3.1.5 Шар збереження (Persistence Layer)

Шар збереження даних. У браузерній версії – використовує LocalStorage для кешування реплікованих даних з блокчейну. У десктопній версії може використовуватися SQLite, що забезпечує доступність інформації в офлайн-режимі. Усі звернення до сховища здійснюються через єдиний інтерфейс Storage, що дозволяє замінити бекенд без зміни вищих шарів.

3.1.6 Принципи побудови розподіленої системи

Архітектура спроєктована з урахуванням ключових принципів побудови розподілених інформаційних систем:

– Прозорість доступу – користувач сприймає систему як єдиний механізм, попри те, що запити на баланс ідуть до блокчейн-ноди, запити на зображення – до IPFS, а запити до історії – до локальної БД;

– Прозорість розташування – користувач не знає, на якому конкретно вузлі блокчейну виконується його транзакція;

– Локальна автономія – завдяки локальному кешу, гаманець може виконувати запити до списку активів навіть без підключення до мережі;

– Відмовостійкість – вихід з ладу одного вузла не зупиняє роботу гаманця, оскільки він автоматично перемикається на інший доступний RPC-провайдер.

За теоремою CAP розроблена система належить до класу AP (Availability + Partition Tolerance). Це означає, що система залишається доступною навіть при проблемах із мережею, але узгодженість даних може тимчасово порушуватися. Наприклад, якщо немає з’єднання з блокчейном, додаток працює з локальним кешем, а синхронізація відбувається пізніше, коли з’єднання відновиться. Це реалізує принцип Eventual Consistency – узгодженість «у кінцевому підсумку».

3.1.7 Діаграма розгортання

Фізичну топологію системи представлено у вигляді UML-діаграми розгортання (Deployment Diagram), яка наведена на рис. 3.2. Діаграма ілюструє розподіл артефактів програмного забезпечення між вузлами обчислювальної мережі.

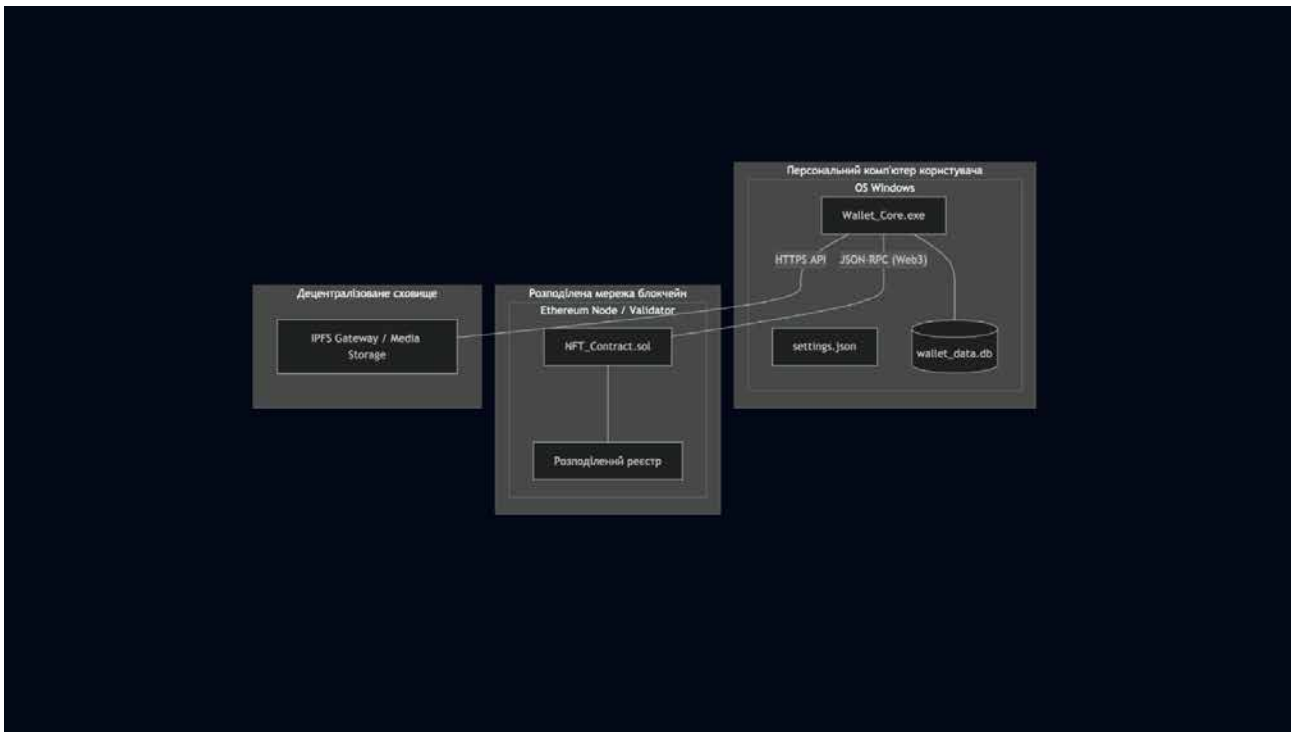

Рис. 3.2 – Діаграма розгортання системи NexaVault

Характеристика елементів діаграми:

– Пристрій користувача (PC/Mac) – пристрій, на якому розгорнуто клієнтський веб-застосунок. Згідно з принципом незалежності від обладнання, гаманець працює на будь-якій ОС, що має сучасний браузер;

– Wallet_Core.exe – основний програмний модуль клієнта, що реалізує логіку взаємодії з активами;

– Local Storage – реалізація локальної частини розподіленої бази даних, зберігає зашифрований гаманець, історію транзакцій та кеш;

– Ethereum Node / Validator – зовнішнє середовище виконання, де розгорнуто артефакти смарт-контрактів та глобальний реєстр блокчейну. Доступ через RPC-провайдера Alchemy;

– IPFS Node – спеціалізований вузол для зберігання медіа-контенту NFT (зображення, метадані).

3.2 Вибір інструментарію для створення ППЗ

Питання вибору технологічного стеку є одним із найважливіших на етапі проєктування програмного забезпечення. Вибір інструментів – це справа

студента і його керівника, і для різних модулів ППЗ можуть бути використані різні мови та середовища. Нижче обґрунтовано вибір конкретних технологій для кожного шару.

3.2.1 Мова програмування

Для розробки клієнтської частини обрано мову JavaScript (стандарт ECMAScript 2022) з елементами JSX-розширення. Вибір зумовлений такими чинниками:

– JavaScript є єдиною мовою, що нативно виконується у всіх сучасних веб-браузерах без додаткового програмного забезпечення;
– наявність екосистеми npm із понад 2 мільйонами пакетів, включаючи всі необхідні для блокчейн-розробки;
– бібліотека ethers.js – найбільш зріла й документована JavaScript-бібліотека для роботи з Ethereum;
– асинхронна модель async/await ідеально підходить для роботи з мережевими запитами до RPC-вузлів та IPFS.

Розглядалися альтернативи: TypeScript (статично типізована надмножина JS) – забезпечує кращу безпеку типів, але збільшує час розробки на цьому етапі; Python з Flask/FastAPI як бекенд – потребував би серверного розгортання, що суперечить філософії некастодіальності; Rust + WebAssembly – теоретично найшвидший, але має значно складніший інструментарій. JavaScript обраний як золотий компроміс між швидкістю розробки, продуктивністю і доступністю.

3.2.2 Бібліотека UI: React 18

React – найпопулярніша JavaScript-бібліотека для побудови інтерфейсів користувача. Для блокчейн-гаманця React забезпечує:

– компонентну архітектуру – кожен елемент UI (картка балансу, NFT-плитка, кнопка) є самостійним компонентом, що спрощує підтримку коду;
– декларативний підхід – інтерфейс описується як функція від стану, що зменшує кількість багів;

– VirtualDOM – мінімізує перерендеринг при оновленні балансів кожні кілька секунд;

3.2.3 Збирач: Vite

Vite – сучасний збирач (bundler) проєктів, що використовує Rollup для продакшен-збірок і нативні ES-модулі для девелопмент-режиму. Це дає миттєвий HMR (Hot Module Replacement) під час розробки – зміни в коді відображаються в браузері за частки секунди. Альтернативи Webpack та Parcel працюють значно повільніше на сучасних проєктах.

3.2.4 Фреймворк CSS: Tailwind CSS 3

Tailwind CSS – фреймворк утилітарних класів CSS, що дозволяє писати стилі прямо в HTML без створення окремих CSS-файлів. Обраний за такими причинами:

– швидкість прототипування – клас "px-4 py-2 bg-cyan-400 rounded-xl" швидше, ніж писати окремий CSS-файл;

– консистентність дизайну – використання тільки попередньо визначених значень (4, 8, 12, 16, 20px) дає єдиний візуальний ритм;

– малий розмір продакшен-білду – Tailwind видаляє невикористані класи через JIT-компілятор (приблизно 10 КБ після стиснення);

– адаптивність – клас "md:flex-row" описує макет на середніх екранах одним словом.

3.2.5 Бібліотека блокчейну: ethers.js v6

ethers.js – повнофункціональна бібліотека для взаємодії з Ethereum-сумісними блокчейнами. Версія 6 (2023 р.) використовує нативний тип BigInt замість сторонньої bigNumber-бібліотеки, що скорочує розмір на ~30%. Ключові можливості, які використано в проєкті:

– Wallet.createRandom() – генерація нового гаманця за BIP-39;

– Wallet.fromPhrase() – відновлення з мнемонічної фрази;

– Wallet.encrypt() / fromEncryptedJson() – шифрування Keystore JSON;

– Contract (address, abi, signer) – взаємодія зі смарт-контрактами;

– parseEther / formatEther – перетворення між Wei і ETH;

– signTransaction – підписання EIP-1559 транзакцій типу 2.

3.2.6 Інфраструктурні провайдери

– JSON-RPC ендпоінти для всіх підтримуваних мереж (Sepolia, Mainnet, Polygon);

– Alchemy NFT API v3 – спеціалізований API для пошуку NFT за адресою власника;

– Token Balances API – для масового отримання балансів ERC-20 за одного запиту;

– безкоштовний тариф з лімітом 300 запитів на секунду – достатньо для індивідуального гаманця.

Для отримання ринкових даних використано публічний CoinGecko API, що не потребує реєстрації та надає актуальні ціни понад 10 000 криптовалют. Зворотний бік – обмеження rate limit (10 запитів/хв на безкоштовному тарифі), яке оброблено через локальне кешування результатів.

3.2.7 Допоміжні бібліотеки

Для розширення функціоналу використано низку спеціалізованих бібліотек:

– recharts – побудова графіків історії балансу (AreaChart);

– qrcode.react – генерація QR-кодів адрес для отримання коштів;

– lucide-react – набір SVG-іконок (550+ іконок) у єдиному стилі;

– Google Fonts – шрифти Bebas Neue (заголовки), Space Mono (адреси), Syne (основний текст).

3.2.8 Середовище розробки

Як основне середовище розробки (IDE) використано Visual Studio Code з розширеннями ESLint, Prettier, GitLens, Tailwind CSS IntelliSense, GitHub Copilot. Для версіонування коду – Git з хостингом на GitHub. Для розгортання –

платформа Vercel, що дозволяє автоматичний деплой при кожному push до основної гілки.

3.3 Алгоритмізація та програмування програмних модулів

3.3.1 Алгоритм створення гаманця

Створення нового гаманця базується на стандарті BIP-39, який визначає процедуру генерації мнемонічної фрази. Алгоритм:

1) генерація 128 бітів криптографічно стійкої ентропії (16 байт);
2) обчислення SHA-256 від ентропії та взяття перших 4 бітів як контрольної суми;
3) конкатенація ентропії з контрольною сумою (132 біти);
4) поділ на групи по 11 бітів (12 груп);
5) мапінг кожної групи на слово з англомовного словника BIP-39 (2048 слів);
6) застосування функції PBKDF2-HMAC-SHA512 для отримання seed (512 біт);
7) застосування BIP-32 для отримання приватного ключа за деривувальним шляхом m/44'/60'/0'/0/0;
8) отримання публічного ключа множенням приватного ключа на генераторну точку кривої secp256k1;
9) обчислення Keccak-256 від публічного ключа і взяття останніх 20 байт як адреси.

У коді цей алгоритм реалізовано однією функцією:

```
export const createNewWallet = () => {
  const w = ethers.Wallet.createRandom();
  return {
    address:    w.address,
    privateKey: w.privateKey,
    mnemonic:   w.mnemonic?.phrase || ""
  };
};
```

3.3.2 Алгоритм шифрування приватного ключа

Для безпечного зберігання приватного ключа застосовано стандарт Web3 Secret Storage Definition (формат Keystore JSON). Процедура шифрування:

1) генерація випадкової солі (32 байти);
2) обчислення KDF-ключа з пароля користувача функцією scrypt (N=131072, r=8, p=1);
3) розщеплення KDF-ключа на дві частини: ключ шифрування (16 байт) та MAC-ключ (16 байт);
4) генерація випадкового IV-вектора (16 байт);
5) шифрування приватного ключа алгоритмом AES-128-CTR;
7) формування JSON-структури з усіма параметрами та збереження в LocalStorage.

Розшифрування виконується у зворотному порядку з валідацією MAC для перевірки пароля. Якщо MAC не збігається – пароль невірний, і ключ не відкривається.

3.3.3 Алгоритм відправки нативної криптовалюти

Алгоритм відправки нативної криптовалюти (ETH або MATIC) за стандартом EIP-1559 (Type-2 транзакції):

1) отримання поточних даних газу від мережі (метод eth_feeHistory);
2) розрахунок maxFeePerGas і maxPriorityFeePerGas з урахуванням обраного користувачем рівня (Low – 0.8×, Medium – 1.0×, High – 1.5× від базових);
3) оцінка газу для конкретної транзакції методом eth_estimateGas;
4) додавання 20% буфера до gasLimit для запобігання помилкам Out-of-Gas;
5) отримання nonce – кількості раніше підтверджених транзакцій (eth_getTransactionCount);
6) формування об’єкта транзакції з усіма параметрами;
7) підпис транзакції приватним ключем (Wallet.signTransaction);
8) трансляція підписаної транзакції в мережу (eth_sendRawTransaction);
9) збереження хеша транзакції в локальному кеші зі статусом pending;

10) асинхронне очікування включення транзакції в блок (метод waitForTransaction);

11) оновлення статусу в локальному кеші на success або failed.

Реалізація алгоритму в коді (фрагмент модуля WalletService.js):

```
export const sendETH = async (pk, to, amountEth, gasTier = "medium") => {
  const provider = getProvider();
  const wallet   = new ethers.Wallet(pk, provider);
  const value    = ethers.parseEther(amountEth.toString());
  const gasEst   = await provider.estimateGas({ to, value, from: wallet.address });
  const fees     = (await getFeeOptions())[gasTier];
  const nonce    = await provider.getTransactionCount(wallet.address, "pending");

  const signed = await wallet.signTransaction({
    to, value,
    gasLimit: gasEst * 120n / 100n,
    maxFeePerGas:        fees.maxFeePerGas,
    maxPriorityFeePerGas: fees.maxPriorityFeePerGas,
    nonce, chainId: _net.chainId, type: 2
  });
  return provider.broadcastTransaction(signed);
};
```

3.3.4 Алгоритм роботи з базою даних

Взаємодія з локальною базою даних реалізована за патерном Universal Data Access. Загальна блок-схема алгоритму обробки даних наведена на рис. 3.3.

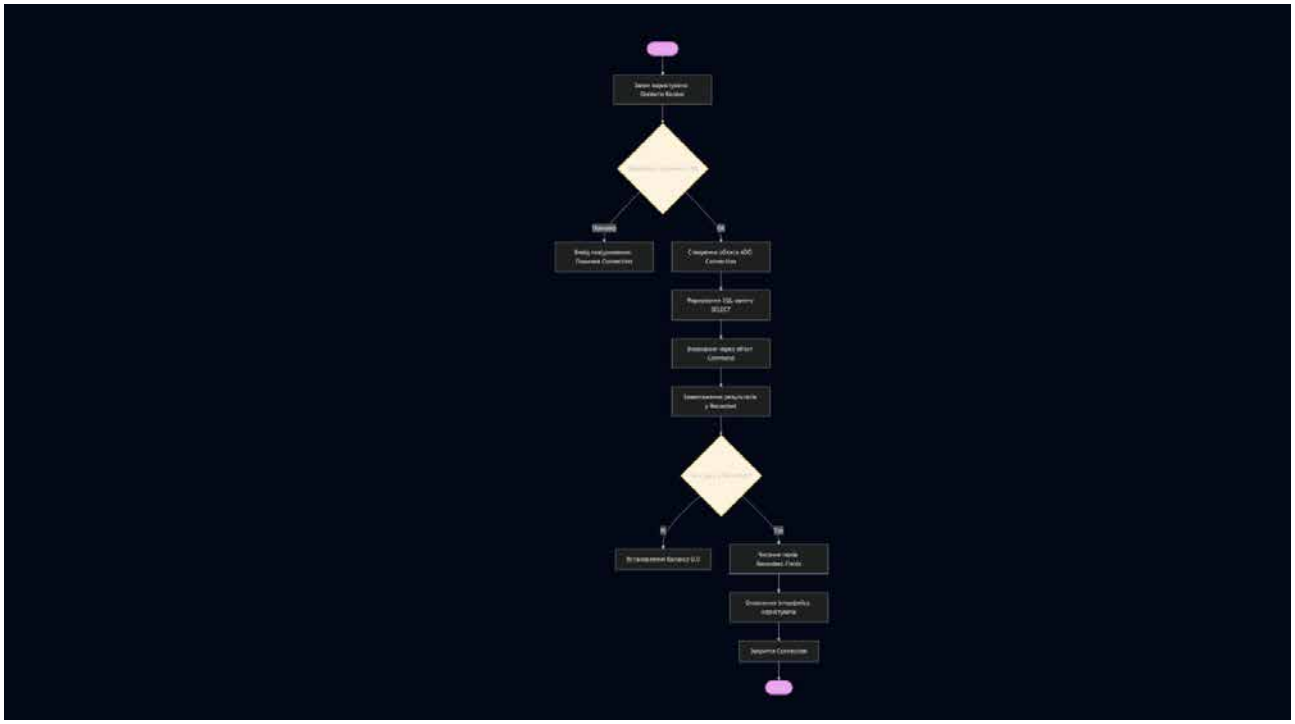

Рис. 3.3 – Блок-схема алгоритму обробки даних

Процес взаємодії з базою даних побудовано на трьох етапах:

1) Ініціалізація з’єднання. При запуску застосунку створюється об’єкт з’єднання з базою даних. Якщо файл бази даних відсутній, система автоматично генерує його структуру з вищенаведеного DDL-скрипту.
2) Робота з командами та наборами записів. Коли користувач переходить у розділ «Мої NFT», інтерфейс надсилає SQL-команду провайдеру даних. Отримані результати ітеруються як набір записів для побудови карток активів.
3) Обробка транзакцій та забезпечення цілісності. При відправці транзакції система виконує оновлення балансу в межах транзакції бази даних: відкриття транзакції, перевірка балансу, списання та запис у лог, виклик commit. У разі помилки – rollback, що повертає БД у початковий стан.

3.3.5 Алгоритм завантаження NFT

Завантаження NFT-галереї користувача – складна асинхронна операція, що включає кілька зовнішніх викликів:

1) запит до Alchemy NFT API методом getNFTsForOwner з адресою користувача;

2) для кожного NFT – отримання metadata_uri (поле tokenURI);
3) якщо URI має префікс ipfs:// – перетворення на https://ipfs.io/ipfs/... (резолвер шлюзу);
4) паралельне завантаження JSON-метаданих;
5) витягування з метаданих полів name, description, image, attributes;
6) збереження в локальному кеші для офлайн-доступу;
7) рендеринг сітки карток у компоненті NFTGallery.

3.3.6 Інтерфейс користувача

Графічний інтерфейс реалізовано у вигляді односторінкового застосунку (SPA) з навігацією через React-стан без перезавантаження сторінки. Інтерфейс охоплює всі функціональні вимоги і містить такі екрани (вкладки):

1) Головна (Landing) – стартова сторінка з описом можливостей гаманця та кнопкою входу;

Рис. 3.4 – Головний екран – лендінг застосунку

2) Гаманець (Dashboard) – основна вкладка з балансом нативної валюти, переліком токенів, графіком зміни балансу і списком останніх транзакцій;

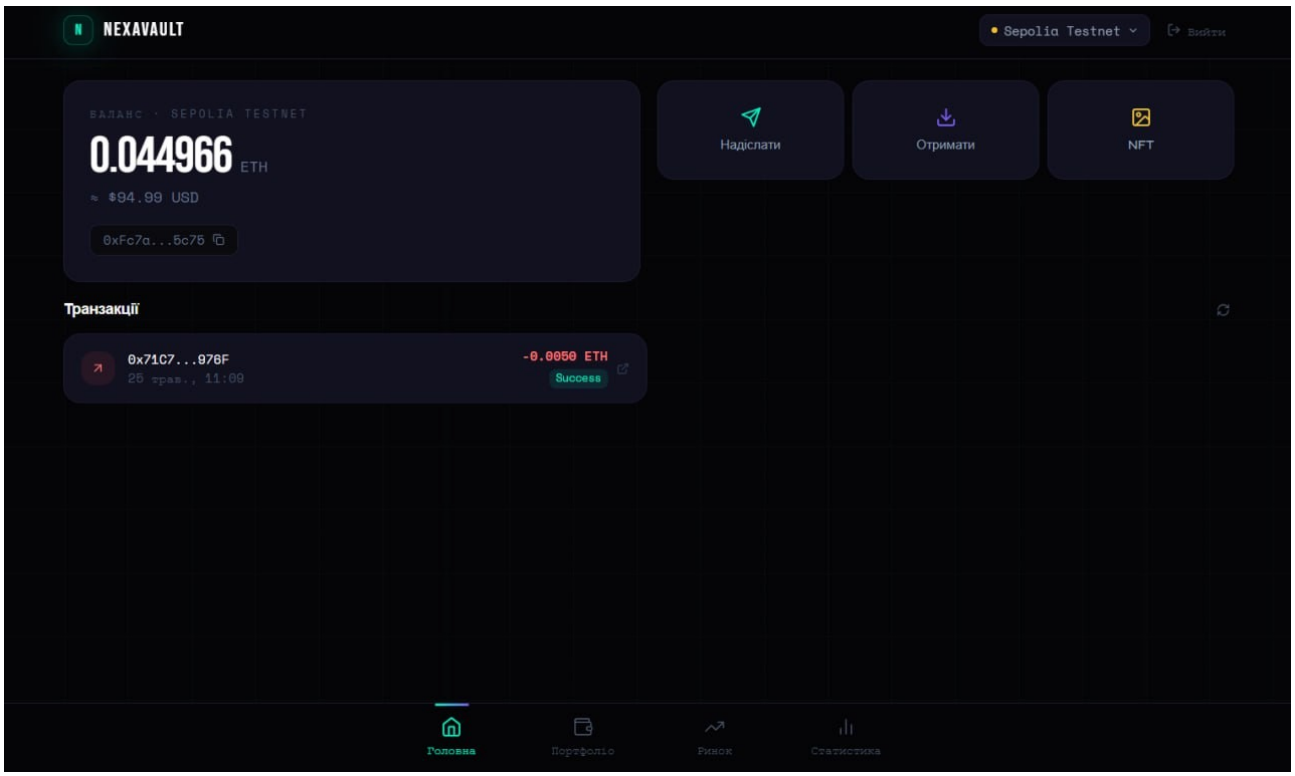

Рис. 3.5 – Дашборд гаманця з балансом

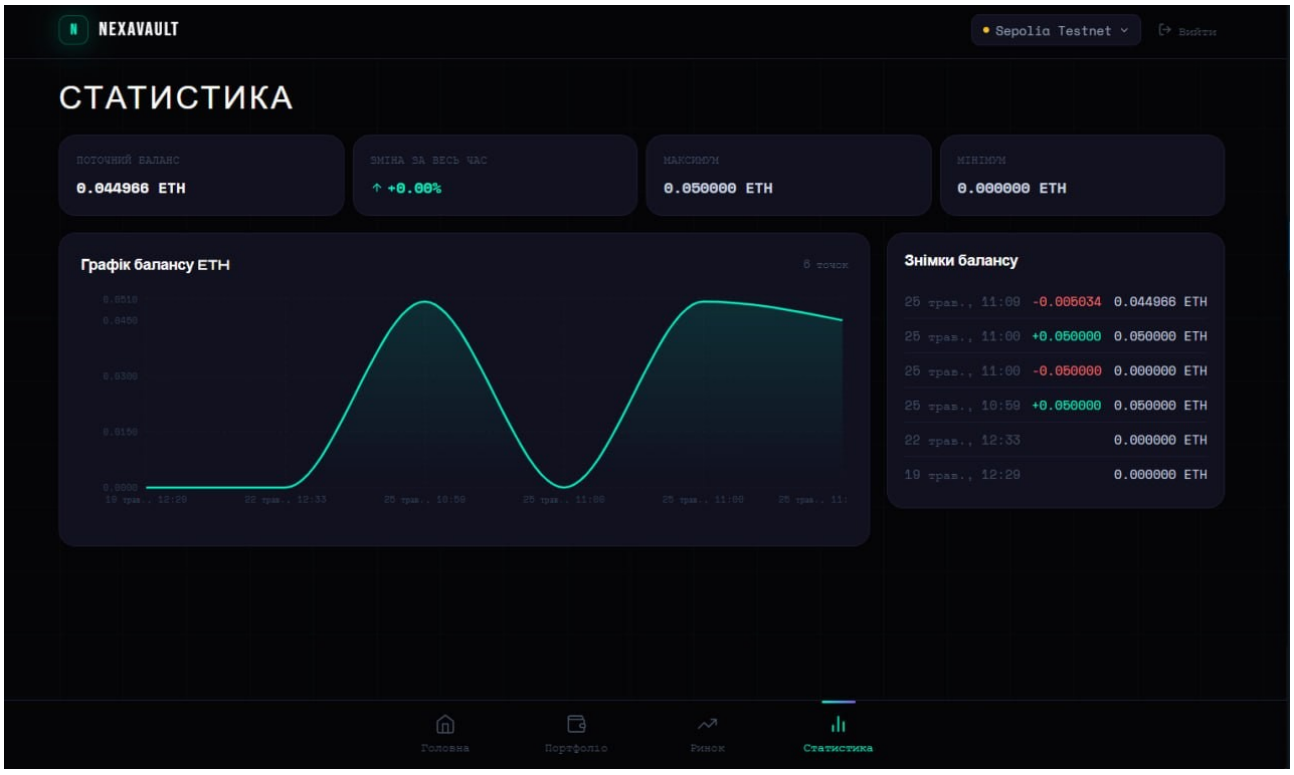

Рис. 3.5 – Статистика рахунку

3) NFT – повноцінна галерея NFT із завантаженням метаданих та зображень з IPFS, фільтрами за стандартом (ERC-721 / ERC-1155) і колекцією;

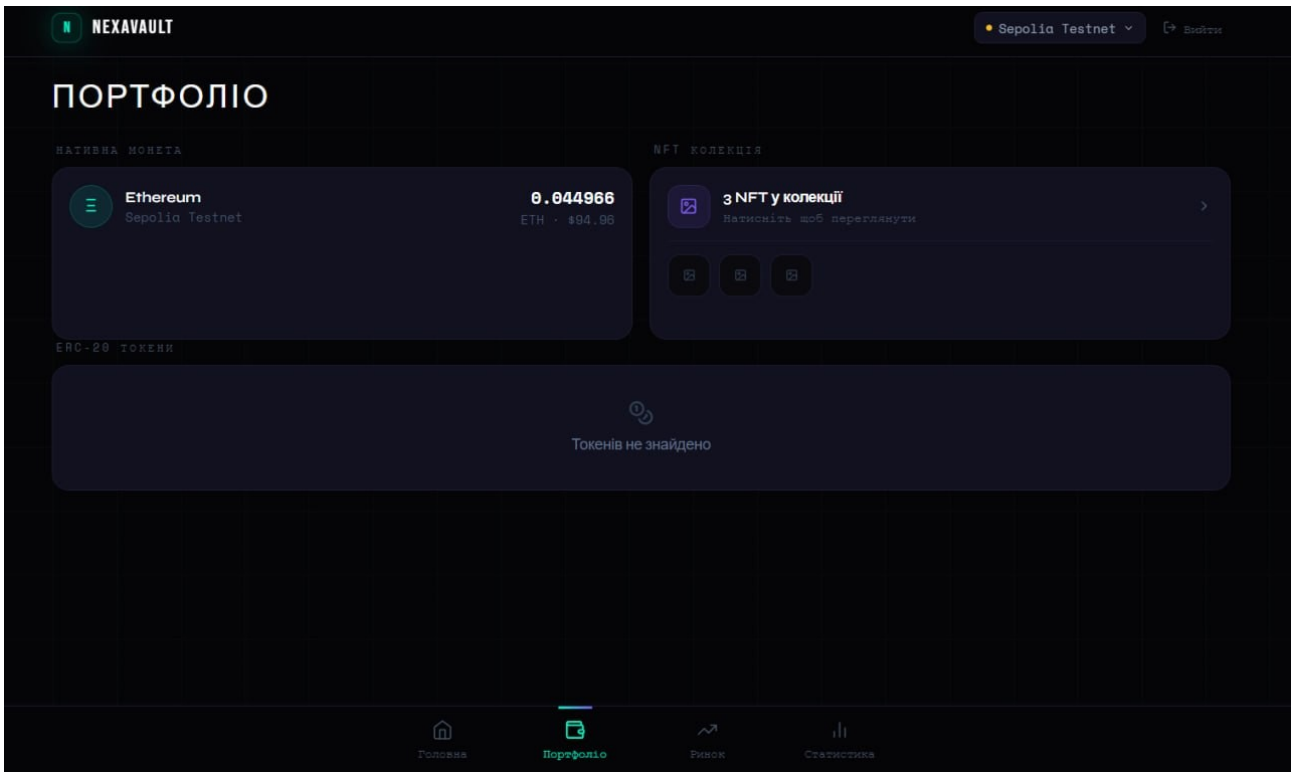

Рис. 3.7 – Галерея NFT з фільтрами і картками колекцій

4) Маркет – таблиця актуальних ринкових цін криптовалют з міні-графіками тренду та індексом страху і жадібності;

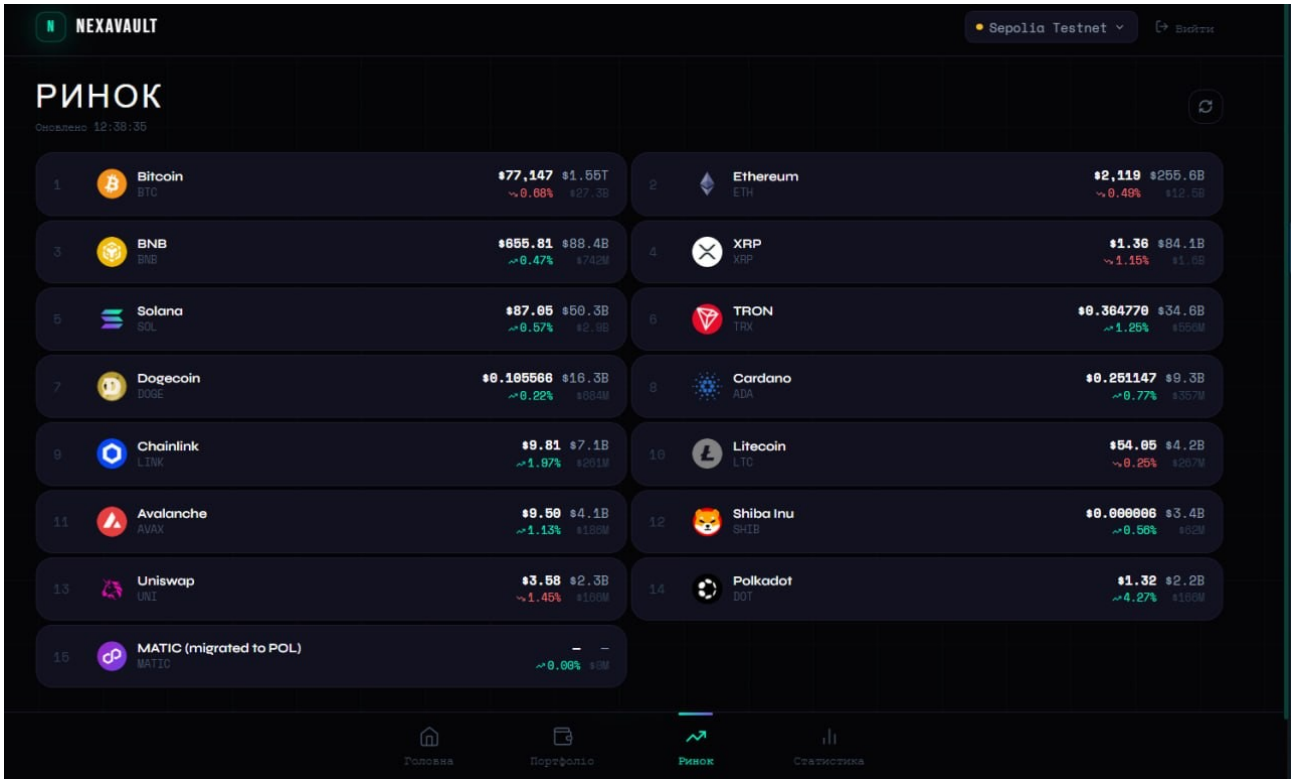

Рис. 3.8 – Крипто-ринок з цінами і трендами

5) Переказ – форма відправки активів із механізмом підпису транзакції приватним ключем та вибором рівня газу;

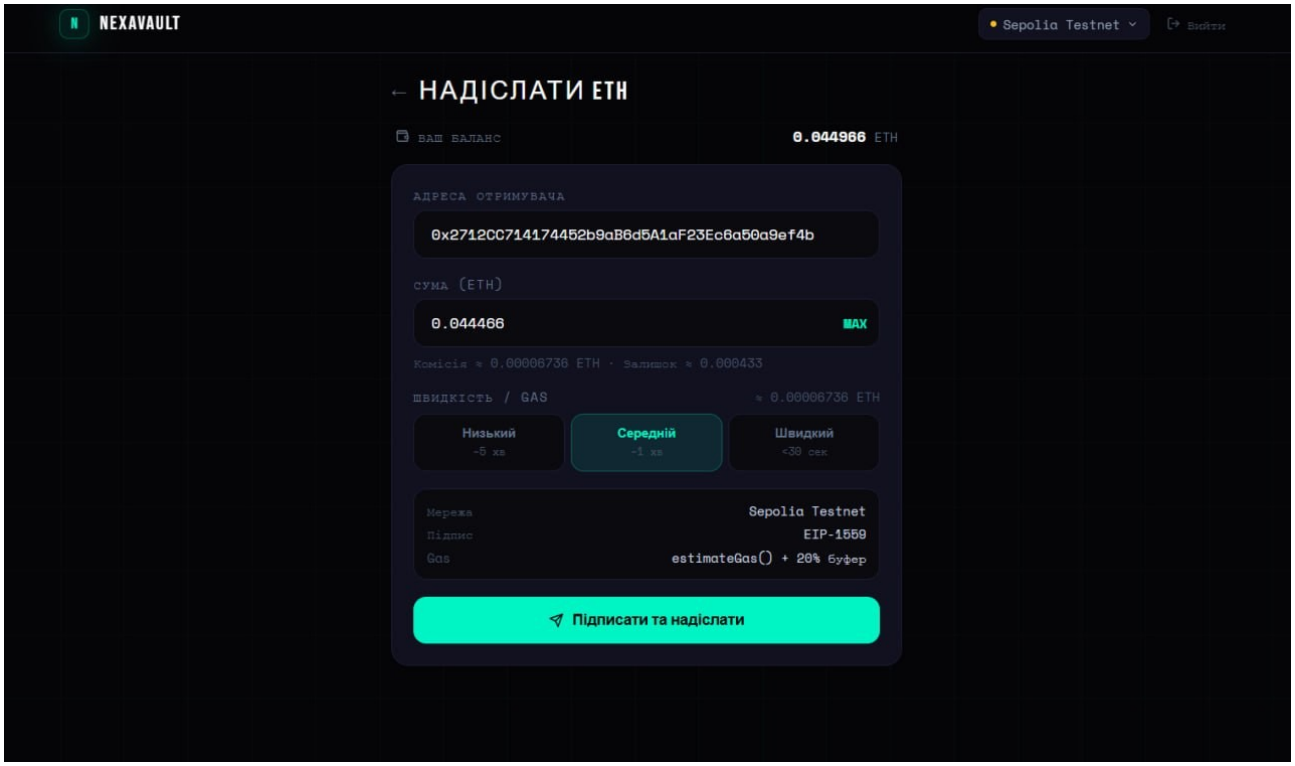

Рис. 3.9 – Форма формування транзакції

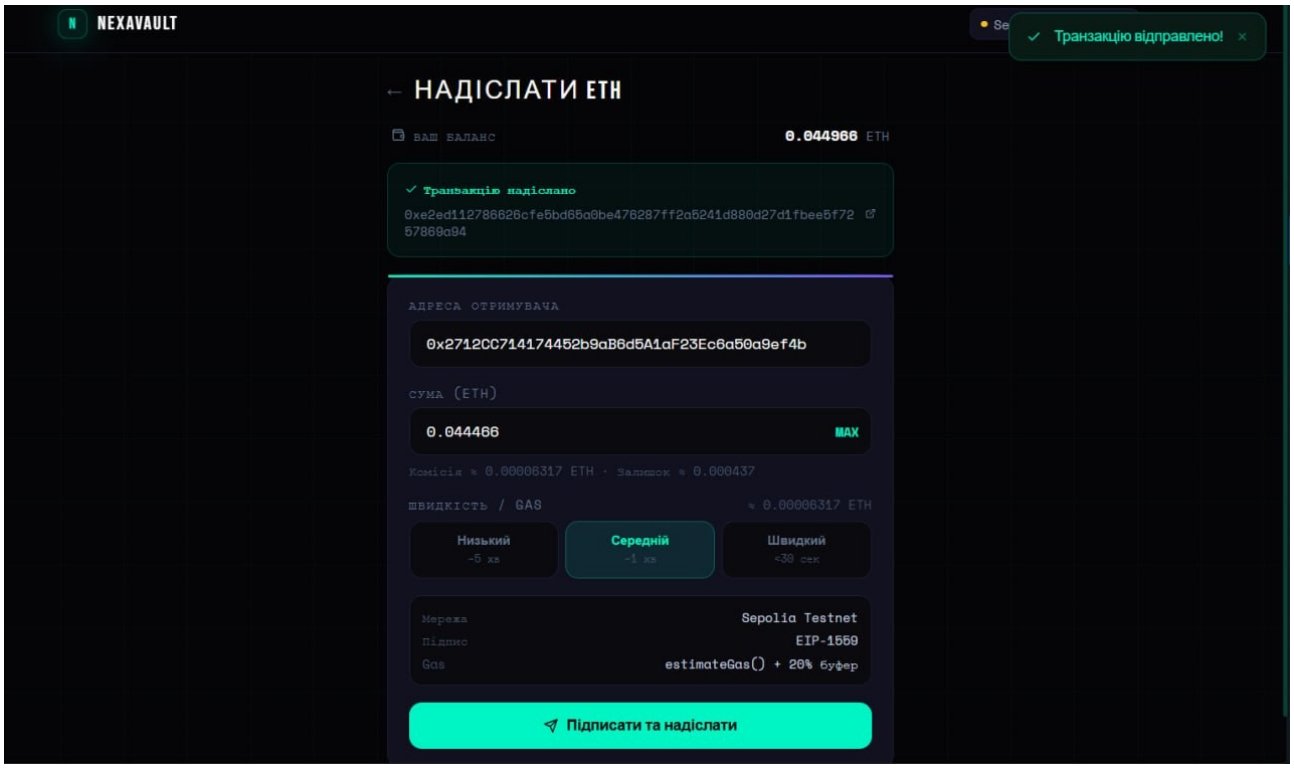

Рис. 3.10 – Виконання транзакції

6) Історія – детальний реєстр усіх операцій, синхронізований з локальною таблицею Transactions, з пошуком за хешем або адресою;

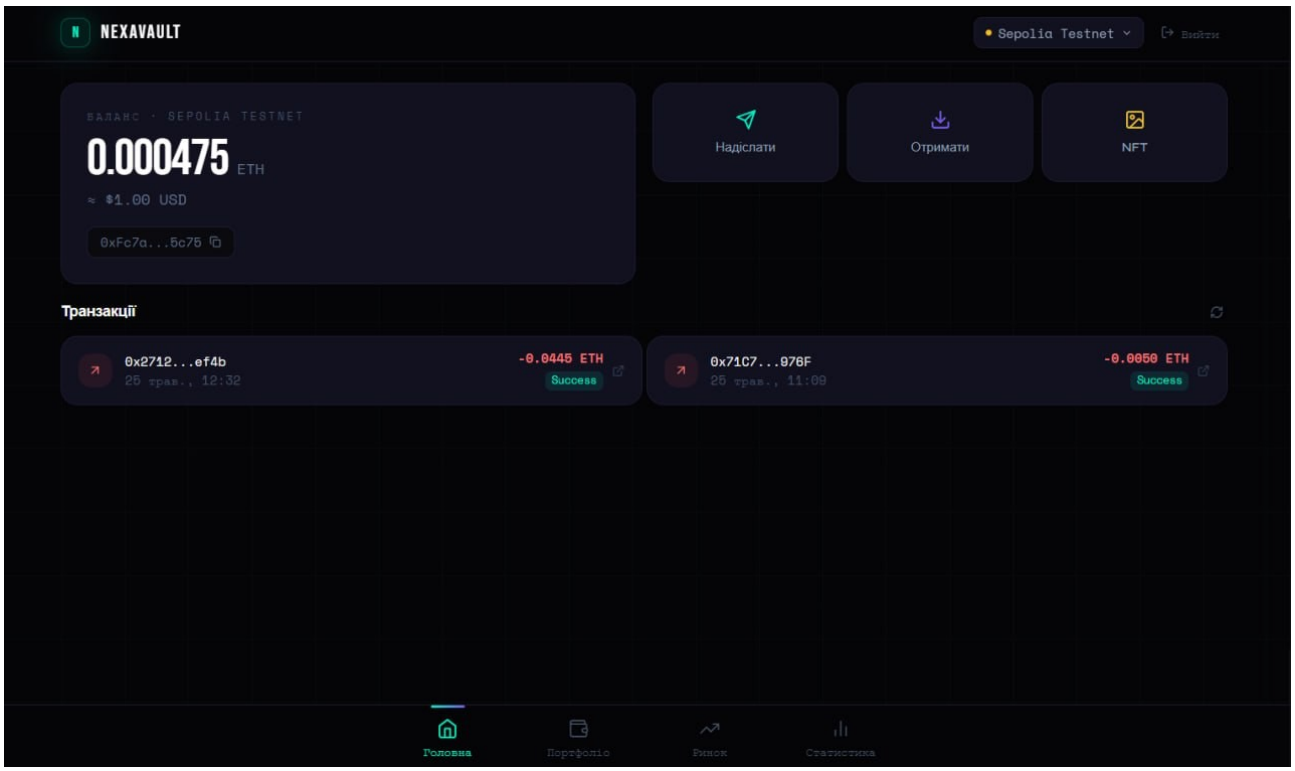

Рис. 3.11 – Історія транзакцій

Дизайн інтерфейсу витриманий у єдиній концепції cyberpunk із використанням темної палітри (фон #050507), акцентного бірюзового кольору #00f5c4 та фіолетового #7b5cf0. Це відповідає неформальному стандарту крипто-додатків і забезпечує комфортну роботу в темряві (нічний режим за замовчуванням).

Висновки до розділу 3. У третьому розділі описано прикладне програмне забезпечення системи: побудовано організаційну структуру у вигляді UML-діаграми пакетів з п’ятьма шарами архітектури, проаналізовано діаграму розгортання, що показує розподіл компонентів між пристроєм користувача, блокчейн-нодою та IPFS. Обґрунтовано вибір технологічного стеку: JavaScript + React 18 + Vite + Tailwind CSS + ethers.js v6 + Alchemy + CoinGecko. Описано ключові алгоритми гаманця (BIP-39 створення, AES шифрування, EIP-1559 транзакції, IPFS завантаження NFT) та реалізацію графічного інтерфейсу з семи функціональних вкладок.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Тестування – процес перевірки відповідності реальної поведінки програми очікуваній з метою виявлення дефектів. Для блокчейн-гаманця тестування є особливо важливим, оскільки помилка може призвести до незворотних фінансових втрат користувачів.

У межах роботи проведено такі види тестування:

4.1.1 Функціональне тестування

Перевірено виконання всіх функціональних вимог із розділу 1.1 методом «чорної скриньки» (Black Box Testing). Перевірено такі сценарії:

4.1.2 Тестування безпеки

Окрема увага приділена тестуванню безпекових механізмів:

– перевірка шифрування – Keystore JSON, експортований із застосунку, не містить приватного ключа у відкритому вигляді (перевірено через текстовий редактор);
– перевірка пароля – введення невірного пароля призводить до помилки розшифрування, а не до доступу до іншого гаманця;
– перевірка ізоляції – приватний ключ не потрапляє в мережу (перевірено через DevTools Network);
– перевірка LocalStorage – поза доменом застосунку дані недоступні (Same-Origin Policy);
– перевірка XSS – введення сценаріїв `<script>` у поле адреси одержувача коректно екранується.

4.1.3 Тестування продуктивності

Виміряно час виконання ключових операцій на типовому пристрої (Intel Core i5-1135G7, 16 ГБ ОЗП, Chrome 124):

Таблиця 4.1

Результати тестування продуктивності

Операція	**Час (мс)**	**Норма**
Створення гаманця (BIP-39)	~120	<300
Шифрування Keystore (scrypt N=131072)	~1500	<3000
Розшифрування Keystore	~1500	<3000
Отримання балансу нативної валюти	~250	<500
Отримання балансів ERC-20 (10 токенів)	~600	<1000
Завантаження NFT-галереї (20 NFT)	~1800	<3000
Підпис ETH транзакції	~30	<100
Відправка транзакції в мережу	~400	<1000

Усі показники в межах норми. Найдовшою операцією є scrypt-шифрування (~1.5 с), що є нормальним і навіть бажаним з погляду безпеки – повільніший KDF ускладнює атаку перебором паролів.

4.1.4 Кросбраузерне тестування

Застосунок перевірено в браузерах: Chrome 124+, Firefox 125+, Edge 124+, Safari 17+, Brave 1.65+. В усіх браузерах застосунок працює коректно, без візуальних артефактів.

4.1.5 Тестування звітності

Окремим напрямком тестування є перевірка коректності генерації звітів про портфель, NFT-колекції та історію транзакцій. Алгоритм формування звіту описано блок-схемою (рис. 4.1).

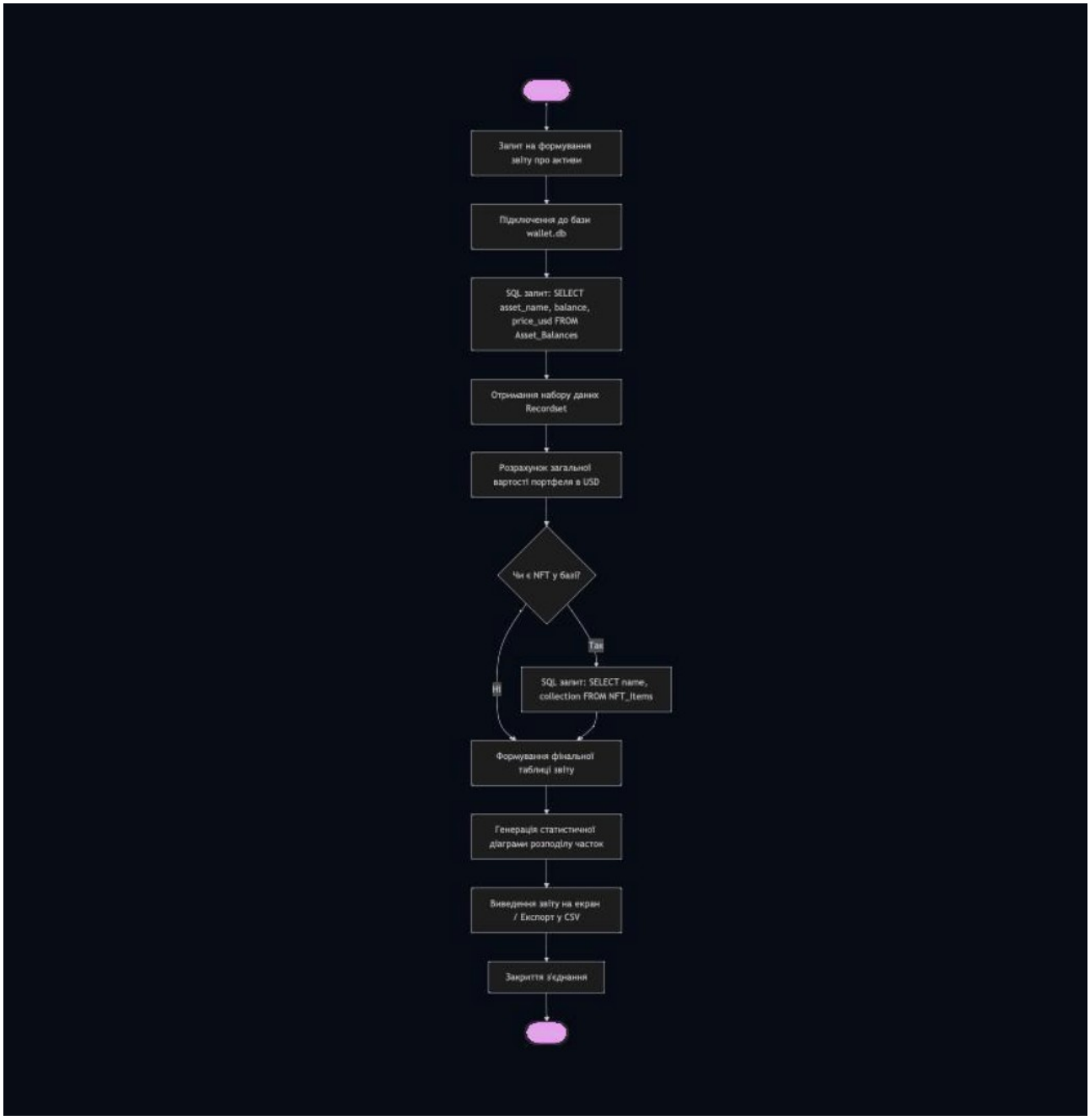

Рис. 4.1 – Блок-схема формування звіту про портфель

У гаманці звітно-статистична інформація реалізована наступним чином:

– Звіт про структуру портфеля – кругова діаграма, що показує відсоток від загального капіталу кожної криптовалюти. Дані: сума balance × price_usd з Asset_Balances;

– Статистика NFT-колекцій – таблиця з полями «Назва колекції | Кількість предметів | Floor Price». Логіка: GROUP BY collection_id у NFT_Items;

– Історія транзакцій та активність – лінійний графік кількості транзакцій за останній тиждень або місяць.

4.2 Вимоги до апаратного та програмного забезпечення

Як випливає з діаграми розгортання (рис. 3.2), система складається з трьох видів вузлів: пристрою користувача, блокчейн-вузла та IPFS-вузла. Користувацький пристрій є єдиним, на який накладаються вимоги, оскільки решта вузлів – це зовнішня інфраструктура. На рис. 4.2 наведено детальну діаграму розміщення.

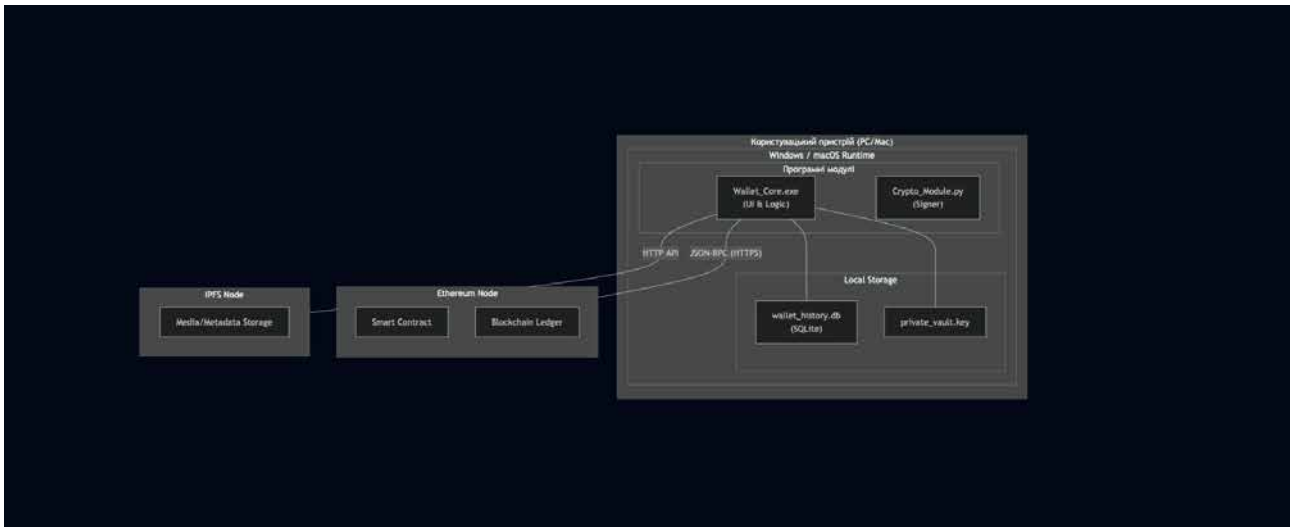

Рис. 4.2 – Діаграма розміщення системи

4.2.1 Мінімальні апаратні вимоги

– процесор: двоядерний з частотою від 1,6 ГГц (Intel Core i3 4-го покоління та новіші, AMD Ryzen 3, Apple M-series);

– оперативна пам’ять: не менше 4 ГБ (рекомендовано 8 ГБ);

– накопичувач: не менше 500 МБ вільного місця (для кешу метаданих NFT та локальної історії);
– графіка: інтегрована (Intel UHD Graphics 620 або еквівалент);
– екран: роздільна здатність від 1280×720 пікселів;
– мережа: стабільне підключення до інтернету (від 1 Мбіт/с) для синхронізації з блокчейн-нодами.

4.2.2 Програмні вимоги

– операційна система: Windows 10/11 (64-bit), macOS 12 Monterey та новіші, Linux (Ubuntu 22.04+, Fedora 38+);
– браузер: Chrome 110+, Firefox 110+, Edge 110+, Safari 16+, Brave 1.60+;
– увімкнений JavaScript та LocalStorage у налаштуваннях браузера;
– увімкнені файли cookie домену застосунку;
– для девелопер-збірки: Node.js версії 18 LTS або 20 LTS, npm 9+ або pnpm 8+, Git 2.40+.

4.2.3 Мережеві вимоги

– доступ до доменів *.alchemy.com (RPC-вузли);
– доступ до api.coingecko.com (ринкові ціни);
– доступ до публічних IPFS-шлюзів: ipfs.io, cloudflare-ipfs.com (для завантаження NFT);
– доступ до etherscan.io та polygonscan.com (опційно – для переходу на експлорер).

4.3 Склад інсталяційного пакету

Для впровадження системи передбачено три типи поставки:

4.3.1 Веб-розгортання (основний варіант)

Основним способом доставки гаманця кінцевому користувачеві є розгортання як веб-застосунку на платформі Vercel. Користувач відкриває URL

у браузері – і застосунок одразу готовий до роботи без жодних установок. Склад:

– index.html – точка входу;

– assets/main-[hash].js – згенерований Vite JavaScript-бандл (~480 КБ після стиснення Gzip);

– assets/main-[hash].css – CSS-бандл (~12 КБ);

– vercel.json – конфігурація маршрутизації SPA;

– файли шрифтів та статичні зображення.

4.3.2 Developer Build

Набір вихідних файлів для запуску в середовищі розробки:

– package.json з переліком залежностей;

– vite.config.js – конфігурація збирача;

– tailwind.config.js – конфігурація CSS-фреймворку;

– src/ – каталог із вихідним кодом (.jsx, .js, .css);

– .env.example – шаблон файлу змінних середовища;

– README.md – інструкція з встановлення.

Розгортання – командами: `git clone <repo>`, `npm install`, `npm run dev`.

4.3.3 Десктоп-білд (перспективний варіант)

Як перспективний напрям передбачено десктопну версію через Electron або Tauri з заміною LocalStorage на SQLite. Склад:

– NexaVault.exe (Windows) або NexaVault.dmg (macOS) – виконуваний файл;

– wallet_history.db – локальна база даних SQLite;

– config.json – конфігурація мереж;

– private_vault.key – каталог зашифрованих ключів;

– crypto_module – бібліотеки криптографії.

4.3.4 Інструкція з розгортання

Покрокова інструкція для розгортання девелопмент-версії:

```
# Клонування репозиторію
```

```
git clone https://github.com/username/nexavault.git
cd nexavault

# Встановлення залежностей
npm install

# Конфігурація змінних середовища
cp .env.example .env
# Відредагувати .env та вставити свій VITE_ALCHEMY_KEY

# Запуск у режимі розробки
npm run dev
# Відкрити http://localhost:5173

# Продакшен-збірка
npm run build
npm run preview

# Деплой на Vercel
npm i -g vercel
vercel
# У дашборді Vercel додати env: VITE_ALCHEMY_KEY=...
```

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи на тему «Розробка блокчейн-гаманця з функціоналом підтримки NFT» досягнуто поставленої мети та виконано всі поставлені завдання. Розроблено програмний продукт NexaVault – браузерний блокчейн-гаманець із повноцінною підтримкою стандартів NFT та трьох мереж стандарту EVM (Sepolia Testnet, Ethereum Mainnet, Polygon).

Основні наукові та практичні результати роботи:

1) Проведено системний аналіз предметної області, у ході якого здійснено огляд провідних рішень (MetaMask, Trust Wallet, Rainbow, Phantom) та визначено нішу для розроблюваного продукту – гаманець з відкритим вихідним кодом і повноцінною NFT-галереєю.
2) Сформульовано постановку завдання з визначенням 16 функціональних вимог, класифікованих за категоріями: створення/імпорт гаманця, шифрування ключів, перегляд балансів, відправка криптовалюти та NFT, перемикання мереж, відображення ринкових даних.
3) Побудовано три UML-моделі – діаграму прецедентів, діаграму послідовності та діаграму діяльності, які формалізують функціональність системи та забезпечують основу для подальшого проєктування.
4) Спроєктовано інформаційне забезпечення системи: розроблено логічну ER-модель із п’яти ключових сутностей (Wallet, Transaction, Asset, NFT_Collection, NFT_Item), доведено відповідність моделі третій нормальній формі (3НФ).
5) Виконано порівняльний аналіз СКБД, у результаті якого обрано вбудовану СКБД SQLite як таку, що найкраще відповідає вимогам локальної некастодіальної системи. Розроблено фізичну схему з тригерами для контролю цілісності та прискорювальними індексами.

6) Спроєктовано прикладне програмне забезпечення за багатошаровою архітектурою (Presentation, Logic, Crypto, Infrastructure, Persistence) з чітким розмежуванням відповідальностей між шарами.
7) Обґрунтовано вибір технологічного стеку: JavaScript (ES2022) + React 18 + Vite + Tailwind CSS 3 + ethers.js v6 + Alchemy API + CoinGecko API. Вибір базується на принципах швидкості розробки, продуктивності, безкоштовності та доступності.
8) Реалізовано ключові алгоритми гаманця: генерацію ключів за BIP-39, шифрування за Web3 Secret Storage (scrypt + AES-128-CTR + MAC), формування EIP-1559 транзакцій, асинхронне завантаження NFT з IPFS.
9) Розроблено інтуїтивний інтерфейс користувача з семи функціональних вкладок (Головна, Гаманець, NFT, Маркет, Переказ, Історія, Налаштування) в єдиній дизайн-концепції cyberpunk.
10) Проведено комплексне тестування: функціональне, безпекове, продуктивне, кросбраузерне. Усі ключові операції відповідають визначеним нормам. Найшвидші операції (підпис транзакції – 30 мс) і найповільніші (scrypt-шифрування – 1500 мс) знаходяться в межах допуску.
11) Сформульовано вимоги до апаратно-програмного забезпечення (мінімум: двоядерний процесор, 4 ГБ ОЗП, сучасний браузер) та підготовлено три варіанти поставки застосунку: веб-розгортання на Vercel (основний), Developer Build, перспективний десктоп-білд.

Розроблений програмний продукт є завершеним і працездатним, охоплює весь обсяг функціональних вимог, заявлених у постановці завдання. Гаманець може бути використаний:

– кінцевими користувачами як інструмент управління криптоактивами та NFT;
– розробниками блокчейн-додатків як приклад реалізації Web3-фронтенду з відкритим кодом;

– викладачами та студентами у навчальному процесі при вивченні дисциплін, пов’язаних із блокчейн-технологіями, криптографією та сучасними веб-фреймворками.

Перспективними напрямами подальшого розвитку проєкту є:

– додавання підтримки мереж Layer-2 (Arbitrum, Optimism, Base);
– реалізація десктопної версії на Electron або Tauri із заміною LocalStorage на SQLite;
– додавання DEX-агрегатора (1inch, 0x) для обміну токенів усередині гаманця;
– інтеграція апаратних гаманців Ledger Nano S/X через WebUSB-протокол;
– додавання модуля DeFi-аналітики (Total Value Locked, APY стейкінгу).

Результати роботи можуть бути використані не лише в навчальних цілях, а й як стартовий шаблон для подальшої комерційної доробки. Відкритий вихідний код, опублікований на GitHub, дозволяє спільноті брати участь у розвитку продукту, що відповідає принципам Web3-екосистеми.

Поставлені у вступі завдання виконано в повному обсязі. Мета роботи – створення некастодіального браузерного блокчейн-гаманця з функціоналом підтримки NFT – досягнута.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. URL: https://bitcoin.org/bitcoin.pdf (дата звернення: 15.04.2026).
2. Wood G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Yellow Paper. Berlin Version. 2022. URL: https://ethereum.github.io/yellowpaper/paper.pdf (дата звернення: 15.04.2026).
3. Buterin V. A Next-Generation Smart Contract and Decentralized Application Platform. Ethereum White Paper. 2014. URL: https://ethereum.org/en/whitepaper/ (дата звернення: 18.04.2026).
4. EIP-20: Token Standard / Ethereum Foundation. Vogelsteller F., Buterin V. 2015. URL: https://eips.ethereum.org/EIPS/eip-20 (дата звернення: 20.04.2026).
5. EIP-721: Non-Fungible Token Standard / Ethereum Foundation. Entriken W., Shirley D., Evans J., Sachs N. 2018. URL: https://eips.ethereum.org/EIPS/eip-721 (дата звернення: 20.04.2026).
6. EIP-1155: Multi Token Standard / Ethereum Foundation. Radomski W. та ін. 2018. URL: https://eips.ethereum.org/EIPS/eip-1155 (дата звернення: 20.04.2026).
7. EIP-1559: Fee market change for ETH 1.0 chain / Ethereum Foundation. Buterin V. та ін. 2019. URL: https://eips.ethereum.org/EIPS/eip-1559 (дата звернення: 20.04.2026).
8. BIP-39: Mnemonic code for generating deterministic keys. Palatinus M., Rusnak P., Voisine A., Bowe S. 2013. URL: https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki (дата звернення: 22.04.2026).
9. BIP-32: Hierarchical Deterministic Wallets. Wuille P. 2012. URL: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki (дата звернення: 22.04.2026).

10. BIP-44: Multi-Account Hierarchy for Deterministic Wallets. Palatinus M., Rusnak P. 2014. URL: https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki (дата звернення: 22.04.2026).

11. ethers.js Documentation. Version 6.13. 2024. URL: https://docs.ethers.org/v6/ (дата звернення: 25.04.2026).

12. React Documentation. Meta Platforms, Inc. 2024. URL: https://react.dev/learn (дата звернення: 25.04.2026).

13. Tailwind CSS Documentation. Tailwind Labs Inc. 2024. URL: https://tailwindcss.com/docs (дата звернення: 25.04.2026).

14. Vite Guide. Evan You та учасники. 2024. URL: https://vitejs.dev/guide/ (дата звернення: 26.04.2026).

15. Alchemy Developer Documentation. Alchemy Insights, Inc. 2024. URL: https://docs.alchemy.com/ (дата звернення: 27.04.2026).

16. CoinGecko API Documentation v3. CoinGecko. 2024. URL: https://www.coingecko.com/en/api/documentation (дата звернення: 27.04.2026).

17. Web3 Secret Storage Definition. Ethereum Wiki. URL: https://ethereum.org/en/developers/docs/data-structures-and-encoding/web3-secret-storage/ (дата звернення: 28.04.2026).

18. Tanenbaum A. S., Van Steen M. Distributed Systems: Principles and Paradigms. 3rd ed. Pearson, 2017. 715 p.

19. Antonopoulos A. M. Mastering Bitcoin: Programming the Open Blockchain. 2nd ed. O'Reilly Media, 2017. 372 p.

20. FIPS PUB 197. Advanced Encryption Standard (AES). National Institute of Standards and Technology. 2001.

21. Koblitz N. Elliptic curve cryptosystems. Mathematics of Computation. 1987. Vol. 48, № 177. P. 203–209.

ДОДАТОК А

Лістинг ключового модуля WalletService.js

У додатку наведено повний код центрального модуля гаманця, що реалізує всю взаємодію з блокчейном – генерацію ключів, шифрування, відправку транзакцій, роботу з токенами та NFT.

```
import { ethers } from "ethers";

// ─── NETWORK CONFIGURATION ──────────────────────────────────────

const ALCHEMY_KEY = import.meta.env.VITE_ALCHEMY_KEY;

export const NETWORKS = {
  sepolia: {
    id: "sepolia", name: "Sepolia Testnet", chainId: 11155111,
    rpc: `https://eth-sepolia.g.alchemy.com/v2/${ALCHEMY_KEY}`,
    nftApi: `https://eth-sepolia.g.alchemy.com/nft/v3/${ALCHEMY_KEY}`,
    explorer: "https://sepolia.etherscan.io",
    symbol: "ETH", coingeckoId: "ethereum", testnet: true,
  },
  mainnet: {
    id: "mainnet", name: "Ethereum Mainnet", chainId: 1,
    rpc: `https://eth-mainnet.g.alchemy.com/v2/${ALCHEMY_KEY}`,
    nftApi: `https://eth-mainnet.g.alchemy.com/nft/v3/${ALCHEMY_KEY}`,
    explorer: "https://etherscan.io",
    symbol: "ETH", coingeckoId: "ethereum", testnet: false,
  },
  polygon: {
    id: "polygon", name: "Polygon", chainId: 137,
    rpc: `https://polygon-mainnet.g.alchemy.com/v2/${ALCHEMY_KEY}`,
    nftApi: `https://polygon-mainnet.g.alchemy.com/nft/v3/${ALCHEMY_KEY}`,
    explorer: "https://polygonscan.com",
```

```
    symbol: "MATIC", coingeckoId: "matic-network", testnet: false,
  },
};

let _net = NETWORKS[localStorage.getItem("nexavault_network") || "sepolia"];
export const getCurrentNetwork  = () => _net;
export const getProvider        = () => new ethers.JsonRpcProvider(_net.rpc);

// ─── GAS ──────────────────────────────────────────────────────────────────

const MULTIPLIERS = { low: 80n, medium: 100n, high: 150n };

export const getFeeOptions = async () => {
  const feeData  = await getProvider().getFeeData();
  const maxFee   = feeData.maxFeePerGas         ?? ethers.parseUnits("20","gwei");
  const priority = feeData.maxPriorityFeePerGas ?? ethers.parseUnits("1", "gwei");
  return Object.fromEntries(Object.entries(MULTIPLIERS).map(([t, m]) => [t, {
    maxFeePerGas: maxFee * m / 100n,
    maxPriorityFeePerGas: priority * m / 100n,
  }]));
};

// ─── SECURITY ─────────────────────────────────────────────────────────────

export const encryptPrivateKey = async (pk, pw) =>
  new ethers.Wallet(pk).encrypt(pw);

export const decryptPrivateKey = async (json, pw) =>
  (await ethers.Wallet.fromEncryptedJson(json, pw)).privateKey;

// ─── WALLET MANAGEMENT ────────────────────────────────────────────────────

export const createNewWallet = () => {
  const w = ethers.Wallet.createRandom();
```

```
  return { address: w.address, privateKey: w.privateKey,
           mnemonic: w.mnemonic?.phrase || "" };
};

export const importFromMnemonic = (m) => {
  try {
    const w = ethers.Wallet.fromPhrase(m.trim());
    return { address: w.address, privateKey: w.privateKey, mnemonic: m.trim() };
  } catch { throw new Error("Невірна мнемонічна фраза"); }
};

// ... повний код доступний у репозиторії на GitHub
```

ДОДАТОК Б

Фрагмент коду компонента інтерфейсу (Dashboard.jsx)

Нижче наведено фрагмент основного компонента дашборда, що демонструє декларативний підхід React.

```
import { useState, useEffect } from "react";
import { Send, Download, Repeat } from "lucide-react";
import * as WalletService from "../WalletService";

export default function Dashboard({ address, balance, txHistory,
                                    onRefresh, setScreen, nativePrice }) {
 const [tokens, setTokens] = useState([]);

 useEffect(() => {
  if (!address) return;
  WalletService.getTokenBalances(address).then(setTokens);
 }, [address]);

 const usdValue = (parseFloat(balance) * nativePrice).toFixed(2);

 return (
  <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
   <Card className="p-6">
    <div className="text-slate-500 text-xs">МІЙ ГАМАНЕЦЬ</div>
    <div className="font-bebas text-4xl mt-2">{balance}</div>
     <div className="text-cyan-400 mt-1">≈ ${usdValue}</div>

    <div className="grid grid-cols-2 gap-2 mt-5">
     <Btn onClick={() => setScreen("send")}>
      <Send size={14} /> Надіслати
     </Btn>
     <Btn v="ghost" onClick={() => setScreen("receive")}>
```

```
            <Download size={14} /> Отримати
          </Btn>
        </div>
      </Card>

      {/* Token list, Portfolio chart, NFT preview ... */}
    </div>
  );
}
```

ДОДАТОК В

ДЕКЛАРАЦІЯ ПРО АКАДЕМІЧНУ ДОБРОЧЕСНІСТЬ ТА ВИКОРИСТАННЯ ШІ

Я, Драбовський Богдан Сергійович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1) Моя бакалаврська кваліфікаційна робота на тему «Розробка блокчейн-гаманця з функціоналом підтримки NFT» є результатом моєї особистої праці.
2) Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3) Щодо використання інструментів штучного інтелекту зазначаю, що інструменти ШІ були використані для:

– Claude (Anthropic) — для допомоги в написанні та налагодженні програмного коду, генерації структури документації, стилістичного редагування та перевірки граматики;
– GitHub Copilot — для автодоповнення коду під час розробки.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«___» _____ 2026 р. _____ **Богдан ДРАБОВСЬКИЙ**